LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

Comparing Linear and Branching Time Temporal Logics

Comparing Linear and Branching Time Temporal Logics

by

Colin Stirling

LFCS Report Series

ECS-LFCS-87-24
(also published as CSR-229-87)

April 1987

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Comparing Linear and Branching Time Temporal Logics

Colin Stirling
Dept. of Computer Science
Edinburgh University

Abstract
An important division of temporal logics is into linear and branching time. Here we propose a general framework for modal and temporal logics and within it offer a formal criterion for distinguishing linear from branching logics. This distinction is based on the CTL* framework. We offer a sound and complete axiomatization of CTL* formulas and we also contrast the general expressiveness of linear and branching time logics.

February 1987

Introduction

An important division of temporal logics used for describing properties of programs is into linear and branching time: [MP] is an example of linear, [CE] of branching. Discussions of this contrast include [EH,Pn]. Here we propose a general framework for modal and temporal logics and within it we offer a formal criterion for distinguishing linear from branching logics based on the ideas of CTL* [EH]. Starting from transition systems as modal structures temporal logic arises when we are interested in formulas that express properties of paths through the transition system. For a variety of operators, temporal modalities, the occurrence of paths in their semantic definition is essential. Then an important division is into operators whose semantic clauses involve path switching, and those that don't: the latter are linear time operators. By appealing to a single quantifier modality to regulate path switching at a point to just those paths which also pass through it we capture a notion of branching time operator. Any linear time logic gives rise to a branching time version by adding the quantifier modality. This turns into a generalization of the framework of CTL*. An important difference, however, is that we do not distinguish between path and state formulas: all temporal formulas are interpreted on paths and then derivatively on states. The result is a more uniform framework. One benefit is that a sound and complete axiomatization of validity of CTL* formulas is just an extension of the axiom system for the underlying linear time logic.

The rest of the paper consists of general expressibility results. We ask: when do two states satisfy the same formulas? The answer for arbitrary linear time logics is in terms of traces while for branching time logics it appeals to bisimulations. We compare these answers with work on equivalences between processes.

There are three sections. Section 1 describes the general framework for modal and temporal logics and distinguishes branching from linear time logics. The axiomatization of CTL* formulas is also stated there. A sketch proof of its completeness is given in section 3 while section 2 covers the expressiveness results.

1. A Framework for Modal and Temporal Logics

Modal Logics

We begin with general modal structures, labelled transition systems.

Definition 1.1 A labelled transition system $T = (S, L, R)$ where
- i. S is a non-empty set (of states)
- ii. L is a non-empty set (of labels)
- iii. for each $a \in L \quad R_a \subseteq S \times S$

Typically S is a set of computational states: program states; states of a data base or states of reactive systems. However, it could also be states of knowledge. For each $a \in L \quad R_a$ is a transition relation: sR_as' represents that state s becomes s' due to, or under the influence of, a where a could be an action, a period of time, a person or whatever. Often we are not interested in the label of a transition so we use the notation sRs' for $\exists a \in L.\ sR_as'$. Examples of the extensive use of transition systems as operational models of activity are [Ke,He,Mi,Si].

Transition systems are also (generalized) Kripke modal frames or structures: possible worlds are replaced by the more concrete states and the single accessibility relation by a family of indexed transition relations. The propositional modal language below involves simple modalities for expressing transitional change. The operator $[\,]$ means 'after every possible transition' while its dual $\neg[\,]\neg$, abbreviated to $\langle\,\rangle$, means 'after some possible transition'. Moreover for each label a there is the modality $[a]$ meaning 'after every possible a-transition', and its dual $\langle a\rangle$. In abstract syntax form the modal language where Q denotes some atomic sentence and a some label is:

$$A \ ::= \ Q \mid \neg A \mid A \wedge A \mid [a]A \mid [\,]A$$

The language is really a family of languages parameterized by the atomic sentences and the set of labels. Each extends the language of propositional logic. Usual abbreviations apply $B \vee C$ for $\neg(\neg B \wedge \neg C)$; $B \rightarrow C$ for $\neg B \vee C$.

When L is the label set of a modal language it is interpreted on frames containing L. Like propositional logic we also need to interpret the atomic sentences. So a modal model $\mathcal{M} = (T, V)$ where V assigns subsets of $T's$ states to each atomic sentence Q : $V(Q)$ is the set of states where Q is deemed to be true under V. Truth of an arbitrary formula A of the language at a state s in T under $\mathcal{M}$ is inductively defined using the notation $s \models_{\mathcal{M}} A$. When A is false of a state s

under M we write $s \not\models_M A$.

$$
\begin{array}{l}
s \models_M Q \text{ iff } s \in V(Q) \\
s \models_M \neg A \text{ iff } s \not\models_M A \\
s \models_M A \wedge B \text{ iff } s \models_M A \text{ and } s \models_M B \\
s \models_M [a]A \text{ iff for all } s'. \text{ if } sR_as' \text{ then } s' \models_M A \\
s \models_M [\,]A \text{ iff for all } s'. \text{ if } sRs' \text{ then } s' \models_M A
\end{array}
$$

Sample intended interpretations are: when L is a set of programs and sR_as' represents program a starting in state s may terminate in s' then formulas of the form $A \rightarrow [a]B$ are generalized partial correctness formulas; where L is a set of people and sR_as' means in state s $a's$ knowledge is of s' then $[a]B$ represents a knows that B while $[\,]B$ can be understood as B is common knowledge.

Various elaborations are possible. Expressiveness may be increased by addition of new modalities. Natural candidates are the reverse operators $\overline{[\,]}, \overline{[a]}$:

$$
\begin{array}{l}
s \models_M \overline{[a]}A \text{ iff } \forall s' \in S. \text{ if } s'R_as \text{ then } s' \models_M A \\
s \models_M \overline{[\,]}A \text{ iff } \forall s' \in S. \text{ if } s'Rs \text{ then } s' \models_M A
\end{array}
$$

More generally we could define what a modal operator is as we do later for temporal modalities.

Secondly, since transition systems can be classified according to the properties of their transition relations, we may wonder how much of this classification is internally representable in terms of validity of modal formulas. For instance validity in T of all instances of $[a]A \rightarrow [a][a]A$ corresponds to the condition that R_a is transitive in T. The exact dividing line here is not clear: although well-foundedness and transitivity of R correspond to modal formulas image-finiteness of R (the condition on T that for each s $\{s'|sRs'\}$ is finite) doesn't. Instead, structure may be introduced into the label sets. The most studied case, dynamic logic, is when the label set is a Kleene algebra. The intended meanings of the operations are reflected in closure conditions on the transition relations: for instance $R_{a;b}$ is defined as $R_b \circ R_a$ where $\circ$ is relational composition. In turn these conditions correspond to modal formulas such as $[a;b]A \leftrightarrow [a][b]A$. Structure can also be introduced into the state set S. One possibility is to treat it as a first-order structure resulting in quantified modal logics. Another possibility is to structure it algebraically as in reactive systems.

Temporal Logics

Our interest, however, is temporal logics. These arise out of modal logics when modal formulas are viewed as properties of paths (or computations) through a transition system. Where $T = (S, L, R)$ then a path through T is a maximal sequence in both directions of the form:

$$\ldots s_i R_{a_i} s_{i+1} \ldots$$

with indices drawn from the integers. Maximality guarantees that if a path is finite then its initial state has no transitions into it and its final state has none emanating from it. We use σ to denote paths. Where σ is a path $L(\sigma)$ is its sequence of transition labels : for instance $L(\sigma) = \ldots a_i a_{i+1} \ldots$ for the path above. Sometimes we are not interested in the transition labels in which case we may assume that a path σ is a sequence of states $\ldots s_i s_{i+1} \ldots$ (with $s_j R s_{j+1}$ for all j). When no conditions are placed on the transition relations, sRs' ($sR_a s'$) represents s' is a possible next moment of s (via a). Now the modalities appear inexpressive, able only to express various modes of 'next' or 'previous': $[\,]$ ($[\overline{\ }]$) means 'at all possible next (previous) moments', and similarly for the indexed operators.

To increase natural expressiveness we may impose conditions on R (or define R' to be a closure of R and define the modal semantics with respect to it). When R is transitive (R' is transitive closure of R) $\langle\ \rangle A$ has the more interesting meaning; 'A will be true in some possible future'. While $[\ ]A$ means 'A will always be true throughout all possible futures'. (Similarly, $\overline{\langle\ \rangle}A$ means 'A was true in some past'.) Using paths we can reformulate the semantic clauses for $[\ ]$. A little notation: if $\sigma = ...s_i R_{ai} s_{i+1}...$ then $\sigma(j)$ for defined j is just s_j. Then

$$s \models_{\mathcal{M}} [\ ]A \text{ iff } \forall\sigma\forall i \text{ if } \sigma(i) = s \text{ then } \forall j > i\ \sigma(j) \models_{\mathcal{M}} A$$

However, this appeal to paths is virtual. The right hand side can be replaced with the usual definition. A notion that is not, in general, expressible in this modal language is: 'A will always be true throughout some possible future'. Let $[\Diamond]A$ represent it (with $\langle\Box\rangle A$ its dual, 'A will be true in every possible future'). Its semantic clause is:

$$s \models_{\mathcal{M}} [\Diamond]A \text{ iff } \exists\sigma\exists i\ \sigma(i) = s \text{ and } \forall j > i\ \ \sigma(j) \models_{\mathcal{M}} A$$

Now the presence of $\exists\sigma$ is not virtual: the right hand side is not equivalent to a first-order formula involving quantifiers over states, the relations R between states and $\models_{\mathcal{M}}$ between states and A. One solution is to impose more conditions on R: one possibility is placing a finite bound on the length of each path; another is to include irreflexivity and connectedness as conditons making $[\Diamond]$ and $[\ ]$ equivalent. A third solution, when no conditions are placed on R, is to include recursion in the modal language [EL, La]: for instance, the two eventually's can be expressed as $\mu X.\Diamond(A \vee X)$ and $\mu X(\Box(A \vee X) \wedge \Diamond tt)$ (where $\Diamond$, $\Box$ are modes of 'next' and tt is true).

Remaining within propositional modal logic we need to appeal to paths explicitly in our semantics to express interesting properties. We could just introduce a modal, or temporal language, with base modalities $[\ ]$ and $[\Diamond]$ (and their inverses) [BMP]. But there are still interesting properties of paths which are not expressible such as 'A will almost always be true throughout every future'. If $\#A$ expresses it then its semantic clause is:

$$s \models_{\mathcal{M}} \#A \text{ iff } \forall\sigma\forall i \text{ if } \sigma(i) = s \text{ then } \exists j > i \forall\kappa \geq j\ \ \sigma(\kappa) \models_{\mathcal{M}} A$$

(This modality is also not expressible in the modal μ-calculus.) Such considerations led [CE, EH] to propose CTL*, a general framework for temporal logics. This is the framework, with some fine tuning, we generalize below.

Given a transition system our interest is in paths through it. So let us make this aspect explicit by appealing to general transition systems [He].

Definition 1.2 A general transition system $T = (S, L, R, \Sigma)$ where

i. (S, L, R) is a transition system

ii. R and R^{-1} are total: $\forall s\ \exists s_1\ \exists s_2\ sRs_1$ and s_2Rs

iii. Σ is a non-empty set of paths through (S, L, R)

General transition systems will be our temporal structures. Note that paths are infinite in both directions. As in common in computation when we are not interested in reverse modalities the condition that R^{-1} is total is relaxed. We say that T is saturated if Σ consists of all possible paths through the transition system.

A propositional temporal language consists of atomic formulas closed under boolean and temporal operators. A temporal model for a language is a pair $\mathcal{M} = (T, V)$ where T is a general transition system and V, as in the modal case, assigns subsets of S to atomic formulas: $V(Q) \subseteq S$ for each Q. We then define inductively what it is for a path σ and an index i (the *ith* point in σ) to satisfy a formula in $\mathcal{M}$: we use the notation $\sigma, i \models_{\mathcal{M}} A$ for this. When A is not true of (σ, i) we write $\sigma, i \not\models_{\mathcal{M}} A$. As before we let $\sigma(i)$, which is different from the pair (σ, i), be the *ith* state of σ, a member of S. The semantic clauses for atomic formulas and boolean operations are:

$$\sigma, i \models_{\mathcal{M}} Q \text{ iff } \sigma(i) \in V(Q)$$
$$\sigma, i \models_{\mathcal{M}} \neg A \text{ iff } \sigma, i \not\models_{\mathcal{M}} A$$
$$\sigma, i \models_{\mathcal{M}} A \wedge B \text{ iff } \sigma, i \models_{\mathcal{M}} A \text{ and } \sigma, i \models_{\mathcal{M}} B$$

This just leaves the clauses for the temporal operators. So we define in general terms an n-ary temporal operator, where $L(\sigma, j)$ is the jth label of σ.

Definition 1.3 An n-ary temporal operator 0 is such that:

$\sigma, i \models_{\mathcal{M}} 0(A_1, ..., A_n)$ is determined in a fixed way from the values of
$\sigma', j \models_{\mathcal{M}} A_\kappa\ \ 1 \leq \kappa \leq n$ and
$L(\sigma', j)$ for all j and $\sigma' \in \Sigma$

Example operators are:

i. $\sigma, i \models_{\mathcal{M}} X_a A$ iff $L(\sigma, i) = a$ and $\sigma, i+1 \models_{\mathcal{M}} A$

ii. $\sigma, i \models_{\mathcal{M}} \langle a \rangle A$ iff $\exists \sigma' \in \Sigma\ \sigma'(j) = \sigma(i)$ and $L(\sigma', j) = a$ and $\sigma', j+1 \models_{\mathcal{M}} A$

iii. $\sigma, i \models_{\mathcal{M}} A \overline{U} B$ iff $\exists j < i (\sigma, j \models_{\mathcal{M}} B$ and $\forall \kappa : j < \kappa < i\ \ \sigma, \kappa \models_{\mathcal{M}} A)$

iv. $\sigma, i \models_{\mathcal{M}} \#A$ iff $\forall \sigma' \in \Sigma$ if $\sigma'(j) = \sigma(i)$ then $\exists \kappa > j \forall \kappa' \geq \kappa\ \ \sigma, \kappa' \models_{\mathcal{M}} A$

v. $\sigma, i \models_{\mathcal{M}} GA$ iff $\forall j > i\ \ \sigma, j \models_{\mathcal{M}} A$

As a derivative notion we define what it is for a state s to satisfy a formula A:

$$s \models_{\mathcal{M}} A \text{ iff } \forall \sigma \in \Sigma \forall i \text{ if } \sigma(i) = s \text{ then } \sigma, i \models_{\mathcal{M}} A$$

Moreover we say that a formula is true in $\mathcal{M}$, which we write as $\models_{\mathcal{M}} A$, if every state s in $\mathcal{M}$ satisfies A. Finally, we say that A is valid with respect to a family of general transition systems Λ if A is true in every model on every transition system in Λ. We express this using the notation $\Lambda \models A$. Notice that truth in a model and Λ-validity also cover modal logics when Λ is a family of transition systems.

Notice also that the definition of temporal operator includes the modalities of process logic [HKP] as well as the logics in [BR]. Process logics arise when the label set is structured as in dynamic logic. Likewise structuring the state set yields temporal logics for reactive systems (compare section 2 below).

Branching and Linear Time Temporal Logics

Our interest is in the contrast between linear and branching time temporal logics. We define first a linear time modality by placing a restriction on definition 1.3.

Definition 1.4 An n-ary linear time modality $\#$ is such that:
$\sigma, i \models_{\mathcal{M}} \#(A_1, ..., A_n)$ is determined in a fixed way from the values of
$\sigma, j \models_{\mathcal{M}} A_\kappa \;\; 1 \leq \kappa \leq n$ and $L(\sigma, j)$ for all j

The path σ remains fixed throughout the semantic clause of a linear time modality. A future time (past time) linear modality is one where 'for all $j \geq i (j \leq i)$' replaces 'for all j' on the right hand side. In contrast, a non-linear time modality may include reference to other paths. Examples i. iii and v above are of linear time modalities unlike the other pair. We say that a temporal logic is linear if all its modalities are linear time. Following [EH] we assume a single branching time modality, a path quantifier $\triangle$:

$$\sigma, i \models_{\mathcal{M}} \triangle A \;\; \text{iff} \;\; \forall \sigma' \in \Sigma \forall j \;\; \text{if} \;\; \sigma'(j) = \sigma(i) \;\; \text{then} \;\; \sigma', j \models_{\mathcal{M}} A$$

The idea is that $\triangle$ involves branching by being a quantifier over all paths passing through a particular state. Its dual $\triangledown$ is defined as $\neg \triangle \neg$. A basis branching time temporal language is a linear time language which is also closed under the monadic operator $\triangle$. Finally, we count as a branching time logic any logic whose modalities are definable in a basis language (but not in a linear time language). For instance, the modality in ii above $\langle a \rangle$ is definable as $\triangledown X_a$ while $\#$ in iv is $\triangle \neg G \neg G A$.

A standard future linear time temporal logic consists of the single modality U for 'until' [GPSS] with semantic clause

$$\sigma, i \models_{\mathcal{M}} AUB \;\; \text{iff} \;\; \exists j > i (\sigma, j \models_{\mathcal{M}} B \;\; \text{and} \;\; \forall \kappa : i < \kappa < j \;\; \sigma, \kappa \models_{\mathcal{M}} A)$$

Where tt is true and ff false we can define the usual modalities in terms of U:

next:	$XA = ff\ UA$
it will be:	$FA = tt\ UA$
it will always be:	$GA = \neg F \neg A$

[GPSS] offers a sound and complete axiomatization of validity of this logic. It is also a sound and complete axiomatization of Λ-validity when Λ is the family of all general transition systems. The language of CTL* is equivalent to the basis branching time logic built from this linear language. Unlike the usual account of CTL* [EH] we do not distinguish here between state and path formulas (and so each boolean operation has jut one semantic clause). A sound and complete axiomatization of Λ-validity consists of the linear axioms and rules together with the following extra axioms and rule:

Axioms	1.	$A \rightarrow \triangle Q$ Q atomic
	2.	$\triangledown Q \rightarrow Q$ Q atomic
	3.	$\triangle A \rightarrow A$
	4.	$\triangle(A \rightarrow B) \rightarrow (\triangle A \rightarrow \triangle B)$
	5.	$\triangle A \rightarrow \triangle \triangle A$
	6.	$\triangledown A \rightarrow \triangle \triangledown A$
Rule	Gen if $\vdash A$ then $\vdash \triangle A$	

The modality $\triangle$ is very like the S5 [ ] modality (except on atomic formulas and their negations). The proof of completeness is sketched in section 3. A conjecture: the addition of the above rule and axioms to any sound and complete linear time logic results in a sound and complete axiomiatization of its branching time version (for a sufficiently general class of Λ-validity).

We now compare the global expressiveness of linear and branching time logics. To facilitate this we only deal with future time logics (The results can be generalized at the expense of notation.)

2. Expressibility: Zig-Zags, Bisimulations and Traces

Various expressibility issues arise when examining modal and temporal logics. At the micro-level there is interest in the particular way that individual (or effective sets of) formulas express properties and delimit models. Pertinent here is the extent to which the metalanguage, the language the semantics is couched in, as a first-order language is itself reflected in the modal or temporal object languages. Instead our concern is with the macro-level, the totality of modal or temporal formulas. We examine when two states of a model satisfy the same formulas and apply the results to process equivalences.

Transition systems are widely used as models of processes. Where $T = (S, L, R)$, S is a set of processes; L a set of (observable) actions; and sR_as' means: process s may perform action a and become s'. A variety of behavioural equivalences between processes have been proposed [BBK,BHR,De,DH,Mi,Ol,Pa]. We are interested in the coincidence of a behavioural equivalence and an equivalence induced by a logic - when two states satisfy the same formulas. The classic result in this area is the Hennessy-Milner modal logic characterization of bisimulation equivalence [HM].

The framework developed in section 1 allows us to offer similar general results for temporal logics. They support the argument in [Pn] that the distinction between linear and branching time is important to the difference between testing, or failures, equivalence and bisimulation equivalence. Further support can be found in the different testing framework[Ab]: the role of the test operators $\forall, \exists$ is analogous to the branching time operators Δ, ∇. The results for branching time logics generalize [HS].

Equivalence of States in Modal Logic

Assume a modal model $\mathcal{M} = (S, L, R, V)$. Our interest is in the question: when do two states in S satisfy the same formulas? Equally, we could have asked the question of two states from different models. The answer appeals to bisimulations with a little makeup, or as they are called in modal logic zig-zag relations [Va].

Definition 2.1 A zig-zag on $\mathcal{M}$ is a relation $Z \subseteq S \times S$ such that
if sZs' then

i. $\forall$ atomic Q. $s \in V(Q)$ iff $s' \in V(Q)$

ii. $\forall s_1, a$. if sR_as_1 then $\exists s_1'.(s'R_as_1'$ and $s_1Zs_1')$

iii. $\forall s_1', a$. if $s'R_as_1'$ then $\exists s_1(sR_as_1$ and $s_1Zs_1')$

In [Va] zig-zags are defined between states of different models and for modal languages without the indexed modalities [a]: then the indices on R and the

quantifiers on labels are dropped from ii and iii. The definition of a bisimulation relation omits clause i [Mi]. (Alternatively, it is a zig-zag for modal languages whose sole atomic sentence is tt, true.) When there is a zig-zag relation between s and s' we write $s \leftrightarrows s'$. The following theorem from [Va] is a slight variant of one half of the Hennessy-Milner characterization of bisimulation equivalence. We use the notation $\|s\|^{\mathcal{M}}$ to denote $\{A \mid s \models_{\mathcal{M}} A\}$.

Theorem 2.2 If $s \leftrightarrows s'$ then $\|s\|^{\mathcal{M}} = \|s'\|^{\mathcal{M}}$

Proof As in [HM] by structural induction on modal formulas. □

The converse holds if infinitary conjunction is allowed in the modal language. Alternatively, for finitary languages, it holds if the model $\mathcal{M}$ is image finite in the sense that its transition system is. In effect, the other half of the Hennessy-Milner characterization is:

Theorem 2.3 If $\mathcal{M}$ is image finite and $\|s\|^{\mathcal{M}} = \|s'\|^{\mathcal{M}}$ then $s \leftrightarrows s'$.

Proof By showing that the relation $Z \subseteq S \times S$ such that sZs' iff $\|s\|^{\mathcal{M}} = \|s'\|^{\mathcal{M}}$ is a zig-zag. □

Equivalence of States in Linear and Branching Time Models

Now let $\mathcal{M} = (S, L, R, \Sigma, V)$ be a temporal model. As we concentrate on future time temporal languages we assume Σ is suffix closed: if $\sigma\sigma' \in \Sigma$ and σ has finite length then $\sigma' \in \Sigma$ with $L(\sigma')$ its label sequence. Moreover, let $\Sigma(s)$ be the set of paths in Σ whose initial state is s. This means that all paths through a state s are canonically represented in $\Sigma(s)$. First we introduce a trace relation $\approx \subseteq \Sigma \times \Sigma$.

Definition 2.4 For $\sigma, \sigma' \in \Sigma$ $\sigma \approx \sigma'$ iff
- i. $L(\sigma) = L(\sigma')$
- ii $\forall i \geq 0 \; \forall$ atomic Q. $\sigma(i) \in V(Q)$ iff $\sigma'(i) \in V(Q)$

Condition i is dropped when the linear language doesnt involve indexed future operators like X_α. When the only atomic sentence is tt then condition ii is redundant, in which case $\sigma \approx \sigma'$ if they involve the same sequence of actions. Now a general lemma for any linear language α. Here $\|\sigma, 0\|^{\mathcal{M}}_{\alpha}$ denotes $\{A \in \alpha \mid \sigma, 0 \models_{\mathcal{M}} A\}$

Lemma 2.5 If $\sigma \approx \sigma'$ then $\|\sigma, 0\|^{\mathcal{M}}_{\alpha} = \|\sigma', 0\|^{\mathcal{M}}_{\alpha}$

Proof Suppose $\sigma \approx \sigma'$. Then the proof proceeds by showing $\sigma, 0 \models_{\mathcal{M}} A$ iff $\sigma', 0 \models_{\mathcal{M}} A$ by induction on the structure of A.
The only interesting case is $A = \#(B_1, \cdots, B_n)$ where $\#$ is linear.
Where I is a set of indices then we let the value of
$\sigma, I \models_{\mathcal{M}} B_i$ be determined from the values of $\sigma, j \models_{\mathcal{M}} B_i \;\; j \in I$(and $j \geq 0$).
Also let $L(\sigma, I)$ be the values $L(\sigma, j)$.
Then the value of $\sigma, 0 \models_{\mathcal{M}} A$ is determined in a fixed way from
the values $\sigma, I_i \models B_i$ and $L(\sigma, \cup I_i)$, $1 \leq i \leq n$
Clearly, where $\sigma[k]$ represents the kth suffix of σ then
$\sigma, k \models_{\mathcal{M}} C$ iff $\sigma[k], 0 \models_{\mathcal{M}} C$ for future time linear logics.
Moreover $\sigma \approx \sigma'$ iff $\forall k \geq 0 \;\; \sigma[k] \approx \sigma'[k]$.
Hence by the induction hypothesis and these observations
for each $k \in I_i \;\; \sigma, k \models_{\mathcal{M}} B_i$ iff $\sigma', k \models_{\mathcal{M}} B_i$ □

The converse of this lemma holds for particular languages α where X_a, for each label a, is expressible (or where X is when condition i of definition 2.4 doesnt apply). We use the notation $X_a \in \alpha$ to denote these expressibility conditions.

Lemma 2.6 If $X_a \in \alpha$ and $\|\sigma, 0\|_\alpha^{\mathcal{M}} = \|\sigma', 0\|_\alpha^{\mathcal{M}}$ then $\sigma \approx \sigma'$

Proof Suppose $X_a(X)$ is expressible in α for each a, and $\sigma \not\approx \sigma'$
Then either $L(\sigma) \neq L(\sigma')$ or for some $i \geq 0$ $\sigma(i)$ and $\sigma'(i)$
differ on their satisfiability of atomic formulas.
But then clearly there is an α formula $X_{a1} \cdots X_{an} tt$ or
$X^n Q$, Q atomic, distinguishing $\sigma, 0$ from $\sigma', 0$ □

We now generalize to states by defining a path relation $\downarrow \subseteq \mathcal{S} \times \mathcal{S}$, the linear time version of zig-zags.

Definition 2.7 For $s, s' \in \mathcal{S} \;\; s \downarrow s'$ iff
i $\forall \sigma \in \Sigma(s) \;\; \exists \sigma' \in \Sigma(s') \; \sigma \approx \sigma'$
ii $\forall \sigma' \in \Sigma(s') \;\; \exists \sigma \in \Sigma(s) \; \sigma \approx \sigma'$

The following is almost a corollary of lemma 2.5

Theorem 2.8 If $s \downarrow s'$ then $\|s\|_\alpha^{\mathcal{M}} = \|s'\|_\alpha^{\mathcal{M}}$

Proof Straightforward □

The converse holds if $X_a \in \alpha$ and the sets $\Sigma(s)$ and $\Sigma(s')$ are finite in size.

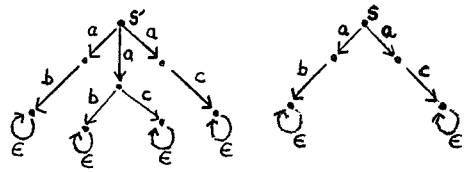
Theorem 2.9 If $X_a \in \alpha$ and $\Sigma(s), \Sigma(s')$ are finite and $\|s\|_\alpha^{\mathcal{M}} = \|s'\|_\alpha^{\mathcal{M}}$ then $s \downarrow s'$

Proof Straightforward from lemma 2.6, the boolean closure of α, and finiteness of $\Sigma(s), \Sigma(s')$ □

In the particular case that $X_a \in \alpha$ and tt is its only atomic sentence then α characterizes traces or strings equivalence of processes when $\Sigma(s)$ and $\Sigma(s')$ are always of finite size. Yet Hennessy-Milner logic, which characterizes bisimulation equivalence, is a sublanguage of the basis branching time temporal logic of α: the modality [a] is just $\triangle\neg X_a\neg$. So there is an enormous chasm in process distinguishability between α and its branching time version. One way of closing this gap is to include more atomic formulas in the temporal languages. One suggestion is to let atomic formulas Q denote finite subsets of labels, and then the valuation V could be

$$s \in V(Q) \text{ iff } \{a \mid \exists s'.sR_as'\} \cap Q \neq \emptyset$$

The resulting equivalence on processes is ready trace equivalence [BBK]. It is finer than both refusals and testing, or failures, equivalence [BRH,DH,Ph] forgetting divergence and how silent moves should be dealt with. For instance the following pair of states s and s' are distinguishable.



where $\xrightarrow{a}$ is R_a and $\in$ is some 'fictional' label to ensure totality of R. Where $Q_1 = \{b\}$ and $Q_2 = \{c\}$ and A is $X_a\neg Q_1 \vee X_a\neg Q_2$ then $s \models_M A$ while $s' \not\models_M A$. In the terminology of [Ab] this is due to 'copying' through the use of $\vee$. A small sublanguage of α, without copying, does indeed characterize testing, or failures, equivalence. Where atomic sentences are finite subsets of α or tt then the sublanguage is:

$$A ::= Q \mid \neg A \mid X_a$$

For every formula of this language is equivalent to an α formula, $n \geq 0$,

$$(X_{a1} \cdots X_{an} P) \vee D$$

where P is Q_i or $\neg Q_i$ and D is tt or a disjunction of formulas drawn from $\{\neg X_{a1}tt,\ X_{a1}\neg X_{a2}tt, \cdots, X_{a1} \cdots X_{an-1}\neg X_{an}tt\}$. But now this language has so little expressive power that its branching time version also characterizes the equivalence.

For branching time logics we need a notion of zig-zag on paths.

Definition 2.10 A path zig-zag on $\mathcal{M}$ is a relation $Z \subseteq \Sigma$ such that if $\sigma Z \sigma'$ then

i. $\sigma \approx \sigma'$

ii. $\forall i \geq 0 \ \forall \sigma_1 \in \Sigma(\sigma(i)) \ \exists \sigma_1' \in \Sigma(\sigma'(i)) \ \sigma_1 Z \sigma_1'$

iii. $\forall i \geq 0 \ \forall \sigma_1' \in \Sigma(\sigma'(i)) \ \exists \sigma_1 \in \Sigma(\sigma(i)) \ \sigma_1 Z \sigma_1'$

If there is a zig-zag between σ and σ' then we write this as $\sigma \leftrightarrows \sigma'$.

Again, if the temporal language doesnt contain indexed modalities then condition i doesnt include the requirement that $L(\sigma) = L(\sigma')$. The correlate of lemma 2.5 for any branching time language $\mathcal{B}$ is:

Lemma 2.11 If $\sigma \leftrightarrows \sigma'$ then $\|\sigma, 0\|_{\mathcal{B}}^{\mathcal{M}} = \|\sigma', 0\|_{\mathcal{B}}^{\mathcal{M}}$

Proof As for lemma 2.5
The only new case is $A = \triangle B$
Suppose $\sigma \leftrightarrows \sigma'$
Then $\sigma, 0 \models_{\mathcal{M}} \triangle B$ iff $\forall \sigma_1 \in \Sigma(\sigma(0)) \ \sigma_1, 0 \models_{\mathcal{M}} B$
But if $\sigma', 0 \not\models_{\mathcal{M}} \triangle B$ then $\sigma_1', 0 \not\models_{\mathcal{M}} B$ for some $\sigma_1' \in \Sigma(\sigma'(0))$.
But then by the induction hypothesis there is a $\sigma_1 \in \Sigma(\sigma(0))$ such that $\sigma_1, 0 \not\models_{\mathcal{M}} B$ □

The converse holds if $X_a \in \mathcal{B}$ and for each $i \geq 0 \ \Sigma(\sigma(i))$ and $\Sigma(\sigma'(i))$ are finite in size. Path zig-zags are now generalized to states: the result is, in effect, extended bisimulations [He].

Definition 2.12 An extended zig-zag on $\mathcal{M}$ is a relation $Z \subseteq \mathcal{S} \times \mathcal{S}$ such that if $s \, Z \, s'$ then

i. $\forall \sigma \in \Sigma(s) \ \exists \sigma' \in \Sigma(s') \ \sigma \leftrightarrows \sigma'$

ii. $\forall \sigma' \in \Sigma(s) \ \exists \sigma \in \Sigma(s) \ \sigma \leftrightarrows \sigma'$

We write $s \leftrightarrows^+ s'$ if s and s' are connected by an extended zig-zag.
The branching time correlate of theorem 2.8 is:

Theorem 2.13 If $s \leftrightarrows^+ s'$ then $\|s\|_{\mathcal{B}}^{\mathcal{M}} = \|s'\|_{\mathcal{B}}^{\mathcal{M}}$

Proof Straightforward from lemma 2.11. □

The converse:

Theorem 2.14 If $X_a \in \mathcal{B}$ and $\forall s_1 \Sigma(s_1)$ is finite and $\|s\|_{\mathcal{B}}^{\mathcal{M}} = \|s'\|_{\mathcal{B}}^{\mathcal{M}}$ then $s \leftrightarrows^+ s'$.

Proof Straightforward □

In the special case that $\mathcal{M}$ is saturated in the sense that Σ contains all paths through the transition system then the pair of relations on states. $\leftrightarrows$, $\leftrightarrows^+$ coincide. The important feature guaranteeing this is limit closedness of paths in the sense that if for all $i, s_0 R_{a0} \cdots R_{ai-1} s_i$ is a prefix of a path in Σ then the infinite path $s_0 R_{a0} \cdots s_i R_{ai} s_{i+1} \cdots$ is also in Σ. The proof of the following lemma is a slight variant of a result in [HS]

Lemma 2.15 If $\mathcal{M}$ is saturated then $s \leftrightarrows s'$ iff $s \leftrightarrows^+ s'$

Proof See lemma 4.3 of [HS] □

This shows that for saturated $\mathcal{M}$ modal logic has the same discriminating power as arbitrary branching time logics, a discriminating power which goes well beyond linear time logics. But, of course, these temporal logics are much richer when expressing particular properties in terms of individual formulas.

3. A Complete Axiomatization of CTL* Formulas

We show that adding the following axioms and rule (from section 1) to the axiomatization of linear time logic in [GPSS] results in a sound and complete axiomatization of Λ-validity of CTL* formulas where Λ is the family of all extended transition systems.

Axioms
1. $Q \rightarrow \triangle Q$ Q atomic
2. $\nabla Q \rightarrow Q$ Q atomic
3. $\triangle A \rightarrow A$
4. $\triangle(A \rightarrow B) \rightarrow (\triangle A \rightarrow \triangle B)$
5. $\triangle A \rightarrow \triangle \triangle A$
6. $\nabla A \rightarrow \triangle \nabla A$

Rule Gen if $\vdash A$ then $\vdash \triangle A$

Soundness is clear. For completeness we extend the proof of [GPSS]. Recall that the linear time logic is a future logic without indexed modalities based on the modality U, for until: the temporal modalities X, F, G are definable in terms of it. We let $ST = \{\triangle A \mid A \in CTL^*\} \cup \{\neg \triangle A \mid A \in CTL^*\}$ be the set of 'state' formulas of CTL*.

A formula A is consistent iff $\not\vdash \neg A$. We assume the standard result: every consistent set of formulas can be extended to a maximal consistent set. Let $\mathcal{MC}$ be the set of all maximal consistent sets with Γ as a typical member. Three relations are defined on $\mathcal{MC}$.

Definition 3.1 For $\Gamma, \Gamma' \in \mathcal{MC}$

i $\Gamma X \Gamma'$ iff $\{A \mid XA \in \Gamma\} \subseteq \Gamma'$
ii $\Gamma < \Gamma'$ iff $\{A \mid GA \in \Gamma\} \subseteq \Gamma'$
iii $\Gamma \equiv \Gamma'$ iff $ST \cap \Gamma = ST \cap \Gamma'$

The idea is that maximal sequences of member of $\mathcal{MC}$ $\Gamma_0 .. \Gamma_i ....$ with $\Gamma_j X \Gamma_{j+1}$, for all $j \geq 0$, represent paths. So X is a 'next' relation while '$<$' is a before relation: $\Gamma_i < \Gamma_j$ for all $j > i$ in the sequence. Finally the state determined by the i^{th} point is the sequence will be $\Gamma_i \cap ST$: so $\equiv$ means 'the same state as'. Some properties of these relations:

Lemma 3.2
i $\forall \Gamma \exists! \Gamma' \; \Gamma X \Gamma'$
ii $<$ is transitive and connected
iii $X \subseteq <$
iv If $FA \in \Gamma$ then $\exists \Gamma' > \Gamma$ and $A \in \Gamma'$
v If $\triangle A \notin \Gamma$ then $\exists \Gamma' \equiv \Gamma$ and $A \notin \Gamma'$

Proof i - iv are standard

v Consider $\Gamma \cap ST = \alpha$
If $\alpha \cup \{\neg A\}$ is consistent then there is a $\Gamma' \supseteq \alpha\ \Gamma' \in MC$
Clearly $\Gamma' \equiv \Gamma$
So suppose $\alpha \vdash A$
then for some $B_1, \cdots, B_n$ $\quad B_1, \cdots B_n \vdash A$
By Gen and axiom 4 $\quad \triangle B_1, \cdots, \triangle B_n \vdash \triangle A$
But each $B_i \in ST$, so $B_i \rightarrow \triangle B_i$ by axioms 5 and 6
Hence $\alpha \vdash \triangle A$ □

We almost have a canonical model. But there is a problem: $<$ is not the transitive closure of X. which is demanded by the definition of a path. For by compactness we know that the set $\{X^n A \mid n > 0\} \cup \{F\neg A\}$ is consistent (where X^n is $\underbrace{X \cdots X}_{n\ times}$). So we resort to filtrations [GPSS].

Definition 3.3 For any formula A. $cl(A)$ is the least set such that:

i $cl(A)$ is closed under subformulas and boolean operations
ii if $Q \in cl(A)$ then $\triangle Q \in cl(A)$ for atomic Q
iii if $XA \in cl(A)$ then $X\neg A \in cl(A)$
iv if $FA \in cl(A)$ then $XA, XFA \in cl(A)$
v if $A \cup B \in cl(A)$ then $FB, XA, X(A \cup B) \in cl(A)$

Any $cl(A)$ is representable up to logical equivalence by a finite subset containing A [GPSS]. The size of the subset is also determinable from the size of A. We now let $cl(A)$ stand for a particular such subset. Next we filter MC sets through $cl(A)$. Let $\overline{MC} = \{\Gamma \cap cl(A) \mid \Gamma \in MC\}$. Now $\overline{MC}$ is finite. We let α, β denote members of $\overline{MC}$. Using finest filtrations, relations $\overline{X}, \overline{<}, \overline{\equiv}$ 'corresponding' to $X, <, \equiv$ are defined:

Definition 3.4 For each $* \in \{X, <, \equiv\}$ and $\alpha, \beta \in \overline{MC}$
$\alpha \overline{*} \beta$ iff $\exists \Gamma, \Gamma' \in MC. \alpha = \Gamma \cap cl(A)$ and $\beta = \Gamma' \cap cl(A)$ and $\Gamma * \Delta$.

So we have

Lemma 3.5 i $\overline{<}$ is transitive
ii $\overline{X}^+ = \overline{<}$

Proof As in [GPSS] □

We now define states and paths.

Definition 3.6 i Let $[\alpha] = \{\beta \mid \alpha \equiv \beta\}$

ii Let $\mathrm{Seq}(\alpha_0)$ be the set of all maximal sequences $\sigma = \alpha_0\alpha_1....$ such that

a) $\alpha_i \overline{X} \alpha_{i+1}$ for all $i \geq 0$

b) for any β if β occurs infinitely often in σ and $\beta \overline{X} \beta'$ then β' also occurs infinitely often in σ.

Theorem 3.7 Every consistent formula A is Λ-satisfiable.

Proof A can be extended to $\Gamma \in MC$
Consider a representative of $cl(A)$ and consider $\overline{MC}$.
Clearly, for some $\alpha \in \overline{MC}$ $A \in \alpha$
Now the model $M = (S, R, \Sigma, V)$ is
$S = \{[\beta] \mid \beta \in \overline{MC}\}$
and $[\beta]R[\gamma]$ iff $\exists \beta' \in [\beta] \exists \gamma' \in [\gamma] \beta' \overline{X} \gamma'$
and $\Sigma = \cup\, Seq(\alpha)$
$\alpha \in \overline{MC}$
and $V(Q) = \{[\alpha] \mid \exists \beta \in [\alpha].Q \in \beta\}$.
Clearly each $\sigma \in \Sigma$ is a path
If $\sigma \in \Sigma$ then let (σ, i) be the i^{th} member of σ in σ
let $\sigma(i) = [(\sigma, i)]$
The proof is completed by showing using induction on B that
for all $B \in cl(A)$. $\sigma, i \models_M B$ iff $B \in (\sigma, i)$.
The proof is as in [GPSS] except for atomic B and $B = \triangle C$.

$B = Q$ Suppose $\sigma, i \models_M Q$
Then $\exists \alpha \in [(\sigma, i)]$ and $Q \in \alpha$
But then by axiom 1 $\triangle Q \in \alpha$
So by closure conditions on $cl(A)$ $\triangle Q \in (\sigma, i)$.
So by axiom 3 $Q \in (\sigma, i)$.
Suppose $\sigma, i \not\models_M Q$
Then $\forall \alpha \in [(\sigma, i)] Q \notin \alpha$
Hence by $cl(A)$ conditions $\neg Q \in \alpha$, for all α.
In particular $\neg Q \in (\sigma, i)$

$B = \triangle C$ Suppose $\sigma, i \models_M \triangle C$
Then for all $\alpha \in [(\sigma, i)] C \in \alpha$ by induction hypothesis
But if $\triangle C \notin (\sigma, i)$ then by lemma 3.2 $\exists \beta . \beta \in [(\sigma, i)]$ and $C \notin B$.
Suppose $\sigma, i \not\models_M \triangle C$.
Then via lemma 3.2 and induction hypothesis $\exists \beta . \beta \in [(\sigma, i)]$ and $C \notin \beta$
Hence $\triangle C \notin (\sigma, i)$. □

Acknowledgement
Thanks to Dorothy McKie and Margaret Melvin for typing.

References

[Ab] Abramsky, S. Observation equivalence as a testing equivalence, draft paper, University of London (1985)

[BBK] Baeten, J., Bergstra, J., Klop, J. 'Ready trace semantics for concrete process algebra with priority operator', Report CS-R8517, Centrum voor Wiskunde en Informatica (1985).

[BHR] Brookes, S., Hoare, C. and Roscoe, A. 'A theory of communicating sequential processes' JACM pp. 560-591 (1984).

[BMP] Ben-Ari, M., Manna, Z., and Pnueli, A. 'The temporal logic of branching time' in POPL pp. 164-176 (1981).

[BR] Brookes, S., Rounds, W. 'Behaviour equivalence relations induced by programming logics', LNCS Vol.154 pp.97-108 (1983).

[CE] Clarke, E.M., Emerson, E.A. 'Using branching time temporal logic to synthesize synchronization skeletons' Science of Computer Programming pp. 241-266 (1982).

[De] DeNicola, R. 'Testing equivalences and fully abstract models for communicating processes'. University of Edinburgh Ph.D thesis (1985).

[DH] De Nicola, R., Hennessy, M. 'Testing equivalences for processes' Theoretical Computer Science pp. 83-133 (1984).

[EH] Emerson, E.A., Halpern, J., ''Sometimes'' and 'not never' revisited: on branching versus linear time temporal logic. JACM pp. 151-178 (1986)

[EL] Emerson, E.A., Lei, C-L., 'Modalities for model checking ; branching time strikes back', In POPL, pp. 84-96 (1985).

[GPSS] Gabbay, D., Pnueli, A., Shelah, A., Stavi, J. 'The temporal analysis of fairness' Proc. 7th POPL pp. 163-173 (1980).

[He] Hennessy, M. 'Axiomatizing finite delay operators' Acta Inform' pp. 61-88 (1984).

[HKP] Harel, D., Kozen, D. and Parikh, R. 'Process logic : expressiveness, decidability, completeness'. J. Comput. System Sci. pp.144-170 (1982).

[HM] Hennessy, M., Milner, R. 'Algebraic laws for nondeterminism and concurrency' JACM pp. 137-161 (1985).

[HS] Hennessy, M., Stirling, C. 'The power of the future perfect in program logics' Information and Control pp. 23-52 (1985).

[La] Larsen, K. 'Proof systems for Hennessy-Milner logic with recursion', draft paper, Aalborg University Centre (1986).

[Mi] Milner, R. 'Calculi for synchrony and asynchrony' Theoretical

Computer Science, pp. 267-310 (1983).

[MP] Manna Z., and Pnueli, A. 'How to cook a temporal proof system for your pet language', in POPL pp. 141-154 (1982).

[Ol] Olderog, E., 'Semantics of concurrent processes: the search for structure and abstraction', EATCS Bulletin No.28 pp. 73-97 (1986).

[Pa] Park, D., 'Concurrency and automata on infinite sequences' LNCS Vol.104 (1981).

[Ph] Phillips, I. 'Refusal testing' LNCS Vol. 226 pp.304-313 (1986).

[Pn] Pnueli, A. 'Linear and branching structures in the semantics and logics of reactive systems' LNCS Vol. 194 pp. 14-32 (1985).

[Si] Sifakis, J. 'A unified approach for studying the properties of transition systems'. Theoretical Computer Science pp. 227-258 (1982).

[Va] Van Bentham, J. 'Correspondence theory' in Vol II of Handbook of Philosophical Logic, ed. Gabbay, D., and Guenthner, F. Reidel, D. pp. 167-247 (1984).