

# Counting on CTL\*: On the Expressive Power of Monadic Path Logic

Faron Moller

*Department of Computer Science, University of Wales Swansea, Singleton Park,  
Sketty, Swansea SA2 8PP, Great Britain*

Alexander Rabinovich

*School of Computer Science, Tel Aviv University, Ramat Aviv, Tel Aviv 69978,  
Israel*

---

## Abstract

Monadic second-order logic (MSOL) provides a general framework for expressing properties of reactive systems as modelled by trees. Monadic path logic (MPL) is obtained by restricting second-order quantification to paths reflecting computation sequences. In this paper we show that the expressive power of MPL over trees coincides with the usual branching time logic CTL\* embellished with a simple form of counting. As a corollary, we derive an earlier result that CTL\* coincides with the bisimulation-invariant properties of MPL. In order to prove the main result, we first prove a new Composition Theorem for trees.

---

## 1 Introduction

Various temporal logics have been proposed for reasoning about so-called “reactive” systems, computer hardware or software systems which exhibit (potentially) non-terminating and nondeterministic behaviour. Such a system is typically represented by the 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 rooted tree, with the initial state of the system represented by the root of the tree.

Various equivalences have also been proposed between such systems, depicting when two systems should be deemed the same. Given such an equivalence, it is 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. This is often but not always the case with popular temporal logics. For example, monadic second-order logic (MSOL) fails this criterion: it is simple to express properties even within first-order logic (FOL) which distinguish between behaviours which are equivalent with respect to any reasonable notion. However, this doesn't preclude MSOL from being considered the "mother of all temporal logics" and being employed as the base language for verification tools such as MONA [1] which have been successfully used in practical verification exercises. This deficiency also affects monadic path logic (MPL) which is MSOL in which set quantification is restricted to paths. The importance of MPL in the study of temporal logics is evidenced by the observation made by John Burgess that the decidability problems for various branching-time logics are reducible to the decidability problem for MPL.

It is often easy to establish the relative expressive strengths of various temporal logics, as encodings from one to another are often straightforward. However, it is necessary to study and understand these relationships more fully in order to guide the choice as to which are appropriate for given applications. For example, it is straightforward to encode propositional modal logic in FOL, but van Benthem [2] goes further and shows that propositional modal logic coincides with the class of FOL properties which do not distinguish between bisimulation equivalent behaviours. One interpretation of this result is that, if you are interested in bisimulation-preserving properties, then propositional modal logic may be preferred over FOL. A related result due to Janin and Walukiewicz [18] shows that the propositional  $\mu$ -calculus coincides with the bisimulation invariant properties expressible in MSOL. Also, in [22] we show that the branching time logic  $\text{CTL}^*$  [3,5] coincides with the bisimulation invariant properties expressible in MPL.

In this paper, we re-examine this last result from a different perspective and consider what facility must be added to  $\text{CTL}^*$  to attain the expressive strength of MPL. The answer turns out to be elegant: a simple notion of counting added to  $\text{CTL}^*$  gives it precisely the same expressive power as MPL. This result seems natural, as bisimilarity ignores multiplicities: one particular future behaviour is indistinguishable from several equivalent future behaviours. Already in [25] Walukiewicz noted the role that such a counting mechanism played in studying the expressive power of temporal logics, specifically MSOL. However, it is by no means *a priori* clear that this is the full extent to which MPL differs from its bisimulation-invariant fragment.

In the remainder of this introduction, we provide the relevant definitions for trees and logics. 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, a Com-

position Theorem for trees. After that, we apply our Composition Theorem to the problem of showing our expressivity result.

### 1.1 Computation Trees

A **tree**  $t$  with **root**  $\varepsilon_t$  consists of a partially-ordered set of **nodes**  $S$  in which the ancestors of any given node  $s \in S$  constitute a well-founded total order with minimal element  $\varepsilon_t$ . Such a tree represents a **computation**, as defined as a sequence of transitions between states; a node in the tree corresponds to a state in a computation, and its ancestors  $\{s' \in S : s' < s\}$  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)  $\varepsilon_t$ . We denote the (immediate) successor relation by  $\rightarrow$ , so that  $s \rightarrow s'$  if and only if  $s < s'$  and there is no  $s''$  with  $s < s'' < s'$ . (If  $s < s'$  then the well-foundedness condition ensures the existence of an immediate successor  $s \rightarrow s'' \leq s'$ .) 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 (transfinite) sequence of transitions  $\varepsilon_t \rightarrow s_1 \rightarrow s_2 \rightarrow \dots \rightarrow s$ .

Furthermore, the nodes of a tree are labelled by elements taken from some finite set  $\Sigma$ , representing the atomic properties which are true at the given state of the computation. With this in mind, we define a  **$\Sigma$ -valued tree** to be a function  $t : S \rightarrow \Sigma$ . (In the literature, states are sometimes labelled by subsets of a set  $\Delta$  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  $\Sigma$  simply by taking  $\Sigma$  to be the collection of all subsets of  $\Delta$ .)

A **path** through a tree  $t$  is a maximal (finite or transfinite) sequence of successive nodes  $\pi = \langle s_0, s_1, s_2, \dots \rangle$  through the tree; that is,  $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$ . (We occasionally use the term “path” to refer to a non-maximal sequence of successive nodes ending at a specified node, but we shall always be explicit with such uses.) If the initial node of a path is the root  $\varepsilon_t$ , then the path is referred to as a **full path** or **branch**. The  $i$ th node  $s_i$  in the path  $\pi$  is denoted by  $\pi_i$ , and we use  $\pi^i = \langle \pi_i, \pi_{i+1}, \pi_{i+2}, \dots \rangle$  to denote the subpath of  $\pi$  rooted at  $\pi_i$ . (In particular,  $\pi = \pi^0$ .) Finally, we use  $t_s$  to denote the subtree of  $t$  rooted at the node  $s$ .

In the literature, various restrictions are often placed on trees. A tree is said to be **total** if each node has a successor; all paths through a total tree are thus infinite. An  **$\omega$ -tree** is a tree in which each node has only a finite number of ancestors; all paths through an  $\omega$ -tree are thus either finite or of length  $\omega$ .

We do not impose any such restrictions except where we explicitly state them.

## 1.2 Monadic Second-Order Logic

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

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

$$(t, s_1, \dots, s_m, S_1, \dots, S_n) \models \varphi(x_1, \dots, x_m, X_1, \dots, X_n)$$

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

We also interpret  $\text{FOL}(<, \Sigma)$  sentences over sequences  $w$  of elements of  $\Sigma$ , 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  $\pi = \langle \pi_0, \pi_1, \pi_2, \dots \rangle$  (that is, every node has at most one successor), and for any  $\text{FOL}(<, \Sigma)$  sentence  $\varphi$ , we have that  $t \models \varphi$  iff  $t(\pi) \models \varphi$ , where  $t(\pi) = t(\pi_0)t(\pi_1)t(\pi_2) \dots$ .

**Monadic path logic** MPL is defined to be the monadic second-order logic as described, but where we restrict the interpretation of set variables 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 [15] this would give no difference in expressive power: denoting quantification over arbitrary paths by  $\hat{\exists}$ , we have the following obvious translations:

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

**Remark 1 (MPL versus first-order logic)** MPL is in a sense no more expressive than first-order logic (cf. Proposition 1 of [13]). 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  $\varphi$ , let  $\varphi^c$  be the first-order formula obtained from  $\varphi$  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  $\alpha'$  is obtained from  $\alpha$  by replacing each occurrence of  $x \in X$  by  $x < \ell$ . Then clearly  $t \models \varphi$  iff  $t^c \models \varphi^c$ .

### 1.3 Counting-CTL\*: CTL\* with Counting

The syntax of the branching time computation tree logic CTL\* (*with counting*) 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 **E** $p$  (“there exists a path such that  $p$ ”), **X** $p$  (“next time  $p$ ”) and  $p$ **U** $p'$  (“ $p$  until  $p'$ ”), along with the counting operator **D** <sup>$n$</sup>  $q$  with  $n > 0$  (“for (at least)  $n > 0$  different successors  $q$ ”). Formally, these two sets of formulas are given by the following equations:

$$\begin{aligned} q &::= P_a \mid q \wedge q' \mid \neg q \mid \mathbf{E}p \mid \mathbf{D}^n q \\ p &::= q \mid p \wedge p' \mid \neg p \mid \mathbf{X}p \mid p\mathbf{U}p' \end{aligned}$$

Counting-CTL\* then consists of the set of state formulas  $q$  generated by the above rules; the basic language of CTL\* consists of the state formulas not involving the counting operator **D** <sup>$n$</sup> . Further common temporal operators are introduced as abbreviations; for example: **A** $p$  (“for all paths,  $p$ ”) abbreviates  $\neg\mathbf{E}\neg p$ ; **F** $p$  (“eventually  $p$ ”) abbreviates  $\text{true}\mathbf{U}p$ ; and **G** $p$  (“always  $p$ ”) abbreviates  $\neg\mathbf{F}\neg p$ .

The set of path formulas not involving the **E** operator (nor the counting operator **D** <sup>$n$</sup> ) defines the propositional linear time logic LTL. It can be more succinctly defined as the set of formulas given by the following equation:

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

Counting-CTL\* formulas are interpreted over trees, and LTL formulas are interpreted over paths, by way of a satisfaction relation  $\models$ . Given a tree  $t$ , a node  $s$  in this tree, and a path  $\pi$  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  $\pi$  in  $t$ . This relation is defined inductively as follows.

$$\begin{aligned}
(t, s) \models \mathbf{P}_a & \quad \text{iff } t(s) = a \\
(t, s) \models q \wedge q' & \quad \text{iff } (t, s) \models q \text{ and } (t, s) \models q' \\
(t, s) \models \neg q & \quad \text{iff } (t, s) \not\models q \\
(t, s) \models \mathbf{E}p & \quad \text{iff there is a path } \pi \text{ in } t \text{ rooted at } s \text{ such that } (t, \pi) \models p \\
(t, s) \models \mathbf{D}^n q & \quad \text{iff } s \rightarrow s' \text{ and } (t, s') \models q \text{ for (at least) } n \text{ different nodes } s' \\
(t, \pi) \models q & \quad \text{iff } (t, \pi_0) \models q \\
(t, \pi) \models p \wedge p' & \quad \text{iff } (t, \pi) \models p \text{ and } (t, \pi) \models p' \\
(t, \pi) \models \neg p & \quad \text{iff } (t, \pi) \not\models p \\
(t, \pi) \models \mathbf{X}p & \quad \text{iff } \text{length}(\pi) > 1 \text{ and } (t, \pi^1) \models p \\
(t, \pi) \models p\mathbf{U}p' & \quad \text{iff there is } i \text{ with } 0 \leq i < \text{length}(\pi) \text{ such that} \\
& \quad (t, \pi^i) \models p' \text{ and } (t, \pi^k) \models p \text{ whenever } 0 \leq k < i.
\end{aligned}$$

Note that most authors (particularly in the verification community) consider only total  $\omega$ -trees. We do not make such a restriction here, but our semantic definitions coincide with the usual interpretation over total  $\omega$ -trees. Also worth noting is that some authors use a slightly different version of the until operator  $\widehat{\mathbf{U}}$ , where  $p\widehat{\mathbf{U}}p'$  makes no restrictions on the initial state. In the presence of  $\mathbf{X}$  these two operators are equally expressive: the alternative operator can be translated into ours 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 interpreted over paths, they can just as well be interpreted over sequences  $w = w_0w_1w_2 \cdots$  over  $\Sigma$  by adapting the definition of the satisfaction relation  $\models$  as follows. (As with paths, the  $i$ th letter of  $w$  is denoted  $w_i$  and we use  $w^i = w_iw_{i+1} \cdots$  to denote the suffix of  $w$  starting at the  $i$ th letter.)

$$\begin{aligned}
w \models \mathbf{P}_a & \quad \text{iff } w_0 = a \\
w \models p \wedge p' & \quad \text{iff } w \models p \text{ and } w \models p' \\
w \models \neg p & \quad \text{iff } w \not\models p \\
w \models \mathbf{X}p & \quad \text{iff } \text{length}(w) > 1 \text{ and } w^1 \models p \\
w \models p\mathbf{U}p' & \quad \text{iff there is } i \text{ with } 0 \leq i < \text{length}(w) \text{ such that} \\
& \quad w^i \models p' \text{ and } w^k \models p \text{ whenever } 0 \leq k < i.
\end{aligned}$$

It is straightforward to verify the correctness of this interpretation, in the sense that for any path  $\pi = \langle \pi_0, \pi_1, \pi_2, \dots \rangle$  in any tree  $t$ , and for any LTL formula  $p$ , we have that  $(t, \pi) \models p$  iff  $t(\pi) \models p$ , where  $t(\pi) = t(\pi_0)t(\pi_1)t(\pi_2) \dots$ .

The following is an important result relating LTL and  $\text{FOL}(<, \Sigma)$  due to Kamp [19]. (More accessible proofs of this result can be found in [8,7,17].)

**Theorem 2 (Kamp)** *Over  $\omega$ -sequences, LTL and  $\text{FOL}(<, \Sigma)$  are equally expressive:*

- (1) *Given any LTL formula  $p$  there is an equivalent  $\text{FOL}(<, \Sigma)$  sentence  $\varphi_p$ :  
for every  $\Sigma$ -labelled  $\omega$ -sequence  $w$ ,  $w \models p$  iff  $w \models \varphi_p$ .*
- (2) *Given any  $\text{FOL}(<, \Sigma)$  sentence  $\varphi$  there is an equivalent LTL formula  $p_\varphi$ :  
for every  $\Sigma$ -labelled  $\omega$ -sequence  $w$ ,  $w \models \varphi$  iff  $w \models p_\varphi$ .*

## 2 Logical 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  $\varphi$  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 [15] they enjoy the following important properties.

### Lemma 3

- (1) *For each  $n$ , the relation  $\equiv_n$  defines finitely-many equivalence classes  $T_1, T_2, \dots, T_m$  of trees; that is,  $t \equiv_n t'$  iff  $t, t' \in T_i$  for some  $i \in \{1, 2, \dots, m\}$ .*
- (2) *For each equivalence class  $T_i$  there is a MPL sentence  $\beta_i$  with  $\text{qd}(\beta_i) \leq n$  which characterises it; that is,  $t \in T_i$  iff  $t \models \beta_i$ .*
- (3) *Every MPL sentence  $\varphi$  with  $\text{qd}(\varphi) \leq n$  is equivalent to a (finite) disjunction of the characterising sentences  $\beta_i$ .*

(The lemma also holds for trees with a specified node or branch and MPL formulas 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  $(s_1, \dots, s_k, \pi_{k+1}, \dots, \pi_n)$  selected in the first tree and  $n$  corresponding nodes and branches  $(s'_1, \dots, s'_k, \pi'_{k+1}, \dots, \pi'_n)$  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  $\in \mathcal{P}_a$ . 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 [4,16].)

**Theorem 4**  $\equiv_n = \sim_n$ . *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 A Composition Theorem for Trees

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 [6] which reduces the first-order theory of generalised products to the first-order theory of its factors. Composition theorems for theories of orderings were first explored by Läuchli [20], and subsequently developed by Shelah [23]. The technique was used in a series of papers by Gurevich and Shelah [9,11,12,14], and outlined in a survey exposition by Gurevich [10]. Thomas [24] provides an 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 3, with  $n$  fixed, we can fix  $m$  as well as the equivalence classes  $T_1, T_2, \dots, T_m$  and sentences  $\beta_1, \beta_2, \dots, \beta_m$  as given in the lemma. We then define the extended alphabet

$$\Sigma' = \Sigma \times \left( \{1, \dots, m\} \rightarrow \{0, \dots, n\} \right) \times \{0, 1, \dots, m\}.$$

Given a tree  $t$  and a (prefix of a) branch  $\pi$  in the tree, we denote by  $v(t, \pi)$  the sequence over  $\Sigma'$  of length equal to that of  $\pi$  whose  $i$ th letter is given by:

$$v(t, \pi)_i = \left( t(\pi_i), f_{\pi_i}, k \right),$$

where for each  $x$  with  $1 \leq x \leq m$ ,

$$f_{\pi_i}(x) = \max \left\{ j \leq n : \pi_i \rightarrow s \text{ with } t_s \in T_x \text{ for } j \text{ different nodes } s \notin \pi \right\};$$

and  $t_{\pi_{i+1}} \in T_k$  (or  $k=0$ , if  $i = \text{length}(\pi)$ ). That is, the  $i$ th letter of  $v(t, \pi)$  is  $(a, f, k)$ , where

- the  $i$ th node  $\pi_i$  in the path  $\pi$  is labelled by  $a \in \Sigma$ ;
- $\pi_i$  has (at least)  $f(x)$  different successors not on the path  $\pi$  (i.e., other than  $\pi_{i+1}$ ) which are the roots of subtrees in the class  $T_x$ ;
- if  $f(x) < n$  then  $\pi_i$  has exactly  $f(x)$  different successors not on the path  $\pi$  which are the roots of subtrees in the class  $T_x$ ;
- if  $\pi_{i+1}$  exists (that is,  $\text{length}(\pi) > i$ ), then  $k$  is defined such that the subtree rooted at  $\pi_{i+1}$  is in the class  $T_k$ ; otherwise (if  $\text{length}(\pi) = i$ )  $k=0$ .

We also use the notation  $v(t, s)$  where  $s$  is a node in the tree  $t$  to mean  $v(t, \pi)$  where  $\pi$  is the (partial) path leading from the root of the tree to  $s$ . The importance of  $v(t, s)$  and  $v(t, \pi)$  is that they capture the whole of  $(t, s)$  and  $(t, \pi)$ , 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 5** *Given a tree  $t$  with node  $s$  and branch  $\pi$ , and a tree  $t'$  with node  $s'$  and branch  $\pi'$ :*

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

**PROOF.** We prove only the first result, as the proof of the second result is virtually identical.

A winning strategy for the second player in the game played on trees  $(t, s)$  and  $(t', s')$  can be based directly on a winning strategy for the second player in the game played on words  $v(t, s)$  and  $v(t', s')$  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. We shall assume the following property holds, along with its symmetric version (interchanging  $t$  and  $t'$ ), and show that it remains invariant:

*If  $s_1 \leq s$ , and  $s'_1 \leq s'$  is the node corresponding to  $s_1$  according to the winning strategy in the word game, and  $s_1 \rightarrow s_2$  with  $s_2 \not\leq s$ , then  $s'_1 \rightarrow s'_2$  for some  $s'_2 \not\leq s'$  such that the subtrees rooted at  $s_2$  and  $s'_2$  come from the same equivalence class and contain corresponding previously-selected nodes and paths.*

This property is certainly true at the start (when  $i=0$ , i.e., before any elements are chosen), as  $s_1$  and  $s'_1$  must have the same label from  $\Sigma'$  and hence have subtrees not on the path leading to  $s$  drawn from the same equivalence classes.

- If the first player chooses a node on  $v(t, s)$ , then the second player simply chooses the corresponding node on  $v(t', s')$  as dictated by the strategy for the word game. The invariant is clearly maintained.

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

- If the first player chooses a node  $s_0$  in  $t$  not on  $v(t, s)$ , then the second



player looks at the last node  $s_1$  on  $v(t, s)$  which is an ancestor of the chosen node, and the node  $s_2$  with  $s_1 \rightarrow s_2 \leq s_0$ , and takes the nodes  $s'_1$  and  $s'_2$  as described in the invariant. The node  $s'_0$  will be chosen by the second player to be in the subtree rooted at  $s'_2$ , thus maintaining the invariant. (In particular, taking the equal counts provided by the labelling of  $s_1$  and  $s'_1$  into consideration, if one has a successor not on  $v(t, s)$  which is the root of a subtree containing no previously-selected objects, then the other must have such a successor as well.) 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  $v(t', s')$ .

- If the first player chooses a branch  $\pi$  in  $t$ , then similar to the previous case, the second player looks at the last node  $s_1$  on  $v(t, s)$  which is on the branch  $\pi$ , and the node  $s_2$  on the branch  $\pi$  with  $s_1 \rightarrow s_2$  and takes the nodes  $s'_1$  and  $s'_2$  as described in the invariant. (If the branch  $\pi$  happens to be a finite path ending at  $s$ , then the matching branch  $\pi'$  is the finite path ending at  $s'$ .) The matching branch  $\pi'$  will be chosen by the second player to be in the subtree rooted at  $s'_2$ , thus maintaining the invariant as in the previous case. The particular choice for  $\pi'$  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.  $\square$

It is worth noting that the proof of this lemma does not refer to the last component of the new labelling of nodes on paths; this component will only have effect in the proof of Lemma 8.

With Lemma 5 in place, we can now state and prove our Composition Theorem.

**Theorem 6 (Composition Theorem)**

(1) For every MPL formula  $\varphi(x)$  with  $\text{qd}(\varphi) \leq n$ , there is a FOL( $<, \Sigma'$ ) sentence  $\psi$  with  $\text{qd}(\psi) \leq n$  such that for all trees  $t$  and all nodes  $s$  in  $t$  we have

$$(t, s) \models \varphi(x) \text{ if and only if } v(t, s) \models \psi.$$

(2) For every MPL formula  $\varphi(X)$  with  $\text{qd}(\varphi) \leq n$ , there is a FOL( $<, \Sigma'$ ) sentence  $\psi$  with  $\text{qd}(\psi) \leq n$  such that for all trees  $t$  and all branches  $\pi$  in  $t$  we have

$$(t, \pi) \models \varphi(X) \text{ if and only if } v(t, \pi) \models \psi.$$

With this theorem, we are thus reducing any property  $\varphi(x)$  or  $\varphi(X)$  of a tree to an equivalent property  $\psi$  of a sequence.

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

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

- $\alpha_1(x), \dots, \alpha_m(x)$  be formulas that define the  $\equiv_n$ -equivalence classes of  $\Sigma$ -labelled trees with a specified node, as given in Lemma 3 (for the case of trees with a specified node). In particular, by Lemma 3(3) we have that  $\varphi(x) \leftrightarrow \bigvee_{i \in I} \alpha_i(x)$  for some  $I \subseteq \{1, \dots, m\}$ ;
- $\beta_1, \dots, \beta_k$  be sentences that define the  $\equiv_n$ -equivalence classes of  $\Sigma'$ -labelled words;
- $J_i = \{j : (t, s) \models \alpha_i(x) \text{ and } v(t, s) \models \beta_j \text{ for some } (t, s)\}$  for each  $i \in \{1, \dots, m\}$  (note that by Lemma 5 these sets must be disjoint); and finally
- $\psi = \bigvee_{i \in I} \bigvee_{j \in J_i} \beta_j$ .

We shall demonstrate that this  $\psi$  satisfies the conditions of the theorem.

Given any tree  $t$  with node  $s$ ,

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

Finally, as  $v(t, s)$  is a word, the formula  $\psi$  can be assumed to be in  $\text{FOL}(<, \Sigma')$ , since any path quantifiers would be redundant and can be removed.  $\square$

## 4 The Expressiveness of MPL

In this section we demonstrate that the expressiveness of MPL coincides with that of Counting-CTL\*. We start by simply noting the easily-established direction.

**Lemma 7** *Given any Counting-CTL\* formula  $q$  there is an equivalent MPL sentence  $\varphi_q$ ; that is, for every tree  $t$ ,  $(t, \varepsilon_t) \models q$  if and only if  $t \models \varphi_q$ .*

The existence of a translation in the opposite direction relies as follows from our Composition Theorem 6.

**Lemma 8** *Given any MPL sentence  $\varphi$  there is an equivalent Counting-CTL\* formula  $q_\varphi$  over  $\omega$ -trees; that is, for every  $\omega$ -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  $\varphi$ . The result is easily obtained for  $\text{qd}(n) = 1$ . For example, if  $\varphi = \exists x(x \in P_a)$  then  $q_\varphi = \mathbf{EFP}_a$ .

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

By Lemma 3 we have  $m$  equivalence classes  $T_1, T_2, \dots, T_m$  of trees with respect to the equivalence relation  $\equiv_n$ , characterised by MPL sentences  $\beta_1, \beta_2, \dots, \beta_m$ , each of quantifier depth no greater than  $n$ , and hence by induction each equivalent to some Counting-CTL\* formula  $q_1, q_2, \dots, q_m$ , respectively.

Consider the case where  $\varphi = \exists x\varphi'(x)$  with  $\text{qd}(\varphi') \leq n$ . We can apply the first part of the Composition Theorem 6 to the subformula  $\varphi'(x)$  to get a  $\text{FOL}(<, \Sigma')$  sentence  $\psi$  with  $\text{qd}(\psi) \leq n$  such that for all trees  $t$  and all nodes  $s$  of  $t$  we have  $(t, s) \models \varphi'(x)$  iff  $v(t, s) \models \psi$ . Let  $\psi' = \exists z\psi^{\leq z}$  where  $\psi^{\leq z}$  is

obtained from  $\psi$  by replacing all subformulas  $\exists x\alpha$  by  $\exists x(x \leq z \wedge \alpha)$ ; then clearly  $\exists z : v(t, z) \models \psi$  iff  $\exists \pi : v(t, \pi) \models \psi'$ . By Kamp's Theorem 2, we can translate this first-order sentence  $\psi'$  into an equivalent LTL formula  $p$ .

The LTL formula  $p$  involves atomic propositions of the form  $P_{(a,f,k)}$  with  $(a, f, k) \in \Sigma'$ ; we wish to replace each such atomic proposition by a suitable Counting-CTL\* path formula  $q_{(a,f,k)}$  expressing that:

- the node of interest satisfies the atomic predicate  $P_a$ ;
- for  $1 \leq i \leq m$  with  $i \neq k$ , there are at least  $f(i)$  successor nodes in equivalence class  $T_i$  (ie, satisfying  $q_i$ ); and exactly  $f(i)$  such successor nodes in the case when  $f(i) < n$ ;
- if  $k \neq 0$ , then there are at least  $f(k)+1$  successor nodes in equivalence class  $T_k$  (ie, satisfying  $q_k$ ); and exactly  $f(i)+1$  such successor nodes in the case when  $f(k) < n$ ; and
- if  $k \neq 0$ , then the next state is in equivalence class  $k$  (ie, satisfies  $q_k$ ).

The substitution is thus as follows (for ease, we break it up into cases).

- in the case where  $k=0$ ,  $q_{(a,f,k)}$  is given as follows:  

$$P_a \wedge \bigwedge_{1 \leq i \leq m} \mathbf{D}^{f(i)} q_i \wedge \bigwedge_{\substack{1 \leq i \leq m \\ f(i) < n}} \neg \mathbf{D}^{f(i)+1} q_i$$
- in the case where  $k \neq 0$  and  $f(k) = n$ ,  $q_{(a,f,k)}$  is given as follows:  

$$P_a \wedge \bigwedge_{\substack{1 \leq i \leq m \\ i \neq k}} \mathbf{D}^{f(i)} q_i \wedge \bigwedge_{\substack{1 \leq i \leq m \\ i \neq k, f(i) < n}} \neg \mathbf{D}^{f(i)+1} q_i \wedge \mathbf{D}^{f(k)+1} q_k \wedge \mathbf{X} q_k$$
- in the case where  $k \neq 0$  and  $f(k) < n$ ,  $q_{(a,f,k)}$  is given as follows:  

$$P_a \wedge \bigwedge_{\substack{1 \leq i \leq m \\ i \neq k}} \mathbf{D}^{f(i)} q_i \wedge \bigwedge_{\substack{1 \leq i \leq m \\ i \neq k, f(i) < n}} \neg \mathbf{D}^{f(i)+1} q_i \wedge \mathbf{D}^{f(k)+1} q_k \wedge \neg \mathbf{D}^{f(k)+2} q_k \wedge \mathbf{X} q_k$$

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

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

A simpler argument based on the second part of the Composition Theorem 6 (not requiring the quantifier relativisation step) handles the case where  $\varphi = \exists X \varphi'(X)$  with  $\text{qd}(\varphi') \leq n$ .  $\square$

## 5 Related Results

### 5.1 CTL\* versus bisimulation-invariant MPL

The main result in the paper complements our earlier result [22] that CTL\* coincides with the set of bisimulation invariant properties expressible in MPL. The proof of the earlier result follows the same compositional approach as the proof of the present result, and exploits the fact that every tree  $t$  is bisimulation equivalent to a so-called *wide* tree  $t^w$ , one in which for every transition  $s \rightarrow s'$  there are infinitely-many transitions  $s \rightarrow s''$  such that  $t_{s'}$  is isomorphic to  $t_{s''}$ . Every MPL property  $\varphi$  is shown to be equivalent over the class of wide trees to some CTL\* formula  $q_\varphi$ ; assuming then that the property  $\varphi$  is bisimulation invariant, we get that  $t \models \varphi$  iff  $t^w \models \varphi$  iff  $(t^w, \varepsilon_{t^w}) \models q_\varphi$  iff  $(t, \varepsilon_t) \models q_\varphi$ .

We can extract this earlier result as a corollary of the new result as follows. Given a formula of MPL, we translate this into an equivalent formula of Counting-CTL\*. If the original MPL formula respects bisimulation, then so must this translated formula of Counting-CTL. We can thus safely replace each occurrence of  $\mathbf{D}^n q$  with  $\mathbf{EX}q$  to get an equivalent formula of CTL\*. (The proof of this is by a simple induction on the structure of formulas.)

Hafer and Thomas [15] demonstrated the correspondence between MPL and CTL\* over full binary trees, again using a suitable composition theorem. This result is simpler in that bisimilarity between full binary trees is trivial: two full binary trees are bisimulation equivalent only if they are isomorphic, and MPL certainly respects isomorphism. Indeed, the result for full binary trees follows from the present result, since over full binary trees, the counting operators add no power to CTL\*:  $\mathbf{D}^1 q$  is equivalent to  $\mathbf{EX}q$ ;  $\mathbf{D}^2 q$  is equivalent to  $\mathbf{AX}q$ ; and  $\mathbf{D}^n q$  for  $n > 2$  is equivalent to false.

### 5.2 The Gurevich-Shelah Composition Theorem

In [13] Gurevich and Shelah reduce the decidability problem for MPL over the class of all trees to the decidability problem for the first-order theory over the class of well-founded binary trees. They then showed this latter problem to be decidable in [14] with the aid of a composition theorem. Our composition

theorem is reminiscent of the one given by Gurevich and Shelah in [14]. Their composition theorem, however, deals with partial first-order theories of binary trees, and we were unable to reduce our composition theorem to it. Below we briefly describe the Gurevich-Shelah composition theorem and its relation to ours.

In [14] first order structures with partial elements are considered. These are structures in which some constant names can be interpreted as undefined elements. Given two such structures  $t$  and  $t'$ , we write  $t \equiv_n^{GS} t'$  if no first order sentence of quantifier depth  $n$  can distinguish between these structures. For the equivalence  $\equiv_n^{GS}$  the analogue of Lemma 3 holds. In particular, for each  $n$ , the relation  $\equiv_n^{GS}$  defines finitely-many equivalence classes  $T_1, T_2, \dots, T_m$  of trees; that is,  $t \equiv_n^{GS} t'$  iff  $t, t' \in T_i$  for some  $i \in \{1, 2, \dots, m\}$ .

The Gurevich-Shelah composition theorem deals with binary trees. Referring to Lemma 3, let  $n$  be fixed, and let equivalence classes  $T_1, T_2, \dots, T_m$  be the  $\equiv_n^{GS}$  equivalence classes over binary  $\Sigma$ -labelled trees. Define the new alphabet

$$\Sigma' = \{0, 1, \dots, m\}.$$

Given a tree  $t$  and a (prefix of a) branch  $\pi$  in the tree, let  $t_i$  be the subtree of  $t$  over the set of nodes  $\{s : \pi_i \leq s \wedge \neg(\pi_{i+1} \leq s)\}$ . We denote by  $v^{GS}(t, \pi)$  the sequence over  $\Sigma'$  of length equal to that of  $\pi$  whose  $i$ th letter is  $k$  iff  $t_i$  is in the  $k$ th  $\equiv_n^{GS}$ -equivalence class,  $t_i \in T_k$ . The Gurevich-Shelah composition theorem then states that the truth value of a first order sentence  $\varphi$  of quantifier depth  $n$  in a tree  $t$  can be effectively reduced to the truth values of another sentence  $\psi$  of quantifier depth  $n$  over  $v^{GS}(t, \pi)$ .

To summarize, the Gurevich-Shelah composition theorem:

- (1) deals with only binary trees while our composition theorem considers trees with arbitrary branching;
- (2) considers structures with partial elements while our theorem considers standard structures; and
- (3) deals with first-order formulas while our theorem deals with monadic path logic formulas.

The first point is not very essential; the Gurevich-Shelah composition theorem can be modified for trees with arbitrary branching. We do not fully understand the impact of the second point. The third point, however, is essential. By Remark 1 there is a translation of MPL into first-order logic; however this translation does not preserve the quantifier depth of formulas. The composition theorem and our translation of MPL into CTL\* with counting operators is sensitive to the quantifier depth of formulas. This is the main reason that we were unable to reduce our composition theorem to the Gurevich-Shelah theorem.

### 5.3 The $\mu$ -calculus versus bisimulation-invariant MSOL

In [18] Janin and Walukiewicz define automata both for recognizing MSOL properties and for recognizing  $\mu$ -calculus properties. The MSOL-automata differ from those for the  $\mu$ -calculus in that they involve elementary counting operations similar to those introduced in Counting-CTL\*. With this, they are able to prove that the  $\mu$ -calculus expresses exactly the bisimulation-invariant properties of MSOL, analogous to the result above relating CTL\* to bisimulation-invariant MPL. Though this paper does not raise the question of adding such a counting mechanism to the syntax of the  $\mu$ -calculus to derive a calculus matching MSOL in expressive power, Walukiewicz [26] confirms that such a result holds.

The present result relating MPL to Counting-CTL\* cannot, however, be derived from the automata-theoretic proof relating MSOL to the  $\mu$ -calculus, as it is unclear what the appropriate automata for MPL would be. To prove our result using automata, we would first need to identify a natural class of automata which has the same expressive power as MPL. As we noted in Remark 1, MPL is closely related to monadic first-order logic, and the counter-free automata of McNaughton and Papert [21] 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.

Equally, it is not clear how one might extend the compositional approach to the MSOL case. Thus the MSOL result cannot be derived from the compositional proof used here for MPL.

## References

- [1] A. Abdelwaheb and D. Basin (2000). Bounded model construction for monadic second-order logics. In *Proceedings of CAV'00: International Conference on Computer-Aided Verification, Lecture Notes in Computer Science* **1855**:99–113, Springer-Verlag.
- [2] J.F.A.K. van Benthem (1976). *Modal Correspondence Theory*. PhD Thesis. Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam.
- [3] 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.
- [4] H.-D. Ebbinghaus and J. Flum (1995). *Finite Model Theory*. Springer Perspectives in Mathematical Logic.

- [5] 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.
- [6] S. Feferman and R.L. Vaught (1959). The first-order properties of products of algebraic systems. *Fundamenta Mathematica* **47**:57–103.
- [7] D. Gabbay, I. Hodkinson and M. Reynolds (1994). *Temporal Logic*. Oxford University Press.
- [8] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi (1980). On the temporal analysis of fairness. *7th Annual Symposium on Principles of Programming Languages*, pp163–173.
- [9] Y. Gurevich (1979). Modest theory of short chains I. *Journal of Symbolic Logic* **44**:481–490.
- [10] Y. Gurevich (1985). Monadic second-order theories. In *Model-Theoretic Logics*, (J. Barwise and S. Feferman, eds.), pp479-506, Springer-Verlag.
- [11] Y. Gurevich and S. Shelah (1979). Modest theory of short chains II. *Journal of Symbolic Logic* **44**:491–502.
- [12] Y. Gurevich and S. Shelah (1979). Rabin’s uniformization problem. *Journal of Symbolic Logic* **48**:1105–1119.
- [13] Y. Gurevich and S. Shelah (1985). To the decision problem for branching time logic. In *Foundations of Logic and Linguistics: Problems and their Solutions*, (P. Weingartner and G. Dorn, eds.), pp181-198, Plenum.
- [14] Y. Gurevich and S. Shelah (1985). The decision problem for branching time logic. *Journal of Symbolic Logic* **50**(3):668–681.
- [15] 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.
- [16] W. Hodges (1993). *Model Theory*. Cambridge University Press.
- [17] I. Hodkinson (1995). *Expressive completeness of Until and Since over dedekind complete linear time*. In *Modal Logic and Process Algebra: A Bisimulation Perspective*, (A. Ponse, M. de Rijke and Y. Venema, eds.), pp171-185, CSLI Publications.
- [18] 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.
- [19] H.W. Kamp (1968). *Tense Logic and the Theory of Linear Order*. PdD Thesis, University of California, Los Angeles.

- [20] 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.
- [21] R. McNaughton and S. Papert (1971). *Counter-Free Automata*. MIT Press.
- [22] F. Moller and A. Rabinovich (1999). On the expressive power of CTL\*. In *Proceedings of LICS'99: The fourteenth IEEE Symposium on Logic in Computer Science*, pp360–369.
- [23] S. Shelah (1975). The monadic theory of order. *Annals of Mathematics* **102**:379–419.
- [24] 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.
- [25] I. Walukiewicz (1996). Monadic second-order logic on tree-like structures. In *Proceedings of STACS'96: International Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science* **1046**:401–414, Springer-Verlag.
- [26] I. Walukiewicz (2001). *Personal Communication*.