School/workshop on Logics and Models for Concurrency
Springer Lecture Notes in Computer Science 354

Expressibility Results for Linear-Time and Branching-Time Logics

E. M. Clarke, I. A. Draghicescu

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract We investigate the expressive power of linear-time and branching-time temporal logics as fragments of the logic CTL*. We give a simple characterization of those CTL* formulas that can be expressed in linear-time logic. We also give a simple method for showing that certain CTL* formulas cannot be expressed in the branching-time logic CTL. Both results are illustrated with examples.

key words: temporal logic, linear-time logic, branching-time logic, computation tree logics, fairness

Contents

[0]This research was partially supported by NSF grant CCR-87-226-33.

1. Introduction

Temporal Logics are widely used for reasoning about concurrent programs and reactive systems. We will model such programs by labelled state-transition graphs, called *Kripke structures* [8]. If some state is designated as the *initial* state, then the Kripke structure can be unwound into an infinite tree with that state as the root. Since paths in the tree represent possible computations of the program, we will refer to the infinite tree obtained in this manner as the *computation tree* of the program. Temporal logics may differ according to how they handle branching in the underlying computation tree. In *linear-time logic*, operators are provided for describing events along a single computation path. In a *branching-time logic* the temporal operators quantify over the paths that are possible from a given state. Each type of logic has its advantages and disadvantages. Testing satisfiability for linear-time formulas appears easier than for branching-time formulas [1], while automatic verification techniques based on *model checking* [5] have lower complexity in the case of branching-time logics.

Lamport [9] was the first to investigate the expressive power of the two types of temporal logic from the perpsective of computer science. His 1980 POPL paper discussed two logics: a simple linear-time logic and a simple branching-time logic. He showed that each logic could express certain properties that could not be expressed in the other. For example, branching-time logic cannot express certain natural fairness properties that can easily expressed in the linear-time logic. Linear-time logic, on the other hand, cannot express the possibility of event occuring at sometime in the future along some computation path, even though this can be expressed in the branching time logic. There were some difficulties with the method that Lamport used for obtaining these results, however. In particular, his approach was somewhat like comparing "apples and oranges", since the truth of a branching-time formula was determined with respect to a state while the truth of a linear-time formula was determined with respect to an individual computation path. This resulted in a notion of *expressive equivalence* that classified some satisfiable formulas as being equivalent to *false*.

Emerson and Halpern [7] provided a uniform framework for investigating this question. They formulated the problem in terms of the expressive power of various fragments of a single logic called CTL* (Computation Tree Logic), which was first discussed in [4] and [6]. This logic combines both linear-time and branching-time operators; its syntax is given in terms of *path formulas* that are interpreted over computation paths and *state formulas* that are true or false in a state. The path formulas use the standard temporal operators **G** (*always*), **F** (*sometimes*), **X** (*nexttime*), and **U** (*until*) and are like the formulas of traditional linear-time logic except that both atomic propositions and state formulas are allowed as primitive components of formulas. A state formula may be obtained from a path formula by prefixing it with a *path quantifier* that can either be an "A" (*for every path*) or a "E" (*there exists a path*). Linear-time logic (LTL) is identified with the set of all CTL* state formulas that have the form **A***f* where *f* is a path formula that does not contain any state sub-formulas. The branching-time part (called CTL) consists of all CTL* state formulas in which every linear-time operator is immediately preceded by a path quantifier.

Since both LTL and CTL consist entirely of state formulas, Emerson and Halpern were able to avoid the uniform framework problem in Lamport's paper. They showed that there exists a formula of LTL that cannot be expressed in CTL and vice versa. In general, the proofs of their inexpressibility results are quite long and complicated. For example, the proof that the linear-time formula for strong fairness is not expressible in CTL uses a complicated inductive argument that requires 3 1/2 journal pages to present. Furthermore, the technique that they use in this proof does not easily generalize to other examples. One reason for this difficulty is that formulas, which superficially appear similar, can have very different properties. For example, although **A**(**FG***p*) cannot be expressed as a CTL formula, **A**(**GF***p*) can be expressed as a CTL formula and in fact is equivalent to **AG**(**AF***p*). Likewise, the CTL formula **AG**(**AF***p*) is expressible in LTL since it is equivalent to **A**(**FG***p*), but the formula **AF**(**AG***p*), obtained by reversing the operators **AF** and **AG**, is not expressible in LTL.

Our paper gives a simple characterization of those CTL* formulas that can be expressed in LTL. We show that a CTL* formula f can be expressed in LTL if and only if it is equivalent to the formula $\mathbf{A}f'$ where f' is obtained from f by deleting the path quantifiers. We also give a necessary condition that a CTL* formula must satisfy in order to be expressible in CTL. The condition is formulated in terms of models that are labelled state transition graphs with *fairness constraints* . Intuitively, a CTL formula is unable to distinguish between two such models when the second is obtained from the first by adding a fairness constraint that extends some constraint of the first model. By using these two results we are able to give simple arguments to show that a number of example formulas cannot be expressed in LTL (in CTL). An additional advantage of our approach is that it provides insight into why CTL and LTL have different expressive powers.

The paper is organized as follows: In Section 2 we describe the logics LTL, CTL and CTL*. Section 3 contains the characterization of those CTL* formulas that can be expressed in LTL. Section 4 gives the necessary condition that a CTL* formula must satisfy in order to be expressible in CTL. It also contains several examples that show how this result can be used to give simple proofs that certain properties like strong fairness cannot be expressed in CTL. The paper concludes in Section 5 with a discussion of some remaining open problems.

2. Computation Tree Logics (CTL, LTL, and CTL*)

There are two types of formulas in CTL*: *state formulas* (which are true in a specific state) and *path formulas* (which are true along a specific path). Let *AP* be the set of atomic proposition names. A state formula is either:

- A, if $A \in AP$.
- If f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas.
- If f is a path formula, then $\mathbf{E}f$ is a state formula.

A path formula is either:

- A state formula.
- If f and g are path formulas, then $\neg f, f \vee g$, $\mathbf{X}f$, and $f\mathbf{U}g$ are path formulas.

CTL* is the set of state formulas generated by the above rules.

CTL ([2], [4]) is a restricted subset of CTL* that permits only branching-time operators–each path quantifier must be immediately followed by exactly one of the operators **G**, **F**, **X**, or **U**. More precisely, CTL is the subset of CTL* that is obtained if the path formulas are restricted as follows:

- If f and g are state formulas, then $\mathbf{X}f$ and $f\mathbf{U}g$ are path formulas.
- If f is a path formula, then so is $\neg f$.

Linear temporal logic (LTL), on the other hand, will consist of formulas that have the form $\mathbf{A}f$ where f is a path formula in which the only state subformulas that are permitted are atomic propositions. More formally, a path formula is either

- An atomic proposition.
- If f and g are path formulas, then $\neg f, f \vee g$, $\mathbf{X}f$, and $f\mathbf{U}g$ are path formulas.

We define the semantics of CTL* with respect to a structure $M = (\mathcal{S}, \mathcal{R}, \mathcal{L})$, where

- $\mathcal{S}$ is a set of states.
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation, which must be total. We write $s_1 \rightarrow s_2$ to indicate that $(s_1, s_2) \in \mathcal{R}$.
- $\mathcal{L} : \mathcal{S} \rightarrow \mathcal{P}(AP)$ is a function that labels each state with a set of atomic propositions true in that state.

Unless otherwise stated, all of our results apply only to *finite* Kripke structures.

We define a *path in M* to be a sequence of states, $\pi = s_0 s_1 \ldots$ such that for every $i \geq 0$, $s_i \rightarrow s_{i+1}$. π^i will denote the *suffix* of π starting at s_i.

We use the standard notation to indicate that a state formula f holds in a structure: $M, s \models f$ means that f holds at state s in structure M. Similarly, if f is a path formula, $M, \pi \models f$ means that f holds along path π in structure M. The relation $\models$ is defined inductively as follows (assuming that f_1 and f_2 are state formulas and g_1 and g_2 are path formulas):

1.	$s \models A$	*iff*	$A \in L(s)$.
2.	$s \models \neg f_1$	*iff*	$s \not\models f_1$.
3.	$s \models f_1 \vee f_2$	*iff*	$s \models f_1$ or $s \models f_2$.
4.	$s \models \mathbf{E}(g_1)$	*iff*	there exists a path π starting with s such that $\pi \models g_1$.
5.	$\pi \models f_1$	*iff*	s is the first state of π and $s \models f_1$.
6.	$\pi \models \neg g_1$	*iff*	$\pi \not\models g_1$.
7.	$\pi \models g_1 \vee g_2$	*iff*	$\pi \models g_1$ or $\pi \models g_2$.
8.	$\pi \models \mathbf{X} g_1$	*iff*	$\pi^1 \models g_1$.
9.	$\pi \models g_1 \mathbf{U} g_2$	*iff*	there exists a $k \geq 0$ such that $\pi^k \models g_2$ and for all $0 \leq j < k$, $\pi^j \models g_1$.

We will also use the following abbreviations in writing CTL* (CTL and LTL) formulas:

•$f \wedge g$	$\equiv$	$\neg(\neg f \vee \neg g)$	•$\mathbf{A}(f)$	$\equiv$	$\neg\mathbf{E}(\neg f)$
•$\mathbf{F}f$	$\equiv$	$\mathit{true}\mathbf{U}f$	•$\mathbf{G}f$	$\equiv$	$\neg\mathbf{F}\neg f$.

The necessary condition for expressability in CTL is given for Kripke structures with fairness consraints. The fairness constraints are specified in essentially the same way as the acceptance sets for Muller automata [10]. A *Kripke structure with fairness constraints* is a 4-tuple $M = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$ where

- $\mathcal{S}, \mathcal{R}, \mathcal{L}$ are as in the definition of the standard Kripke structures.
- $\mathcal{F} \subseteq 2^S$ is a set of fairness constraints.

Let $M = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$ be a Kripke structure with fairness constraints and $\pi = s_0 s_1 \ldots$ a path in M. Let $\mathit{inf}(\pi)$ denote the set of states occurring infinitely often on π. π is fair iff $\mathit{inf}(\pi) \in \mathcal{F}$.

The semantics of CTL* with respect to a Kripke structure with fairness constraints $M = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$ is defined using only the fair paths of the structure. Thus, the relation $\models$ is defined inductively for all states s and fair paths π of M using the same clauses as in the case of ordinary CTL* except the clause 4 is replaced by

4′. $s \models \mathbf{E}(g_1)$ *iff* there exists a fair path π starting with s such that $\pi \models g_1$.

3. Linear Time

For every $n \geq 0$, let $\sim_n$ be the equivalence relation over infinite paths given by

$$\sigma' \sim_n \sigma'' \quad \textit{iff} \quad \text{for any linear formula } f \text{ with } \textit{length}(f) \leq n,\ \sigma' \models f \iff \sigma'' \models f$$

Lemma 1 *Suppose AP, the set of atomic propositions is finite. Let M be a Kripke structure and σ a path in M. Let $n \geq 0$.*

Then there exists a prefix xy of σ such that xy^ω is an infinite path in M and $\sigma \sim_n xy^\omega$.

Proof: It will be given in the completed version.

If ϕ is a CTL* formula, we will denote by ϕ^d the linear formula obtained from ϕ by deleting all its path quantifiers. For instance, if $\phi = \mathbf{AG}(p\mathbf{U}(\mathbf{EX}q))$ then $\phi^d = \mathbf{G}(p\mathbf{U}(\mathbf{X}q))$.

For a Kripke structure M and a path $\sigma = s_0 s_1 \ldots s_{i-1}(s_i \ldots s_{j-1})$ in M we will denote by $M(\sigma)$ the single-path Kripke structure defined by σ. $M(\sigma) = (\mathcal{S}(\sigma), \mathcal{R}(\sigma), \mathcal{L}(\sigma))$, where :
$\mathcal{S}(\sigma) = \{\overline{s}_0, \ldots, \overline{s}_{j-1}\}$
$\mathcal{R}(\sigma) = \{(\overline{s}_0, \overline{s}_1), \ldots, (\overline{s}_{j-2}, \overline{s}_{j-1}), (\overline{s}_{j-1}, \overline{s}_i)\}$
$\mathcal{L}(\sigma) : S(\sigma) \rightarrow 2^{AP}$, $\mathcal{L}(\sigma)(\overline{s}_k) = \mathcal{L}(s_k)$

Let us notice that for any path of the form xy^ω of a Kripke structure M and for any CTL* formula ϕ, we have

$$M(xy^\omega), \overline{s}_0 \models \phi \quad \textit{iff} \quad M(xy^\omega), xy^\omega \models \phi^d$$

Theorem 1 *Let ϕ be a CTL* state formula.*

Then ϕ is expressible in LTL iff ϕ is equivalent to $\mathbf{A}\phi^d$.

Proof: Suppose that ϕ is equivalent to $\mathbf{A}f$, where f is a linear formula. We have to show that ϕ is equivalent to $\mathbf{A}\phi^d$.

Let M be a Kripke structure and s_0 a state in M. We have :

$$\begin{array}{llll}
M, s_0 \models \phi & \text{iff} & \text{for all paths } \sigma \text{ in } M, & M, \sigma \models f \\
 & \text{iff} & \text{for all paths of the form } xy^\omega \text{ in } M, & M, xy^\omega \models f \\
 & & \text{(by Lemma 1)} & \\
 & \text{iff} & \text{for all paths of the form } xy^\omega \text{ in } M, & M(xy^\omega), xy^\omega \models f \\
 & \text{iff} & \text{for all paths of the form } xy^\omega \text{ in } M, & M(xy^\omega), \overline{s_0} \models \phi \\
 & \text{iff} & \text{for all paths of the form } xy^\omega \text{ in } M, & M(xy^\omega), xy^\omega \models \phi^d \\
 & & \text{(as noticed above)} & \\
 & \text{iff} & \text{for all paths of the form } xy^\omega \text{ in } M, & M, xy^\omega \models \phi^d \\
 & \text{iff} & \text{for all paths } \sigma \text{ in } M, & M, \sigma \models \phi^d \\
 & & \text{(by Lemma 1)} & \\
 & \text{iff} & M, s_0 \models \mathbf{A}\phi^d &
\end{array}$$

Theorem 2 *Let ϕ be a CTL* formula.*

Then ϕ is expressible in LTL iff there exists a set $\mathcal{P}$ of paths such that

$M, s_0 \models \phi$ *iff for any path σ starting in s_0, there exists a path $\sigma' \in \mathcal{P}$ such that* $\sigma \sim_{length(\phi)} \sigma'$

Proof: Suppose ϕ is expressible in LTL. Then, by Theorem 1, ϕ is equivalent to $A\phi^d$. Let $\mathcal{P} = \{\sigma \mid \sigma \models \phi^d\}$.

For any Kripke structure M and any state s_0 in M, we have

$M, s_0 \models \phi$ iff for any path σ in M starting in s_0, $M, \sigma \models \phi^d$
iff for any path σ in M staring in s_0, $\sigma \in \mathcal{P}$
iff for any path σ in M staring in s_0, there exists a path $\sigma' \in \mathcal{P}$ such that $\sigma \sim_{length(\phi)} \sigma'$
(as $\sigma \sim_{length(\phi)} \sigma'$ and $\sigma' \in \mathcal{P}$ imply, by the definition of $\mathcal{P}$, that $\sigma \in \mathcal{P}$)

In order to prove the converse, suppose $\mathcal{P}$ is a set of paths with the following property :

$M, s_0 \models \phi$ iff for any path σ starting in s_0, there exists a path $\sigma' \in \mathcal{P}$ such that $\sigma \sim_{length(\phi)} \sigma'$.

By Theorem 1, it is enough to show that $M, s_0 \models \phi \iff M, s_0 \models A\phi^d$.

Suppose that $M, s_0 \models \phi$. Then, by the above property of $\mathcal{P}$, for any path $\sigma = xy^\omega$ in M starting in s_0, there exists a path $\sigma' \in \mathcal{P}$ such that $\sigma \sim_{length(\phi)} \sigma'$. Thus, for any $\sigma = xy^\omega$, the unique path of $M(\sigma)$ is $\sim_{length(\phi)}$-equivalent with some path in $\mathcal{P}$. Using again the property of $\mathcal{P}$; we obtain that $M(\sigma), \overline{s}_0 \models \phi$. This implies that for any $\sigma = xy^\omega$, $M, \sigma \models \phi^d$. Therefore, by Lemma 1, for any path σ in M starting in s_0, $M, \sigma \models \phi^d$, which implies $M, s_0 \models A\phi^d$.

Suppose $M, s_0 \models A\phi^d$. In particular, for any path xy^ω in M starting in s_0, $M(xy^\omega), xy^\omega \models \phi^d$, which implies $M(xy^\omega), \overline{s}_0 \models \phi$ and therefore there exists $\sigma' \in \mathcal{P}$ such that $xy^\omega \sim_{length(\phi)} \sigma'$. Thus, by Lemma 1, for any path σ in M starting in s_0, there exists a path $\sigma' \in \mathcal{P}$ such that $\sigma \sim_{length(\phi)} \sigma'$. Therefore $M, s_0 \models \phi$.

Using the above characterizations, it is easy to check, for instance, whether **AFAG**p is expressible in LTL.

Consider the Kripke structures shown in Figure 1,
$M = (\{s_0, s_1, s_2\}, \{(s_0, s_0), (s_0, s_1), (s_1, s_2), (s_2, s_2)\}, L)$, where $\mathcal{L}(s_0) = \mathcal{L}(s_2) = \{p\}$ and $\mathcal{L}(s_1) = \{\neg p\}$,
$M_0 = (\{s_0\}, \{(s_0, s_0)\}, \mathcal{L}\mid_{M_0}\})$,
$M_j = (\{t_1, \ldots, t_j, s_1, s_2\}, \{(t_1, t_2), \ldots, (t_{j-1}, t_j), (t_j, s_1), (s_1, s_2), (s_2, s_2)\}, \mathcal{L}_j)$, for any $j \geq 1$,
where $callL_j(t_k) = \mathcal{L}(s_2) = \{p\}$ and $\mathcal{L}_j(s_1) = \{\neg p\}$.

It is easy to see that $M, s_0 \not\models$ **AFAG**p but $M, s_0 \models$ A((**AFAG**$p)^d$). This implies, by Theorem 1, that **AFAG**p is not expressible in LTL.

We also have $M_0, s_0 \models$ **AFAG**p and for any $j \geq 1$, $M_j, t_1 \models$ **AFAG**p but $M, s_0 \not\models$ **AFAG**p. As any path of M is $\sim_{length(\mathbf{AFAG}p)}$-equivalent to a path in some M_j, $j \geq 0$, we obtain again, by Theorem 2 this time, that **AFAG**p is not expressible in LTL.

4. Branching Time

A strongly connected component C of a directed graph $G = (\mathcal{V}, \mathcal{R})$ is *non-trivial* if either $\mid C \mid > 1$ or $C = \{c\}$ and c has a self loop—i.e. $(c, c) \in R$. If $M = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$ is a Kripke structure with fairness constraints, then we can assume without loss of generality that each set $F \in \mathcal{F}$ determines a non-trivial strongly connected

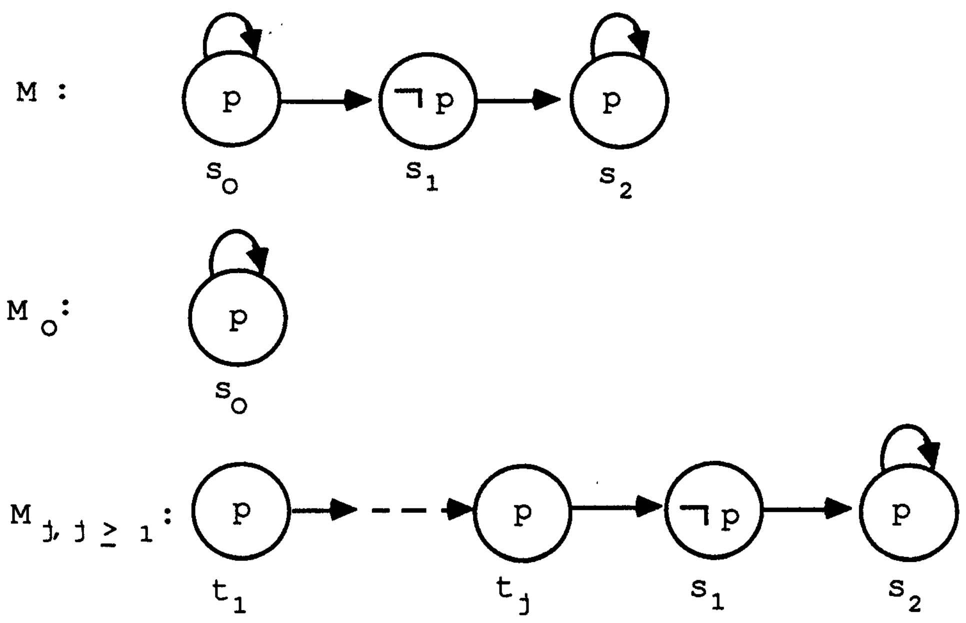

Figure 1: Kripke Structures for **AFAG**p

subgraph of the graph of M. If $\mathcal{F}$ and $\mathcal{F}'$ are two sets of fairness constraints, then we will say that $\mathcal{F}'$ *extends* $\mathcal{F}$ if $\mathcal{F}' = \mathcal{F} \cup \{F'\}$ where F' is a superset of some set $F \in \mathcal{F}$.

Theorem 3 *Let $M = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$ be a Kripke structure with fairness constraints, and let $M' = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F}')$ where the set of constraints $\mathcal{F}'$ extends $\mathcal{F}$. Then for all CTL formulas f and all states $s \in S$,*

$$M, s \models f \quad \text{iff} \quad M', s \models f$$

Proof: We prove the theorem by induction on the structure of f. We have the following cases:

- f is an atomic proposition: This case is trivial.
- $f = f_1 \vee f_2$ or $f = \neg f_1$: This case follows directly from the inductive hypothesis.
- $f = \mathbf{EX}f_1$ or $f = \mathbf{E}[f_1 \mathbf{U} f_2]$: We consider $f = \mathbf{E}[f_1 \mathbf{U} f_2]$; the other case is similar. We first show that the set of finite prefixes of the M-fair paths coincides with the set of finite prefixes of M'-fair paths. To see that this is true let P be the set of prefixes of M-fair paths that start at s and let P' be the corresponding set for M'. We must show that $P = P'$. It is easy to see that $P \subseteq P'$. Since $\mathcal{F} \subseteq \mathcal{F}'$, it must be the case that every M-fair path starting at s is also M'-fair path. To show that $P' \subseteq P$, let $p' \in P'$. Assume that p' is a prefix of some M'-fair path π'. If $inf(\pi') \in \mathcal{F}$, then π' is also an M-fair path and $p \in P$. If $inf(\pi') = F'$, then π must pass infinitely often through F since $F \subseteq F'$. Let p be a prefix of π' that includes all of p' and ends in a state of F. Since F determines a nontrivial strongly connected component of the graph of M, we can extend p to an M-fair path π such that $inf(\pi) = F$. Consequently, $p \in P$.

  Assume that $M, s \models \mathbf{E}[f_1 \mathbf{U} f_2]$. There must be a M-fair path π that starts at s such that for some $k \geq 0$ $M, \pi^k \models f_2$ and for all $0 \leq j < k$, $M, \pi^k \models f_1$. By the above observation there is an M'-fair path π' that has the same prefix of length k as π. By the inductive hypothesis $M', (\pi')^k \models f_2$ and for all $0 \leq j < k$, $M', (\pi')^k \models f_1$. It follows that $M', \pi' \models f_1 \mathbf{U} f_2$ and that $M', s \models \mathbf{E}[f_1 \mathbf{U} f_2]$. Exactly the same argument can be used to show that if $M', s \models \mathbf{E}[f_1 \mathbf{U} f_2]$, then $M, s \models \mathbf{E}[f_1 \mathbf{U} f_2]$.

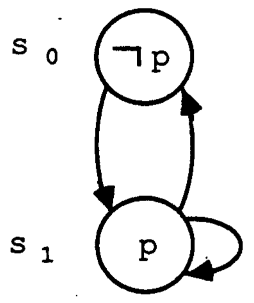

Figure 2: Kripke Structure for A(**FG**p)

- f = **EG**f_1: If $M, s \models$ **EG**f_1 then, as any M-fair path is also a M'-fair path, it follows by the inductive hypothesis that $M, s \models$ **EG**f_1. For the other direction suppose that $M', s \models$ **EG**f_1 and let π be the M'-fair path that satisfies **G**f_1. If $inf(\pi) \in \mathcal{F}$ then we are done. Otherwise $inf(\pi) = F'$ and F' is strongly connected. As $F \subseteq F'$ is also strongly connected, there exists a path π_1 starting in s such that $inf(\pi_1) = F$ and any state on π_1 is also on π. It follows π_1 is M-fair and, by inductive hypothesis, $M, \pi_1 \models$ **G**f_1, which implies $M, s \models$ **EG**f_1.

We illustrate how the Theorem 3 can be used to prove that A(**FG**p) is not expressible in CTL. Let M be the Kripke structure shown in Figure 2 with the fairness constraint $\mathcal{F} = \{\{s_1\}\}$. The set $\{s_1\}$ determines a non-trivial strongly connected component of the graph of M. A(**FG**p) is true in state s_0 of M, since all fair paths must eventually loop forever in state s_1. The set $\{s_0, s_1\}$ is certainly a superset of the set $\{s_1\}$. If we let $\mathcal{F}' = F \cup \{\{s_0, s_1\}\}$ and M' be the corresponding Kripke structure with $\mathcal{F}'$ replacing $\mathcal{F}$, then M and M' will satisfy the same CTL formulas. However, A(**FG**p) is not true in state s_0 of M' since the path $\pi = s_0 s_1 s_0 s_1 \ldots$ is fair, but does not satisfy the path formula **FG**p. It follows that no CTL formula is equivalent to A(**FG**p).

The same two Kripke structures M and M' can be used to show that the formula **AF**($p \wedge$ **X**p) is not expressible in CTL. If π is a fair path in M, then p must hold almost always on π. Consequently, $\pi \models$ **F**($p \wedge$ **X**p). It follows that **AF**($p \wedge$ **X**p) is true in state s_0 of M. However, $\pi' = s_0 s_1 s_0 s_1 \ldots$ is a fair path in M' that does not satisfy **F**($p \wedge$ **X**p), so **AF**($p \wedge$ **X**p) is false in state s_0 of M'.

5. Conclusion

In the linear-time case we have obtained two necessary and sufficient conditions for a CTL* formula to be expressible in LTL. In the branching-time case we have only given a necessary condition for a CTL* formula to be expressible in CTL. It would be useful to have a complete characterization in this case as well. One possibility would be to prove the converse for Theorem 3, which we state as a conjecture below:

Conjecture 1 *If f is not expressible in CTL, then it is possible to find two Kripke structures $M = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F})$ and $M' = (\mathcal{S}, \mathcal{R}, \mathcal{L}, \mathcal{F}')$ with $\mathcal{F}'$ an extension of $\mathcal{F}$ such that for some state $s \in S$*

$$\textit{either} \quad M, s \models f \textit{ and } M', s \not\models f \quad \textit{or} \quad M, s \not\models f \textit{ and } M', s \models f.$$

So far, we have been unable to prove or disprove this conjecture. If it is true, we believe that the proof is likely to be difficult.

Another problem with the result in Section 4 is that it is possible to have a CTL* formula that is equivalent to *false* over ordinary Kripke models and, therefore, is expressible in CTL, but is not expressible in CTL when the models are fair Kripke structures. In order to construct such an example we use a result from [3], which shows that it is possible to completely characterize an ordinary Kripke structure in the logic CTL. Let M and M' be two Kripke structures. Let s_0 be a state of M and s'_0 be a state of M'. Then M, s_0 is *CTL*-equivalent* to M', s'_0 iff for all CTL* formulas f, $M, s_0 \models f$ *iff* $M', s'_0 \models f$.

Given a Kripke structure M and a state s_0 of M, there is a CTL formula $C(M, s_0)$ such that $M', s'_0 \models C(M, s_0)$ iff M, s_0 is CTL*-equivalent to M', s'_0. For the model shown in Figure 2, $C(M, s_0)$ is given by

$$p \land \mathbf{AG}(p \rightarrow (\mathbf{EX}(\neg p) \land \mathbf{AX}(\neg p)) \land \mathbf{AG}(\neg p \rightarrow (\mathbf{EX}(\neg p) \land \mathbf{EX}(p)).$$

Now, consider the formula $C(M, s_0) \land \mathbf{A}(\mathbf{FG}p)$. This formula is equivalent to *false* if the models are ordinary Kripke structures. Since $\mathbf{A}(\mathbf{FG}p)$ is false in M, s_0, it follows that if $M', s'_0 \models C(M, s_0)$ then $M', s'_0 \models \neg\mathbf{A}(\mathbf{FG}p)$. If we modify M to include the fairness constraint $\mathcal{F} = \{\{s_1\}\}$, then $C(M, s_0) \land \mathbf{A}(\mathbf{FG}p)$ is true in s_0. Thus, the formula is not equivalent to *false* over fair Kripke structures. Essentially the same argument as in the first example of Section 4 shows that it is not expressible in CTL in this case. It would be useful to have a version of Theorem 3 that applied to ordinary Kripke structures and avoided such pathological examples.

References

[1] E.M. Clarke A.P. Sistla. Complexity of propositional temporal logics. *Journal of the Association for Computing Machinery*, 32(3):733–749, July 1986.

[2] M. Ben-Ari, Z. Manna, and A. Pneuli. The temporal logic of branching time. In *8th Annual ACM Symp. on Principles of Programming Languages*, pages 164–177, 1981.

[3] M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing Kripke structures in temporal logic. In *1987 Colloquium on Trees in Algebra and Programming*, Pisa, Italy, March 1987.

[4] E.M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Proc. of the Workshop on Logic of Programs*, Springer-Verlag, Yorktown Heights, NY, 1981.

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

[6] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. System Sci.*, 30(1):1–24, 1985.

[7] E.A. Emerson and J.Y. Halpern. 'Sometimes' and 'not never' revisited: on branching versus linear time. In *Proc. 10th ACM Symp. on Principles of Programming Languages*, 1983.

[8] G.E. Hughes and M.J. Creswell. *An Introduction to Modal Logic*. Methuen and Co., 1977.

[9] L. Lamport. 'Sometimes' is sometimes 'not never'. In *Seventh Annual ACM Symposium on Principles of Programming Languages*, pages 174–185, Association for Computing Machinery, Las Vegas, January 1980.

[10] D. E. Muller. Infinite sequences and finite machines. In *Proc. 4th Annual IEEE Symposium of Switching Theory and Logical Design*, pages 3–16, 1963.