On the Expressive Power of CTL*

Faron Moller
Uppsala University
fm@csd.uu.se

Alexander Rabinovich*
Tel Aviv University
rabino@math.tau.ac.il

Abstract

We show that the expressive power of the branching time logic CTL coincides with that of the class of bisimulation invariant properties expressible in so-called monadic path logic: monadic second order logic in which set quantification is restricted to paths. In order to prove this result, we first prove a new Composition Theorem for trees. This approach is adapted from the approach of Hafer and Thomas in their proof that CTL* coincides with the whole of monadic path logic over the class of full binary trees.*

1 Introduction

Various temporal logics have been proposed for reasoning about so-called "reactive" systems, computer hardware or software systems which exhibit (potentially) nonterminating and nondeterministic behaviour. Such a system is typically represented by the (potentially) infinite sequences of computation states through which it may evolve, where we associate with each state the set of atomic propositions which are true in that state, along with the possible next state transitions to which it may evolve. Thus its behaviour is denoted by a (potentially) infinite rooted tree, with the initial state of the system represented by the root of the tree.

Various equivalences have been proposed between such systems as well, depicting when two systems should be deemed the same. Given such an equivalence, it is most desirable that the temporal logic being employed does not distinguish between two equivalent behaviours: a temporal property which holds of a particular system should hold for all equivalent systems.

In this paper, we shall be interested specifically in bisimulation equivalence [19, 18], the branching time temporal logic CTL* [2, 4], and so-called monadic path logic MPL [11, 12]: monadic second order logic in which set quantification is restricted to paths. It is well known that CTL* respects bisimulation equivalence in the above sense, while already first order logic does not. However, Hafer and Thomas [12] demonstrate that every CTL* property can be expressed in MPL, and that the reverse is also true if you restrict attention to full binary trees. (Two full binary trees are bisimulation equivalent only if they are isomorphic, and MPL respects isomorphism.) In this paper we modify their argument to show that the reverse is true over all trees when we restrict attention to the bisimulation invariant subset of MPL properties, those MPL properties which do not distinguish between bisimulation equivalent trees. We thus show that CTL* corresponds precisely to bisimulation invariant MPL.

The first result of this kind is due to van Bentham [1], where the notion of a bisimulation first appears (though the concept is referred to there as a "zig-zig" relation); van Bentham's result is that propositional modal logic coincides with the bisimulation invariant properties expressible in first order logic. A further closely related result due to Janin and Walukiewicz [14] shows that the propositional μ-calculus coincides with the bisimulation invariant properties expressible in the whole of monadic second order logic. However, whereas the proof of Janin and Walukiewicz relies on an automata-theoretic characterisation of such logical properties, the present proof is automata-free, based instead on a Composition Theorem which is proven with the aid of the appropriate Ehrenfeucht-Fraissé game.

In the remainder of this introduction, we provide the relevant definitions for trees, bisimulation equivalence, monadic second order logics, and CTL*. In the next section we present standard results about characterising equivalences using Ehrenfeucht-Fraissé games, and in the following section we present and prove the main technical result of the paper, our Composition Theorem for trees. In the final section, we apply our Composition Theorem to the problem of showing our expressive completeness result for CTL* with respect to the bisimulation invariant fragment of MPL.

*This work was carried out while the second author was visiting Uppsala University supported by a grant from the Swedish STINT Fellowship Programme.

1.1 Computation Trees

A ***tree*** t consists of a partially-ordered set of ***nodes*** S_t in which the ancestors of any given node $s \in S_t$ constitute a finite total order with a common minimal element ε_t, referred to as the ***root*** of the tree. Computationally, a node in the tree corresponds to a state in a computation, and its ancestors $\{s' \in S_t : s'<s\}$ correspond to the states passed through in the computation leading up to the state (corresponding to node) s, starting from the initial state (corresponding to node) ε_t. We shall denote the (immediate) successor relation by $\rightarrow$, so that $s \rightarrow s'$ if and only if $s<s'$ and there is no s'' with $s<s''<s'$. Computationally, $s \rightarrow s'$ means that there is a transition from state s to state s', and the set of ancestors of a state can thus be listed with s as such a sequence of transitions: $\varepsilon_t \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n \rightarrow s$.

Furthermore, the nodes of a tree are labelled by elements taken from some finite set Σ, representing the atomic properties which are true at the given state of the computation. Hence a tree is in fact given as a labelling function $t : S_t \rightarrow \Sigma$, and we shall refer to such a tree as a ***Σ-valued tree***. (In the literature, states are sometimes labelled by subsets of a set Δ of atomic properties, with each state s assigned the label corresponding to the set of atomic properties which hold at s; indeed, this is how we described the situation at the beginning of the Introduction. However, we can restrict the labelling to elements of Σ simply by taking Σ to be the collection of all subsets of Δ.)

A ***path*** through a tree t is an ω-sequence—or a maximal finite sequence—of successive nodes $\pi = \langle s_1, s_2, s_3, \ldots\rangle$ through the tree; that is, $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \cdots$; and if this sequence is finite, then there is no successor for the final node. (In fact, we shall occasionally use the term "path" to refer to a non-maximal finite sequence of successive nodes ending at a specified node, but we shall always be explicit with such uses.) If the first node of a path is the root ε_t, then the path is referred to as a ***full path*** or ***branch***. The ith node s_i in the path π is denoted by π_i, and we shall use $\pi^i = \langle \pi_i, \pi_{i+1}, \pi_{i+2}, \ldots\rangle$ to denote the subpath of π rooted at π_i. (In particular, $\pi = \pi^1$.) Finally, we shall use t_s to denote the subtree of t rooted at the node s.

Remark 1.1 (More general trees) As described, the trees which we are using are ω-trees; that is, each branch is either a finite sequence or an ω-sequence of nodes. However, all of our results—and in particular our Composition Theorem— hold for a more general class of trees, namely all partially-ordered sets in which the ancestors of any given node constitute a well-founded total order with a common root ε. The well-foundedness ensures that the successors of any node are defined; that is, if $s<s'$ then $s \rightarrow s''$ for some $s''\leq s'$. However, for ease of presentation, we restrict ourselves to the above computationally-motivated subclass of ω-trees.

1.2 Bisimulation Equivalence

One popular equivalence between computation trees is that of bisimulation equivalence. This equivalence catches subtle differences between trees based on their branching structures. It has a very appealing mathematical theory, and is generally regarded as the finest behavioural equivalence of interest for concurrency (it is often argued that concurrent systems giving rise to bisimulation equivalent computation trees are indistinguishable for all reasonable notions of observation).

Informally two trees are bisimulation equivalent if they differ only up to multiplicity and ordering of their subtrees. Formally we have the following co-inductive definition. A ***bisimulation*** is a binary relation $\mathcal{R} \subseteq S_{t_1} \times S_{t_2}$ between the nodes of two trees t_1 and t_2 which relates their roots: $\langle \varepsilon_{t_1}, \varepsilon_{t_2}\rangle \in \mathcal{R}$; and such that whenever $\langle s_1, s_2\rangle \in \mathcal{R}$ we have that:

- s_1 and s_2 have the same labels: $t_1(s_1) = t_2(s_2)$;
- if $s_1 \rightarrow s_1'$ then $s_2 \rightarrow s_2'$ with $\langle s_1', s_2'\rangle \in \mathcal{R}$; and
- if $s_2 \rightarrow s_2'$ then $s_1 \rightarrow s_1'$ with $\langle s_1', s_2'\rangle \in \mathcal{R}$.

Two trees t_1 and t_2 are ***bisimulation equivalent***, written $t_1 \sim t_2$, if they are related by some bisimulation.

We shall be interested in distinguishing between trees only up to bisimulation equivalence. In particular, we shall associate with any given tree t a "wide" normal form tree t^w which is obtained by reproducing every subtree an infinite number of times. Formally, we say that a ***wide*** tree is one in which for every transition $s \rightarrow s'$ there are infinitely many transitions $s \rightarrow s''$ such that $t_{s'}$ is isomorphic to $t_{s''}$. Clearly every tree t is bisimulation equivalent to the wide tree t^w whose construction is as describe above. Formally, the tree t^w can be defined as follows.

1. The nodes are $S_{t^w} = \Big\{(s,u) : s \in S_t$ and u is a finite sequence of integers of length equal to the length of the (partial) path from the root ε to $s\Big\}$; its root is $\varepsilon_{t^w} = (\varepsilon, \lambda)$ (where λ denotes the empty word).
2. $(s,u) < (s',u')$ iff $s < s'$ and $u' = uv$ for some v.
3. The labelling function is $t^w(s,u) = t(s)$.

Remark 1.2 (Bisimulation equivalence is trivial over full binary trees) A ***full binary tree*** is a tree in which every node has exactly two successors. Note that any two full binary trees which are bisimulation equivalent must in fact be isomorphic.

1.3 Monadic Second Order Logic and Monadic Path Logic

The monadic second order logic MSOL$(<, \Sigma)$ appropriate for expressing properties of Σ-valued trees has individual variables $x, y, z, \ldots$ (representing nodes), set variables X, Y, Z (representing sets of nodes), and predicate constants P_a ($a \in \Sigma$). Formulas are built up from atomic formulas of the form $x = y$, $x < y$, $x \in X$ and $x \in P_a$ using the propositional connectives $\wedge$ and $\neg$, and the quantifier $\exists$. We shall denote by FOL$(<, \Sigma)$ the subset of first order formulas, those that do not involve set variables. We shall write $\varphi(x_1, x_2, \ldots, x_m, X_1, X_2, \ldots, X_n)$ to indicate the variables which (may) appear ***free*** in φ, that is, not within the scope of a quantifier. The ***quantifier depth*** of a formula φ, denoted by qd(φ), is inductively defined to be the maximum number of nested quantifiers in φ: $\mathrm{qd}(\varphi) = 0$ for atomic formulas φ; $\mathrm{qd}(\varphi \wedge \varphi') = \max(\mathrm{qd}(\varphi), \mathrm{qd}(\varphi'))$; and $\mathrm{qd}(\exists x \varphi) = \mathrm{qd}(\exists X \varphi) = 1 + \mathrm{qd}(\varphi)$.

As usual, a formula is ***closed*** if it involves no free variables, in which case it is referred to as a ***sentence***. Note that every formula must involve some first order variable, so there are no sentences with quantifier depth 0; and sentences of quantifier depth 1 have only first order variables occurring within them. We write

$$(t, s_1, s_2, \ldots, s_m, S_1, S_2, \ldots, S_n) \models \varphi(x_1, x_2, \ldots, x_m, X_1, X_2, \ldots, X_n)$$

if the formula $\varphi(x_1, x_2, \ldots, x_m, X_1, X_2, \ldots, X_n)$ is satisfied in the Σ-valued tree t with x_i interpretted by the node s_i ($1 \leq i \leq m$) and X_j interpretted by the set of nodes S_j ($1 \leq j \leq n$).

We shall also interpret FOL$(<, \Sigma)$ sentences over words $w \in \Sigma^*$ and ω-sequences $w \in \Sigma^\omega$, writing $w \models \varphi$ to mean the expected: variables represent positions in the sequence with $=$ and $<$ representing relative position, and $x \in P_a$ means that the letter at position x is a. It is straightforward to verify the correctness of this interpretation, in the sense that for any tree t which just consists of a path $\pi = \langle \pi_1, \pi_2, \ldots \rangle$ (that is, every node has at most one successor), and for any FOL$(<, \Sigma)$ sentence φ, we have that $t \models \varphi$ iff $t(\pi) \models \varphi$, where $t(\pi) = t(\pi_1)t(\pi_2)\cdots$.

Monadic path logic MPL is defined to be the monadic second order logic as described, but where we restrict the interpretation of set variables X to range not over arbitrary sets of nodes but over branches. We could equally consider quantification over arbitrary paths, but as noted by Hafer and Thomas [12] this would give no difference in expressive power: denoting quantification over arbitrary paths by $\widehat{\exists}$, we have the following obvious translations:

- $\exists X \varphi \;=\; \widehat{\exists} X \exists r \forall x[(r<x \;\vee\; r=x) \;\wedge\; r \in X \;\wedge\; \varphi]$.
- $\widehat{\exists} X \varphi \;=\; \exists X \exists r \varphi'$, where φ' is obtained from φ by replacing all atomic formulas of the form $x \in X$ by $(r<x \;\vee\; r=x) \;\wedge\; x \in X$.

Remark 1.3 (MPL versus first order logic) MPL is in a sense no more expressive than first order logic. Given a tree t, we can consider its *completion* t^c obtained by extending each branch with a limit node, so that each path in t^c has a maximal element. Given any MPL sentence φ, let φ^c be the first order formula obtained from φ by replacing all subformulas $\exists x \alpha$ by $\exists x \exists \ell (x < \ell \wedge \alpha)$, and replacing all subformulas $\exists X \alpha$ by $\exists \ell (\neg \exists x (\ell < x) \wedge \alpha')$ where α' is obtained from α by replacing each occurrence of $x \in X$ by $x < \ell$. Then clearly $t \models \varphi$ iff $t^c \models \varphi^c$.

1.4 Computation Tree Logics

The syntax of the propositional (branching time) computation tree logic CTL* is specified by inductively defining two sets of formulas, ***state formulas*** q and ***path formulas*** p, starting from a finite set of ***atomic propositions*** $\{P_a \;:\; a \in \Sigma\}$ using the path operators $\mathbf{E}p$ ("*there exists a path such that* p"), $\mathbf{X}p$ ("*next time* p") and $p\mathbf{U}p'$ ("p *until* p'"). Formally, these two sets of formulas are given by the following equations:

$$q \;::=\; P_a \;\mid\; q \wedge q' \;\mid\; \neg q \;\mid\; \mathbf{E}p$$
$$p \;::=\; q \;\mid\; p \wedge p' \;\mid\; \neg p \;\mid\; \mathbf{X}p \;\mid\; p\mathbf{U}p'$$

CTL* then consists of the set of state formulas q generated by the above rules. Further common temporal operators can then be introduced as abbreviations; for example: $\mathbf{A}p$ ("*for all paths,* p") abbreviates $\neg\mathbf{E}\neg p$; $\mathbf{F}p$ ("*eventually* p") abbreviates $\underline{\mathrm{true}}\mathbf{U}p$; and $\mathbf{G}p$ ("*always* p") abbreviates $\neg\mathbf{F}\neg p$.

The set of path formulas not involving the $\mathbf{E}$ operator defines the propositional linear time logic LTL. It can be more succinctly defined as the set of formulas given by the following equation:

$$p \;::=\; P_a \;\mid\; p \wedge p' \;\mid\; \neg p \;\mid\; \mathbf{X}p \;\mid\; p\mathbf{U}p'$$

CTL* formulas are interpretted over trees, and LTL formulas are interpretted over paths, by way of a satisfaction relation $\models$. Given a tree t, a node s in this tree, and a path π through this tree, we write $(t, s) \models q$ to mean that state formula q is true at node s in the tree t, and $(t, \pi) \models p$ to mean that path formula p is true of the path π in t. This relation is defined inductively as follows.

$$
\begin{array}{lcll}
(t,s) & \models & \mathsf{P}_a & \text{iff} \;\; t(s) = a \\
(t,s) & \models & q \wedge q' & \text{iff} \;\; (t,s) \models q \;\text{ and }\; (t,s) \models q' \\
(t,s) & \models & \neg q & \text{iff} \;\; (t,s) \not\models q \\
(t,s) & \models & \mathbf{E}p & \text{iff} \;\; \text{there is a path } \pi \text{ in } t \text{ rooted at } s \\
 & & & \phantom{\text{iff}} \;\; \text{such that } (t,\pi) \models p
\end{array}
$$

$$
\begin{array}{lcll}
(t,\pi) & \models & q & \text{iff} \;\; (t,\pi_1) \models q \\
(t,\pi) & \models & p \wedge p' & \text{iff} \;\; (t,\pi) \models p \;\text{ and }\; (t,\pi) \models p' \\
(t,\pi) & \models & \neg p & \text{iff} \;\; (t,\pi) \not\models p \\
(t,\pi) & \models & \mathbf{X}p & \text{iff} \;\; \text{length}(\pi)>1 \;\text{ and }\; (t,\pi^2) \models p \\
(t,\pi) & \models & p\mathbf{U}p' & \text{iff} \;\; \text{there is } i \text{ with } 1 \leq i \leq \text{length}(\pi) \\
 & & & \phantom{\text{iff}} \;\; \text{such that } (t,\pi^i) \models p' \text{ and} \\
 & & & \phantom{\text{iff}} \;\; (t,\pi^k) \models p \;\; \text{whenever} \;\; 1 \leq k < i.
\end{array}
$$

Note that most authors consider only *total* trees, thus stipulating that all paths are infinite. We do not make such a restriction, but our semantic definitions coincide with the usual interpretation for total trees. Also worth noting is that some authors use a slightly different yet equally expressive version of the until operator $\widehat{\mathbf{U}}$, where $p\widehat{\mathbf{U}}p'$ makes no restrictions on the initial state. Thus this operator translates into our use as $p\widehat{\mathbf{U}}p' = \mathbf{X}p\mathbf{U}p'$; and conversely our operator can be translated as $p\mathbf{U}p' = p' \vee (p \wedge p\widehat{\mathbf{U}}p')$.

As LTL formulas are actually interpretted over paths, they can equally be interpretted over words $w = w_1 w_2 \cdots w_n \in \Sigma^*$ and ω-sequences $w = w_1 w_2 \cdots \in \Sigma^\omega$ by adapting the definition of the satisfaction relation $\models$ as follows. (As with paths, the ith letter of $w \in \Sigma^* \cup \Sigma^\omega$ is denoted w_i and we use $w^i = w_i w_{i+1} \cdots$ to denote the suffix of w starting at the ith letter.)

$$
\begin{array}{lcll}
w & \models & \mathsf{P}_a & \text{iff} \;\; w_1 = a \\
w & \models & p \wedge p' & \text{iff} \;\; w \models p \;\text{ and }\; w \models p' \\
w & \models & \neg p & \text{iff} \;\; w \not\models p \\
w & \models & \mathbf{X}p & \text{iff} \;\; \text{length}(w)>1 \;\text{ and }\; w^2 \models p \\
w & \models & p\mathbf{U}p' & \text{iff} \;\; \text{there is } i \text{ with } 1 \leq i \leq \text{length}(w) \\
 & & & \phantom{\text{iff}} \;\; \text{such that } w^i \models p' \text{ and} \\
 & & & \phantom{\text{iff}} \;\; w^k \models p \;\; \text{whenever} \;\; 1 \leq k < i.
\end{array}
$$

It is straightforward to verify the correctness of this interpretation, in the sense that for any path $\pi = \langle \pi_1, \pi_2, \ldots \rangle$ in any tree t, and for any LTL formula p, we have that $(t,\pi) \models p$ iff $t(\pi) \models p$, where $t(\pi) = t(\pi_1)t(\pi_2)\cdots$.

The following is an important result relating LTL and FOL$(<,\Sigma)$ due to Kamp [15]. (See [6] for a more accessible proof of this result.)

Theorem 1.4 (Kamp) *LTL and FOL$(<,\Sigma)$ are equally expressive:*

1. *Given any LTL formula p there is an equivalent FOL$(<,\Sigma)$ sentence φ_p:*

$$\text{for every } w \in \Sigma^* \cup \Sigma^\omega, \;\; w \models p \;\; \textit{iff} \;\; w \models \varphi_p.$$

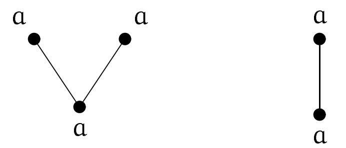
2. *Given any FOL$(<,\Sigma)$ sentence φ there is an equivalent LTL formula p_φ:*

$$\text{for every } w \in \Sigma^* \cup \Sigma^\omega, \;\; w \models \varphi \;\; \textit{iff} \;\; w \models p_\varphi.$$

Remark 1.5 (Binary trees versus general trees) The logics FOL$(<,\Sigma)$, MPL, and CTL* all obey different laws when interpretted over binary trees than when interpretted over arbitrary trees. For example, the following pair of distinct CTL* formulas are equivalent over the class of binary trees (assuming that $a \neq b$): $\mathbf{EX}\mathsf{P}_a \wedge \mathbf{EX}\mathsf{P}_b$ and $\mathbf{EX}\mathsf{P}_a \wedge \mathbf{EX}\mathsf{P}_b \wedge \mathbf{AX}(\mathsf{P}_a \vee \mathsf{P}_b)$. Furthermore, Hafer and Thomas [12] demonstrate that over full binary trees, MPL and CTL* coincide; however, this result fails immediately in general. This is due to the fact that CTL* respects bisimulation equivalence (any two bisimulation equivalent trees must satisfy the same CTL* formulas), but MPL, and indeed already first order logic, sentences do not. (This does not contradict the result of Hafer and Thomas, as we noted in Remark 1.2 that two full binary trees are only bisimulation equivalent when they are actually isomorphic, and MPL certainly respects isomorphism.) For example, Hafer and Thomas consider the first order sentence

$$\exists x \exists y [\neg(x<y \;\vee\; x=y \;\vee\; y<x) \;\wedge\; x \in \mathsf{P}_a \;\wedge\; y \in \mathsf{P}_a]$$

which expresses that there exist two incomparable nodes labelled a. They note that over binary trees this is equivalent to the CTL* formula $\mathbf{EF}(\mathbf{AXEF}\mathsf{P}_a)$; however, this sentence cannot be translated into CTL* in general, as it is satisfied by only the first of the following two bisimulation equivalent trees:



2 Equivalences and Games

Given two trees t and t', we write $t \equiv_n t'$ if no MPL sentence of quantifier depth n can distinguish between these trees. Formally, $t \equiv_n t'$ if and only if for any MPL sentence φ with $\text{qd}(\varphi) \leq n$ we have $t \models \varphi$ iff $t' \models \varphi$. Equally, we write $(t,s) \equiv_n (t',s')$ if no MPL formula $\varphi(x)$ with $\text{qd}(\varphi) \leq n$ can distinguish between these trees with specified nodes; and finally we write $(t,\pi) \equiv_n (t',\pi')$ if no MPL formula $\varphi(X)$ with $\text{qd}(\varphi) \leq n$ can distinguish between these trees with specified branches (full paths).

The relations $\equiv_n$ are clearly equivalence relations over trees, trees with specified nodes, and trees with specified branches, and as indicated in [12] they enjoy the following important properties.

Lemma 2.1

1. *For each* n*, the relation* $\equiv_n$ *defines finitely many equivalence classes* $T_1, T_2, \ldots, T_m$ *of trees; that is,* $t \equiv_n t'$ *iff* $t, t' \in T_i$ *for some* $i \in \{1, 2, \ldots, m\}$.

2. *For each equivalence class* T_i *there is a MPL sentence* β_i *with* $\mathrm{qd}(\beta_i) \leq n$ *which characterises it; that is,* $t \in T_i$ *iff* $t \models \beta_i$.

3. *Every MPL sentence* φ *with* $\mathrm{qd}(\varphi) \leq n$ *is equivalent to a (finite) disjunction of the characterising sentences* β_i.

(The lemma also holds for trees with a specified node or branch and MPL sentences with one free variable with quantifier depth bounded by n. However, for ease of presentation, we only explicitly describe the case for trees and sentences.)

The proof of the above lemma is easy once you realize that there are only finitely many semantically-distinct formulas with at most one free variable of a fixed quantifier depth n. This fact itself can be shown easily by induction on quantifier depth.

The equivalences $\equiv_n$ have an elegant characterisation in terms of the following ***Ehrenfeucht-Fraissé game***. The game is played by two players on two trees t and t', and involves the first player choosing a node or branch in one of the two trees, after which the second player responds by choosing the same type of object (node or branch) in the other tree which she believes 'matches' the object chosen by the first player. After n rounds, there will be n nodes and branches $(s_1, \ldots, s_k, \pi_{k+1}, \ldots, \pi_n)$ selected in the first tree and n corresponding nodes and branches $(s'_1, \ldots, s'_k, \pi'_{k+1}, \ldots, \pi'_n)$ selected in the second tree. The second player is deemed the winner if the mapping $s_i \mapsto s'_i$ and $\pi_j \mapsto \pi'_j$ respects the relations $<$, $\in$, and $\in P_a$. If the second player has a ***winning strategy***, that is, a strategy to follow when choosing her responses to the first player's moves which will guarantee her a win, then we say that t and t' are ***n-game equivalent***, and we write $t \sim_n t'$. The relations $(t, s) \sim_n (t', s')$ and $(t, \pi) \sim_n (t', \pi')$ are defined analogously, where the mapping is extended with $s \mapsto s'$ in the first instance and $\pi \mapsto \pi'$ in the second instance. The characterisation theorem is then as follows. (For a proof, we refer to [3, 13].)

Theorem 2.2 $\equiv_n = \sim_n$. *That is,*

- $t \equiv_n t'$ *iff* $t \sim_n t'$*;*
- $(t, s) \equiv_n (t', s')$ *iff* $(t, s) \sim_n (t', s')$*; and*
- $(t, \pi) \equiv_n (t', \pi')$ *iff* $(t, \pi) \sim_n (t', \pi')$.

3 The Composition Theorem

Composition Theorems are tools which reduce sentences about some compound structure to sentences about its parts. A seminal example of such a result is the Feferman-Vaught Theorem [5] which reduces the first-order theory of generalised products to the first order theory of its factors. Composition theorems for theories of orderings, used as an alternative to the automata-theoretic approach popularized by Büchi, were first explored by Läuchli [16], and subsequently developed greatly by Shelah [20]. The technique was used in a series of papers by Gurevich and Shelah [7, 9, 10, 11], and outlined in a survey exposition by Gurevich [8]. Hafer and Thomas [12] provide a composition theorem for MPL over full binary trees, and in the present paper we prove a composition theorem for MPL over wide trees. Thomas [21] provides a recent overview on using composition theorems where he suggests that, despite their success, such techniques are still largely ignored by the theoretical computer science community in favour of the well-established automata-theoretic techniques. He emphasizes the importance of the approach for decidability questions, though it is evident that it is of importance as well to questions of definability, as in the present paper.

Referring to Lemma 2.1, with n fixed, we can fix m as well as the equivalence classes $T_1, T_2, \ldots, T_m$ and sentences $\beta_1, \beta_2, \ldots, \beta_m$ as given in the lemma. We then define the extended alphabet $\Sigma' = \Sigma \times \mathcal{P}\{1, 2, \ldots, m\}$. Given a tree t and a prefix π of a branch in the tree, we denote by $\nu(t, \pi)$ the (ω-)word over Σ' of length equal to that of π whose ith letter is given by:

$$\nu(t, \pi)_i \quad = \quad \Big(t(\pi_i),\ \{j\ :\ \pi_i \to s \text{ with } t_s \in T_j\}\Big).$$

That is, the ith letter of $\nu(t, \pi)$ is labelled (a, J) if and only if the ith node π_i in the path π is labelled by $a \in \Sigma$, and the subtrees of this node are drawn precisely from the equivalence classes $\{T_j\ :\ j \in J\}$. We shall also use the notation $\nu(t, s)$ where s is a node in the tree t to mean $\nu(t, \pi)$ where π is the (partial) path leading from the root of the tree to s. The importance of $\nu(t, s)$ and $\nu(t, \pi)$ is that they capture the whole of (t, s) and (t, π), respectively, with respect to the distinguishing power of MPL formulas of quantifier depth n. This fact is formulated in terms of games in the following.

Lemma 3.1 *Given a wide tree* t *with node* s *and branch* π*, and a wide tree* t' *with node* s' *and branch* π'*:*

1. *if* $\nu(t, s) \sim_n \nu(t', s')$ *then* $(t, s) \sim_n (t', s')$*;*
2. *if* $\nu(t, \pi) \sim_n \nu(t', \pi')$ *then* $(t, \pi) \sim_n (t', \pi')$.

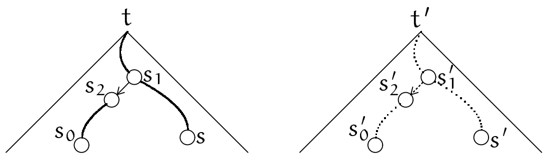
Proof We shall prove only the first result, as the proof of second result is virtually identical.

A winning strategy for player II in the game played on trees (t, s) and (t', s') can be based directly on a winning strategy for player II in the game played on words $\nu(t, s)$ and $\nu(t', s')$ as follows. Assume that some number $i < n$ of rounds have been played in the tree game, and that the first player is about to choose a new node or branch.

- If the first player chooses a node on $\nu(t, s)$, then the second player simply chooses the corresponding node on $\nu(t', s')$ as dictated by the strategy for the word game.

  A symmetric strategy applies if the first player chooses a node on $\nu(t', s')$.

- If the first player chooses a node s_0 in t not on $\nu(t, s)$, then the second player looks at the last node s_1 on



  $\nu(t, s)$ which is an ancestor of the chosen node, and finds the corresponding node s_1' on $\nu(t', s')$ as dictated by the strategy for the word game. The matching node s_0' will come from a subtree rooted at a successor node s_2' of s_1' which comes from the same equivalence class as the subtree rooted at the successor node s_2 of s_1 on the (partial) path to s_0; this is possible, since s_1 and s_1' must have the same label from Σ', and hence have subtrees drawn from the same equivalence classes. As these are wide trees, there are infinitely-many such subtrees; the one which we take is the one which contains the elements chosen previously by one of the two players which correspond to elements chosen previously by the other player which are in the subtree rooted at s_2. If there are no previously-chosen elements in the subtree rooted at s_2, then we choose s_2' so that the subtree below it also contains no previously-chosen elements. Note that this argument relies on the fact that we have wide trees. The particular choice for s_0' is then dictated by the strategy for the game played on these subtrees which are drawn from the same equivalence class, and hence admit a winning strategy in the game for the second player.

  A symmetric strategy applies if the first player chooses a node in t' not on $\nu(t', s')$.

- If the first player chooses a branch π in t, then similar to the previous case, the second player looks at the last node s_1 on $\nu(t, s)$ which is on the branch π, and finds the corresponding node s_1' on $\nu(t', s')$ as dictated by the strategy for the word game. Assuming that the branch π is not a finite path ending at s (in which case the matching branch π' is the finite path ending at s'), the matching branch π' will run through a subtree rooted at a successor node s_2' of s_1' which comes from the same equivalence class as the subtree rooted at the successor node s_2 of s_1 on the branch π; this is possible, since s_1 and s_1' must have the same label from Σ', and hence have subtrees drawn from the same equivalence classes. As these are wide trees, there are infinitely-many such subtrees; the one which we take is the one which contains the elements chosen previously by one of the two players which correspond to elements chosen previously by the other player which are in the subtree rooted at s_2. If there are no previously-chosen elements in the subtree rooted at s_2, then we choose s_2' so that the subtree below it also contains no previously-chosen elements. Note that this argument relies on the fact that we have wide trees. The particular choice for π' is then dictated by the strategy for the game played on these subtrees which are drawn from the same equivalence class, and hence admit a winning strategy in the game for the second player.

  A symmetric strategy applies if the first player chooses a branch in t'.

It is clear that this indeed provides a winning strategy for the second player. $\square$

With this lemma in place, we can now state and prove our Composition Theorem.

Theorem 3.2 (Composition Theorem for wide trees)

1. *For every MPL formula* $\varphi(x)$ *with* $\mathrm{qd}(\varphi) \leq n$*, there is a* $\mathrm{FOL}(<, \Sigma')$ *sentence* ψ *with* $\mathrm{qd}(\psi) \leq n$ *such that for all wide trees* t *and all nodes* s *in* t *we have* $(t, s) \models \varphi(x)$ *if and only if* $\nu(t, s) \models \psi$.

2. *For every MPL formula* $\varphi(X)$ *with* $\mathrm{qd}(\varphi) \leq n$*, there is a* $\mathrm{FOL}(<, \Sigma')$ *sentence* ψ *with* $\mathrm{qd}(\psi) \leq n$ *such that for all wide trees* t *and all branches* π *in* t *we have* $(t, \pi) \models \varphi(X)$ *if and only if* $\nu(t, \pi) \models \psi$.

With this theorem, we are thus reducing any property $\varphi(x)$ *or* $\varphi(X)$ *of a wide tree to an equivalent property* ψ *of some* (ω-)*word.*

Proof Again we only prove the first result, as the proof of the second result is virtually identical.

Let $\varphi(x)$ with $\mathrm{qd}(\varphi) \leq n$ be fixed. Then let

- $\alpha_1(x), \ldots, \alpha_m(x)$ be formulas that define the $\equiv_n$-equivalence classes of Σ-labelled trees with a specified node, as given in Lemma 2.1 (for the case of trees with a specified node). In particular, by Lemma 2.1(3) we have that $\varphi(x) \leftrightarrow \bigvee_{i \in I} \alpha_i(x)$ for some $I \subseteq \{1, \ldots, m\}$;

- $\beta_1, \ldots, \beta_k$ be sentences that define the $\equiv_n$-equivalence classes of Σ'-labelled words;

- $J_i = \Big\{ j : (t,s) \models \alpha_i(x)$ and $\nu(t,s) \models \beta_j$ for some $(t,s) \Big\}$ for each $i \in \{1, \ldots, m\}$ (note that by Lemma 3.1 these sets must be disjoint); and finally let

- $\psi = \bigvee_{i \in I} \bigvee_{j \in J_i} \beta_j$.

We shall demonstrate that this ψ satisfies the conditions of the theorem.

Given any wide tree t with node s,

$$\begin{array}{lll} & (t,s) \models \varphi(x) & \\ \text{iff} & (t,s) \models \alpha_i(x) & \text{for some } i \in I \\ \text{iff} & \nu(t,s) \models \beta_j & \text{for some } i \in I \text{ and } j \in J_i \\ \text{iff} & \nu(t,s) \models \psi. & \end{array}$$

Finally, as $\nu(t,s)$ is a word, the formula ψ can be assumed to be in FOL$(<, \Sigma')$, since any path quantifiers would be redundant and can be removed. □

Remark 3.3 (More general trees) As noted in Remark 1.1, this Composition Theorem actually holds for arbitrary well-founded wide trees. In fact, by maintaining multisets of subtrees, we can prove a more general Composition Theorem for arbitrary well-founded trees.

4 The Expressive Completeness of CTL*

In this section we demonstrate the expressive completeness of CTL* with respect to MPL sentences which respect bisimulation equivalence. We start by noting the following easy direction, which is given as Proposition 4.1 in [12].

Lemma 4.1 *Given any CTL* formula q there is an equivalent MPL sentence φ_q; that is, for every tree t, $(t, \varepsilon_t) \models q$ if and only if $t \models \varphi_q$.*

It is well-known (and easily demonstrated) that CTL* respects bisimulation equivalence, so the MPL sentence φ_q in the above Lemma must also respect bisimulation equivalence. The reverse translation follows as a corollary of the following lemma for wide trees, the proof of which relies on our Composition Theorem 3.2.

Lemma 4.2 *Given any MPL sentence φ there is a CTL* formula q_φ which is equivalent to it over the class of wide trees; that is, for every wide tree t, $t \models \varphi$ if and only if $(t, \varepsilon_t) \models q_\varphi$.*

Proof The proof of this result is by induction on the quantifier depth of φ. The result is easily obtained for qd$(n) = 1$. For example, if $\varphi = \exists x(x \in \mathsf{P}_a)$ then $q_\varphi = \mathbf{EF}\mathsf{P}_a$.

In the induction step, we assume that any MPL sentence with quantifier depth no greater than n is equivalent to some CTL* formula over wide trees. The only cases of interest are $\varphi = \exists x \varphi'(x)$ and $\varphi = \exists X \varphi'(X)$, where qd$(\varphi') = n$.

By Lemma 2.1 we have m equivalence classes $T_1, T_2, \ldots, T_m$ of trees with respect to the equivalence relation $\equiv_n$, characterised by MPL sentences $\beta_1, \beta_2, \ldots, \beta_m$, each of quantifier depth no greater than n, and hence by induction each equivalent to some CTL* formula $q_1, q_2, \ldots, q_m$, respectively, over the class of wide trees.

Consider the case where $\varphi = \exists x \varphi'(x)$ with qd$(\varphi') \leq n$. We can apply the first part of the Composition Theorem 3.2 to the subformula $\varphi'(x)$ to get a FOL$(<, \Sigma')$ sentence ψ with qd$(\psi) \leq n$ such that for all wide trees t and all nodes s of t we have $(t,s) \models \varphi'(x)$ iff $\nu(t,s) \models \psi$. Let $\psi' = \exists z \psi^{\leq z}$ where $\psi^{\leq z}$ is obtained from ψ by replacing all subformulas $\exists x \alpha$ by $\exists x(x \leq z \wedge \alpha)$; then clearly $\exists z : \nu(t,z) \models \psi$ iff $\exists \pi : \nu(t, \pi) \models \psi'$. By Kamp's Theorem 1.4, we can translate this first-order sentence ψ' into an equivalent LTL formula p.

The formula p involves atomic propositions of the form $\mathsf{P}_{(a,I)}$ with $(a, I) \in \Sigma'$; we wish to replace each of these atomic propositions by a suitable CTL* formula $q_{(a,I)}$ expressing that the node of interest satisfies the atomic predicate P_a and that its successor nodes are represented precisely by the equivalence classes T_i with $i \in I$; that is: each successor satisfies some q_i with $i \in I$, and for each $i \in I$ there is a successor which satisfies q_i. The substitution is thus as follows:

$$q_{(a,I)} \quad = \quad \mathsf{P}_a \;\wedge\; \bigwedge_{i \in I} \mathbf{EX} q_i \;\wedge\; \mathbf{AX} \bigvee_{i \in I} q_i$$

Our desired CTL* formula is then $\mathbf{E}p'$, where p' denotes the CTL* path formula which we obtain from p after performing the above substitutions: given any wide tree, t,

$$\begin{array}{lll} t \models \varphi & \text{iff} & \exists s : (t,s) \models \varphi'(x) \\ & \text{iff} & \exists s : \nu(t,s) \models \psi \\ & \text{iff} & \exists \pi : \nu(t,\pi) \models \psi' \\ & \text{iff} & \exists \pi : \nu(t,\pi) \models p \\ & \text{iff} & \exists \pi : (t,\pi) \models p' \\ & \text{iff} & (t, \varepsilon_t) \models \mathbf{E}p' \end{array}$$

A simpler argument based on the second part of the Composition Theorem 3.2 (not requiring the quantifier relativisation step) handles the case where $\varphi = \exists X \varphi'(X)$ with qd$(\varphi') \leq n$. □

Corollary 4.3 *Given any MPL sentence φ which is invariant under bisimulation equivalence, there is an equivalent*

CTL formula* q_φ*; that is, for every tree* t, $t \models \varphi$ *if and only if* $(t, \varepsilon_t) \models q_\varphi$.

Proof Given φ, the appropriate q_φ is as given in Lemma 4.2: given any t, let t^w be a bisimulation equivalent wide tree. Then

$$\begin{array}{lll} t \models \varphi & \text{iff} \;\; t^w \models \varphi & \text{(by assumption on } \varphi\text{,} \\ & & \text{ given } t \sim t^w) \\ & \text{iff} \;\; (t^w, \varepsilon_{t^w}) \models q_\varphi & \text{(by Lemma 4.2)} \\ & \text{iff} \;\; (t, \varepsilon_t) \models q_\varphi & \text{(since CTL* respects} \\ & & \text{ bisimulation equivalence).} \end{array}$$

□

Finally, combining Lemma 4.1 and Corollary 4.3 gives us our desired result:

Theorem 4.4 (Expressive Completeness of CTL*) *CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.*

5 Concluding Remarks

5.1 On the Role of Bisimulation Equivalence

The properties of bisimulation equivalence which we exploit in our argument are the following.

1. Every tree is bisimulation equivalent to some wide tree.

2. CTL* does not distinguish between bisimulation equivalent trees.

In fact these are the only properties which we exploit; this is true also of Janin and Walukiewicz [14] (with CTL* replaced by μ-calculus). Thus the role of *bisimulation* equivalence is not so crucial in these arguments: any equivalence which satisfies the above properties could play the role of bisimulation equivalence. In general, if we let $\equiv$ be any equivalence such that every tree is $\equiv$-equivalent to some wide tree, then given any logic $\mathcal{L}$ defined on trees its $\equiv$-invariant properties can be studied on wide trees.

5.2 On an Automata-Theoretic Proof

The result of Janin and Walukiewicz [14] that the propositional μ-calculus coincides with the bisimulation invariant properties expressible in the whole of MSOL is closely related to our expressive completeness result. In order to get their result, Janin and Walukiewicz rely on a lemma (Lemma 12 in [14]) corresponding to our Lemma 4.2 that for every MSOL sentence φ there is a μ-calculus formula q_φ which is equivalent to it over the class of wide trees. This is motivated by the observation made above regarding the role of bisimulation in the result. (They use the term "ω-expanded" rather than "wide".) However, as we noted in the Introduction, the proofs of these lemmas are different: whereas their proof relies on an automata-theoretic characterisation of the bisimulation-invariant properties expressible in MSOL, our proof is automata-free, based instead on our Composition Theorem 3.2.

We may well ask if there is an automata-theoretic proof of our result. For this, we would first need to identify a natural class of automata which has the same expressive power on wide trees as MPL (or equivalently CTL*). As we noted in Remark 1.3, MPL is closely related to monadic first-order logic, and the counter-free automata of McNaughton and Papert [17] have the same expressive power on words as monadic first-order logic. Therefore one might look for a generalization of counter-free automata in order to provide an automata-theoretic proof of the expressive completeness of CTL*.

5.3 On Complexity

The proof presented in this paper is constructive, and from it one can extract translation algorithms between CTL* and MPL. The translation from CTL* to MPL is compositional and has liner-time complexity. An upper bound for the complexity of our translation algorithm from MPL to CTL* can be extracted from a careful analysis of the proofs of the Composition Theorem 3.2 and Lemma 4.2. The best upper bound which we were able to extract is non-elementary. (Recall that a function f is non-elementary if there is no m such that $f(n)$ is less than $\exp_m(n)$ for all sufficiently large n, where

$$\exp_m(k) \;=\; \left.2^{2^{2^{\cdot^{\cdot^{\cdot^{k}}}}}}\right\} m$$

is the m-times iterated exponential function. The reason for such an enormous upper bound is that the number of n-equivalence classes may be on the order of $\exp_n(2)$.

5.4 On the Role of **X** and **U**

The proof of the main Lemma 4.2, presenting the translation from MPL to CTL*, makes explicit use of the *next time* operator **X**; it also makes use of the *until* operator **U** but only implicitly through invoking Kamp's Theorem 1.4. It is worth observing at this point that we can replace **X** and **U** in the argument with *any* set of modalities which is expressively complete with respect to monadic first-order logic of order over the class of linear orders, that is, for which Kamp's Theorem holds.

References

[1] J.F.A.K. van Bentham (1976). *Modal Correspondence Theory*. PhD Thesis. Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam.

[2] E.M. Clarke and E.A. Emerson (1981). Design and verification of synchronous skeletons using branching time temporal logic. *Lecture Notes in Computer Science* **131**:52–71.

[3] H.-D. Ebbinghaus and J. Flum (1995). *Finite Model Theory*. Springer Perspectives in Mathematical Logic.

[4] E.A. Emerson and J.Y. Halpern (1986). 'Sometimes' and 'not never' revisited: on branching versus linear time temporal logics. *Journal of the ACM* **33**(1):151–178.

[5] S. Feferman and R.L. Vaught (1959). The first-order properties of products of algebraic systems. *Fundamenta Mathematica* **47**:57–103.

[6] D. Gabbay, I. Hodkinson and M. Reynolds (1994). *Temporal Logic*. Oxford University Press.

[7] Y. Gurevich (1979). Modest theory of short chains I. *Journal of Symbolic Logic* **44**:481–490.

[8] Y. Gurevich (1985). Monadic second-order theories. In *Model-Theoretic Logics*, (J. Barwise and S. Feferman, eds.), pp479-506, Springer-Verlag.

[9] Y. Gurevich and S. Shelah (1979). Modest theory of short chains II. *Journal of Symbolic Logic* **44**:491–502.

[10] Y. Gurevich and S. Shelah (1979). Rabin's uniformization problem. *Journal of Symbolic Logic* **48**:1105–1119.

[11] Y. Gurevich and S. Shelah (1985). The decision problem for branching time logic. *Journal of Symbolic Logic* **50**(3):668–681.

[12] T. Hafer and W. Thomas (1987). Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *Proceedings of ICALP'87: International Colloquium on Automata, Languages and Programming*, *Lecture Notes in Computer Science* **267**:269–279, Springer-Verlag.

[13] W. Hodges (1993). *Model Theory*. Cambridge University Press.

[14] D. Janin and I. Walukiewicz (1996). On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *Proceedings of CONCUR'96: International Conference on Concurrency Theory*, *Lecture Notes in Computer Science* **1119**:263–277, Springer-Verlag.

[15] H.W. Kamp (1968). *Tense Logic and the Theory of Linear Order*. PdD Thesis, University of California, Los Angeles.

[16] H. Läuchli (1968). A decision procedure for the weak second order theory of linear order. In *Contributions to Mathematical Logic, Proceedings of Logic Colloquium Hanover 1966*, North-Holland.

[17] R. McNaughton and S. Papert (1971). *Counter-Free Automata*. MIT Press.

[18] R. Milner (1989). *Communication and Concurrency*. Prentice-Hall.

[19] D.M.R. Park (1981). Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science* **104**:168–183.

[20] S. Shelah (1975). The monadic theory of order. *Annals of Mathematics* **102**:379–419.

[21] W. Thomas (1997). Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science: A Selection of Essays in Honor of A. Ehrenfeucht*, *Lecture Notes in Computer Science* **1261**:118-143, Springer-Verlag.