Limiting Until in Ordered Tree Query Languages

MICHAEL BENEDIKT, Oxford University Computer Science
CLEMENS LEY, Oxford University Computer Science

Marx and de Rijke have shown that the navigational core of the w3c XML query language XPath is not first-order complete – that is it cannot express every query definable in first-order logic over the navigational predicates. How can one extend XPath to get a first-order complete language? Marx has shown that Conditional XPath – an extension of XPath with an "Until" operator – is first order complete. The completeness argument makes essential use of the presence of upward axes in Conditional XPath. We examine whether it is possible to get "forward-only" languages that are first-order complete for Boolean queries on ordered trees. It is easy to see that a variant of the temporal logic CTL* is first-order complete; the variant has path quantifiers for downward, leftward and rightward paths, while along a path one can check arbitrary formulas of linear temporal logic (LTL). This language has two major disadvantages: it requires path quantification in *both* horizontal directions (in particular, it requires looking backward at the prior siblings of a node), and it requires the consideration of formulas of LTL of arbitrary complexity on vertical paths. This last is in contrast with Marx's Conditional XPath, which requires only the checking of a single Until operator on a path. We investigate whether either of these restrictions can be eliminated. Our main results are negative ones. We show that if we restrict our CTL* language by having an until operator in only one horizontal direction, then we lose completeness. We also show that no restriction to a "small" subset of LTL along vertical paths is sufficient for first order completeness. Smallness here means of bounded "Until Depth", a measure of complexity of LTL formulas defined by Etessami and Wilke. In particular, it follows from our work that Conditional XPath with only forward axes is not expressively complete; this extends results proved by Rabinovich and Maoz in the context of infinite unordered trees.

ACM Reference Format:
Benedikt, M. and Ley, C. 2011. Limiting Until in Ordered Tree Languages ACM Trans. Comput. Logic ?, ?, Article ? (????), 34 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Introduction

On finite words, the relationship of first order logic to modal languages – ones with operators for navigating forward and backward in a word – is well-understood. A starting point is the language Linear Temporal Logic (LTL), which defines formulas that hold at the beginning of a word. Formulas can be built up from atomic propositions (i.e. node labels) via Boolean operators and the modalities U (until), X (next), and F (eventually). Indeed, it suffices to have only one modality, the "strong" variant of until (which we use in this work)[Gabbay et al. 1980]: $\varphi \mathsf{U} \psi$ is true at the beginning of a word if there is a proper suffix β of the word such that ψ is true at β and φ is true on every suffix properly containing β. A refinement of Kamp's Theorem [Kamp 1968] shows that LTL is *first-order complete* over words – it can express every property of words that is

Author's addresses: M. Benedikt and Clemens Ley, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

definable in first-order logic, where there are unary predicates for the word labels and binary predicates for the ordering relation.

Is the full power of the until operator necessary? Etessami and Wilke [Etessami and Wilke 2000] showed that it is. In particular they define a notion of "until-depth" by restricting the number of nestings of the U operator. The main result of [Etessami and Wilke 2000] is that the subsets UD_k formed by restricting the number of nested U operators to k form a strict hierarchy in expressiveness on words. More precisely, they show this on infinite words, and use a variant of LTL that contains both past and future operators – their until hierarchy thus looks at the nesting of of both the U operator and its backward analog S (for "Since"). The result for U only follows from their proof.

Let us turn to the situation for ordered trees. XPath [World Wide Web Consortium 1999] is the w3c standard for querying XML documents; the navigational core of XPath is a query language on finite labeled ordered trees. XPath is a model language, analogous to temporal logic with only F (eventually). Marx and de Rijke [Marx and de Rijke 2005] showed that this language is incomplete in a fundamental sense – there are properties expressible in first-order logic over the navigational predicates that XPath cannot express. This incompleteness manifests itself in other shortcomings of XPath. For instance, XPath is not closed under complement of path relations; indeed, Marx has shown [Marx 2005b] that such closure for an extension of XPath is equivalent to first-order completeness. The incompleteness of XPath can be seen as the analog of the result that an U operator is needed on trees.

How can one extend XPath to get a first-order complete language? One answer is given by Marx in [Marx 2005a], who defines a first-order complete query language *Conditional XPath* (CXPath). Roughly speaking, CXPath extends XPath by an "until operator" $\phi\, U\, \psi$: this operator holds at a node in an ordered tree iff there is a finite path from the node that satisfies ϕ up to the end of the path, at which point it must satisfy ψ. Marx considers four kinds of paths: leftward within the siblings of a given node, rightward within the siblings, upward through the ancestors, or downward through some chain of descendants.

CXPath may seem analogous to LTL in adding an U operator. But the analogy is misleading, since the U operator of CXPath is combined with selection of a path. Thus the first-order completeness is a surprising result. Still CXPath has a disadvantage that it involves both "forward navigation" (downward and to the right) and "backward navigation": this makes it less amenable to one-pass evaluation. The forward-only fragment of CXPath, which we denote by $\mathrm{ForCXPath}$, is the variant of CXPath where the paths allowed in the until operator are downward and rightward, with an additional filter added to detect whether a child is the first among its siblings. Is $\mathrm{ForCXPath}$ complete for arbitrary ordered trees? It follows from results of [Benedikt and Jeffrey 2007] that this language is first-order complete over finite ordered trees of any fixed depth. In this paper, we determine that $\mathrm{ForCXPath}$ is *not* first-order complete over arbitrary ordered trees.

Another way of extending results from words to trees is by separating out path quantification and node quantification into separate operators. A natural way to do this it to consider a variant of the temporal logic CTL* [Emerson 1990]. CTL* contains *path formulas*, which hold with respect to a path within a tree, and also *state formulas*, which hold with respect to a node within a tree. Path formulas are built up from state formulas via the LTL operators, while state formulas are built up from atomic propositions via Boolean operators and path quantification. CTL* is not complete over trees, since it has no means of determining the number of children of a node with a certain property. But this ability to distinguish among children is known to be the only obstacle to completeness. For example, if we look at infinite binary trees, where a child is la-

belled as either left or right child, Hafer and Thomas have shown that CTL* extended with predicates for left and right sibling is first-order complete [Hafer and Thomas 1987]. Related results on completeness up to bisimulation equivalence can be found in [Moller and Rabinovich 1999]. To extend this to arbitrary ordered trees, we can distinguish between vertical and horizontal paths, and add a path quantifier for each. A simple variation of the argument of Hafer and Thomas (given in Section 1) shows that this language is first-order complete over ordered trees – this is noted in Barcelo and Libkin's work (Theorem 3.4 of [Barcelo 2005]). Indeed, we will consider a variant $\text{CTL}^*_{\lhd\rhd}$ with downward paths, rightward paths, and leftward paths, where on the horizontal paths we allow only simple until operators, with no nesting – this language will turn out to be first-order complete. $\text{CTL}^*_{\lhd\rhd}$ has two disadvantages: the first is that it requires paths in both horizontal directions, the second is that it requires all of LTL as a sublanguage in the vertical dimension, unlike CXPath. Can we make due without one of these two restrictions?

We give negative answers to these questions. In our first result, we show that one cannot weaken the horizontal navigation to look only in one direction. Indeed one cannot make due with a language that has the U operator in one horizontal direction (in addition to F and X), but only the F and X operators in the other direction.

In our second result, we consider restricting the power in the vertical direction. For any collection $\mathcal{F}$ of formulas over words (e.g. collections of LTL formulas) we let $\text{CTL}^*_{\lhd\rhd}(\mathcal{F})$ be the ordered tree language that restricts $\text{CTL}^*_{\lhd\rhd}$ as follows: node formulas are built up as before from label predicates and a last child test, while being closed under quantification of downward and rightward paths; path formulas are built up by substituting node formulas as propositions within the formulas of $\mathcal{F}$. For example, the language ForCXPath is contained in $\text{CTL}^*_{\lhd\rhd}(\mathcal{F})$ where $\mathcal{F}$ contains all formulas $p\,U\,q$. Our question can now be formalised as: for which subsets $\mathcal{F}$ is $\text{CTL}^*_{\lhd\rhd}(\mathcal{F})$ first-order complete?

We show that if $\mathcal{F}$ is any fragment of LTL with bounded "until-depth", then $\text{CTL}^*_{\lhd\rhd}(\mathcal{F})$ is not first-order complete. In fact, our results show that languages $\text{CTL}^*_{\lhd\rhd}(\mathcal{F}_n)$ where $\mathcal{F}_n$ is the subset of LTL formulas of depth at most n, form a strict hierarchy. In particular it follows from our results that ForCXPath is not first-order complete. Furthermore, it shows that any forward-only first-order complete language must be fairly large.

Related Work. The question of a complete first-order forward-only language was considered in the context of *unordered trees* by Rabinovich and Maoz [Rabinovich and Maoz 2000]. They consider languages of the form $\text{CTL}^*(\mathcal{F})$ where $\mathcal{F}$ is a fragment of first-order logic on ω-words. The main result of [Rabinovich and Maoz 2000] is that $\text{CTL}^*(QR_n)$ is incomplete, where QR_n is the set of first-order formulas of quantifier rank at most n. The results are proven for both finite and infinite trees. In [Rabinovich 2002] an extension is announced, stating that $\text{CTL}^*(AD_n)$ is incomplete, where AD_n is the set of formulas of bounded quantifier-alternation depth.

There has also been work in the ordered case. Barcelo and Libkin consider several logics on ordered trees in [Barcelo 2005]; in the first-order context, the main result is that a variant of CTL^* with quantifiers for vertical and horizontal paths is first-order complete. In [Bojańczyk 2008], Bojańczyk defines a hierarchy of logics using a general notion of a "tree operator" – a tree automaton with "holes" for lower level formula. The result announced in [Bojańczyk 2008] is that no logic based on a finite set of such operators can be first-order complete.

Our notion of "until-depth" is taken from Etessami and Wilke's work [Etessami and Wilke 2000]. They deal with infinite words, and use a variant of LTL that contains both past and future operators. They define a hierarchy within this based on the number

of nestings of the operators U and its backward analog S (for "Since"). The main result of [Etessami and Wilke 2000] is that the subsets UD_k formed by restricting the number of nested U or S operators to k, form a strict hierarchy in expressiveness on words.

Our first result states that the restrictions of $\mathrm{CTL}^*_{\triangleleft\triangleright}$ where the use of U is limited in one of the horizontal directions is incomplete. This contradicts an earlier claim from Barcelo and Libkin ([Barcelo 2005], again Theorem 3.4: the contradiction is only with the "Moreover" addendum). It is incomparable with the results of Bojańczyk and those of Rabinovich and Maoz in [Rabinovich and Maoz 2000], since these works are concerned with operators that "look downwards".

Our second main result is that the languages $\mathrm{CTL}^*_{\triangleleft\triangleright}(UD_k)$ increase in expressiveness as k increases; in our case the until-depth ud is defined by bounding the number of nestings of U in any path formula, while allowing arbitrary nestings of the temporal operators X and F Our proof technique blends the techniques of Etessami and Wilke with that of Rabinovich and Maoz. We compare this result with the prior results in the unordered case. It follows from our results that none of the languages $\mathrm{CTL}^*(UD_k)$ are complete on unordered trees. This in turn implies the incompleteness theorem of [Rabinovich and Maoz 2000], since the sets of formulas QD_k they consider are finite for any fixed vocabulary, and hence each is contained in some UD_k. However, Rabinovich has also shown [Rabinovich 2008] that formulas of Until-depth k have bounded alternation-depth. Combining this with the result on incompleteness of bounded alternation-depth claimed in [Rabinovich 2002] implies the restriction of our result to unordered trees. When restricted to unordered trees, our results are incomparable to those announced by Bojańczyk in [Bojańczyk 2008], since each of our sets $\mathrm{CTL}^*_{\triangleleft\triangleright}(UD_i)$ contains formulas of unbounded Operator Depth.

We note that the case of ordered trees does present new difficulties over the unordered case. Briefly put, our separation theorems, as well as the earlier ones, rely on the construction of families of pairs trees that are "highly indistinguishable" in the syntactically smaller language, but which disagree on a fixed sentence within the syntactically larger language. In the ordered case both the construction of these pairs and the proofs of their equivalence are more difficult, since many properties of the depth of the trees can be distinguished by small formulas. An explanation of the particular difficulties arising in our second result is given in Section 4.

Our results were motivated by the goal of obtaining a natural ordered tree query language that is first-order complete and allows "separation" – unary formulas can be split into uni-directional components. On words, the separation theorem of Gabbay et. al. [Gabbay et al. 1980] shows that LTL has this property. In [Marx 2004], Marx claimed an ordered tree anolog of this, stating that CondXPath formulas can be split as a Boolean combination of pure future and pure past parts – In particular, this would imply that $\mathrm{ForCXPath}$ is first-order complete for nodes at the root (i.e. for Boolean queries). The argument in [Marx 2004] has a flaw, and indeed our main result disproves this claim. This flaw does not impact the main results of Marx in [Marx 2004], which are re-proven by other means in [Marx 2005a]. [Benedikt and Jeffrey 2007] shows that this separation result does hold when the depth of trees is fixed.

Notes and Acknowledgements: This paper is an extended version of the conference paper [Ley and Benedikt 2009]. We are heavily indebted to the referees of both ICDT and TOCL for numerous suggestions corrections.

Organisation: Section 1 defines the languages we deal with formally, and states the main results of the paper. The rest of the paper is dedicated to the proof of our main incompleteness result. Section 2 defines the variant of Ehrenfeucht-Fraïssé games used in the arguments, along with the method of building examples trees that will witness the incompleteness of the languages. Section 3 gives our first main result, about

incompleteness of languages that restrict the use of horizontal navigation. Section 4 gives the second result, concerning incompleteness of languages restricting vertical navigation. Section 5 gives conclusions.

1. FIRST ORDER COMPLETE QUERY LANGUAGES

1.1. Logics for Words

Linear Temporal Logic (LTL) over a set of propositions Σ has formulas built up from the grammar:

$$\varphi = a \mid \neg\varphi \mid \varphi \wedge \varphi \mid X\,\varphi \mid F\,\varphi \mid \varphi\,U\,\varphi$$

where $a \in \Sigma$.

The semantics of LTL is generally defined with respect to infinite words. We will give a variant for finite labelled words: that is, finite sequences $a_1 \dots a_n$ over a finite alphabet Σ. For a word $w = a_1 \dots a_n$, we set $|w| = n$. For $i \leq j \leq |w|$ we let $w[i] = a_i$ and we denote by $w[i,j]$ the infix $a_i \dots a_j$. The suffix $a_i \dots a_n$ of w is denoted by $w[i,*]$. We note that a word $w = a_1 \dots a_n$ can also be considered a structure $(\mathrm{Dom}, \leq, \mathrm{lab}())$ where $\mathrm{Dom} = \{1, \dots, n\}$, $\leq$ is the usual linear order on Dom, and $\mathrm{lab}()$ is a labeling function defined by $\mathrm{lab}(i) = a_i$ for all $i \leq n$.

We now define the semantics of LTL:

$w \vDash a$ iff a is the label of $w[1]$.
$w \vDash X\,\varphi$ iff $|w| > 1$ and $w[2,*] \vDash \varphi$.
$w \vDash F\,\varphi$ iff there is a j with $1 < j \leq |w|$ and $w[j,*] \vDash \varphi$.
$w \vDash \varphi\,U\,\psi$ iff there is a j with $1 < j \leq |w|$ and $w[j,*] \vDash \psi$
and for all i, if $1 < i < j$ then $w[i,*] \vDash \varphi$.

The semantics of LTL over infinite words is analogous.

Here we use the "strong variant" of U, in which $\varphi\,U\,\psi$ asserts the existence of a node satisfying ψ. It is known that the expressiveness of LTL would be unaffected if we had replaced this by the usual notion, which asserts only that if such a node exists, all the nodes to the right of the first satisfy φ [Emerson 1990]. Using this variant X and F become redundant, but they will not be redundant when we restrict the nesting of the U operator. Indeed, since our main negative results are expressive bounds on the fragment where the use of U is restricted, the use of strong until will make our negative results stronger. The use of this form of U will not impact our positive results.

We define the *until-depth* $\mathrm{ud}(\varphi)$ of an LTL formula φ to be the nesting depth of the until-operator:

$$\begin{aligned}
&\mathrm{ud}(a) = 0\\
&\mathrm{ud}(\varphi \wedge \psi) = max\{\mathrm{ud}(\varphi), \mathrm{ud}(\psi)\}\\
&\mathrm{ud}(\neg\varphi) = \mathrm{ud}(F\,\varphi) = \mathrm{ud}(X\,\varphi) = \mathrm{ud}(\varphi)\\
&\mathrm{ud}(\varphi\,U\,\psi) = max\{\mathrm{ud}(\varphi), \mathrm{ud}(\psi)\} + 1
\end{aligned}$$

This is precisely the definition of [Etessami and Wilke 2000], restricted to LTL formulas with only future operators. Note that there are infinitely many semantically different formulas of any fixed until-depth.

We define the next/eventually-depth ned of an LTL formula similarly:

$$\begin{aligned}
&\mathrm{ned}(a) = 0\\
&\mathrm{ned}(\varphi \wedge \psi) = \mathrm{ned}(\varphi\,U\,\psi) = max\{\mathrm{ned}(\varphi), \mathrm{ned}(\psi)\}\\
&\mathrm{ned}(\neg\varphi) = \mathrm{ned}(\varphi)\\
&\mathrm{ned}(F\,\varphi) = \mathrm{ned}(X\,\varphi) = \mathrm{ned}(\varphi) + 1
\end{aligned}$$

Since X and F are redundant for full LTL, the notion of next/eventually-depth will only be interesting when we place additional restrictions – e.g. on the until-depth.

1.2. Logics for Trees

Let Σ be a finite set of labels. An *ordered tree over* Σ consists of
— an acyclic parent/child relation in which every node has in-degree at most one and there is a unique node – called *root* – which has in-degree 0
— a right-sibling relation which is the successor relation of a linear order on the children of any given node
— a labeling function assigning an element of Σ to each node.
We refer to the usual derived notions on ordered trees, such as the ancestor, descendant, following-sibling, and preceding-sibling relations. We do not require ordered trees to be finite, but we will always deal with ordered trees such that there are only finitely many ancestors of any node.

A *downward path* in an ordered tree is a sequence of nodes $p_1 \dots$ in which p_{n+1} is a child of p_n in the tree for all n such that p_n and p_{n+1} exist. A *downward fullpath* is a downward path that contains a leaf node. Similarly, a *rightward path* is a sequence of nodes where $p_1 \dots$ in which p_{n+1} is a right sibling of p_n, and a *rightward fullpath* is a rightward path that contains a node with no right sibling. A *leftward path* and *leftward fullpath* are defined analogously for left siblings. Clearly a path can be considered a word. A *rooted downward fullpath* is a downward fullpath that contains the root.

CTL* – to be defined in a minute – is a known tree logic for unordered trees. We define an extension CTL$^*_{\triangleleft\triangleright}$ of CTL* that can also access the sibling order. For a set of labels Σ, and $k \geq 0$, we inductively define the sets P_k and N_k of CTL$^*_{\triangleleft\triangleright}$ *path formulas* and *node formulas*.

— Boolean combinations of symbols in Σ are the only formulas in N_0.
— Any LTL formula over propositions for formulas in N_k is in P_k.
— If $\varphi \in P_k$ then $\exists_\triangleleft \varphi$, $\exists_\triangledown \varphi$, and $\exists_\triangleright \varphi$ are in N_{k+1}.
— Both N_k and P_k, $k > 0$ are closed under Boolean operations.

CTL$^*_{\triangleleft\triangleright}$ is the union over k of the formulas in N_k and P_k.

Given a tree T, a node x in T, and a paths π in T, we define the semantics of CTL$^*_{\triangleleft\triangleright}$ by induction:

— $T, x \vDash a$ iff x is labelled with a in T
— $T, x \vDash \exists_\triangleleft \varphi$ iff there is a leftward fullpath π starting at x such that $T, \pi \vDash \varphi$.
— $T, x \vDash \exists_\triangledown \varphi$ iff there is a downward fullpath π starting at x such that $T, \pi \vDash \varphi$.
— $T, x \vDash \exists_\triangleright \varphi$ iff there is a rightward fullpath π starting at x such that $T, \pi \vDash \varphi$.
— $T, \pi \vDash \varphi$, for $\varphi \in P_k$ iff $N_k(\pi) \vDash \varphi$, where $N_k(\pi)$ is the labelled linear order formed by labeling each node in π with the formulas of N_k that it satisfies in T.

For a node formula φ, we write $T \vDash \varphi$ iff $T, \mathrm{root}(T) \vDash \varphi$.

We can also consider the language CTL$^*_{\triangleleft\triangle\triangleright}$ formed by allowing a quantifier $\exists_\triangle$; that is, extending the rules for node formulas to say that if $\varphi \in P_k$ then $\exists_\triangle \varphi$ in N_{k+1}. The semantics is analogous to that of CTL$^*_{\triangleleft\triangleright}$, but the quantifier $\exists_\triangle$ quantifies over *upward fullpaths*, which are defined analogously to downward fullpaths (the successor of a node in the sequence must be its parent in the tree).

For ϕ a path formula, let $LTL(\phi)$ be the LTL formula obtained by replacing every node subformula by a proposition. Let $\mathrm{PropSub}(\phi)$ be the node subformulas of ϕ other than ϕ itself.

The downward until depth $\mathrm{ud}_\triangledown(\varphi)$ of a node formula is defined recursively. For a node formula φ of the form $\exists_\triangledown \rho$ is the maximum of the until depth of $LTL(\rho)$ and

the maximum of $\mathrm{ud}_\triangledown(\varphi')$ for $\varphi' \in \mathrm{PropSub}(\varphi)$. For all other node formulas it is the maximum of $\mathrm{ud}_\triangledown(\varphi')$ over proper subformulas.

That is, we look at the maximal until-depth of LTL formulas that are asserted on downward paths. The left-until-depth $\mathrm{ud}_\triangleleft(\varphi)$ and right-until-depth $\mathrm{ud}_\triangleleft(\varphi)$ are defined analogously, considering only until-depth of path formulas on paths quantified within leftward (resp. rightward) path quantifiers, while the until-depth of a node formula considers arbitrary path formulas.

The *next/eventually-depth* $\mathrm{ned}(\varphi)$ of a $\mathrm{CTL}^*_{\triangleleft\triangleright}$ formula φ is the maximal next/eventually-depth of $LTL(\rho)$ where ρ ranges over path subformula of φ.

The *down-path-depth* $\mathrm{pd}_\triangledown(\varphi)$ is the nesting depth of $\exists_\triangledown$-quantifiers within φ. Formally

$$\begin{aligned}
&\mathrm{pd}_\triangledown(a) = 0\\
&\mathrm{pd}_\triangledown(\varphi \wedge \psi) = \mathrm{pd}_\triangledown(\varphi\, U\, \psi) = \max\{\mathrm{pd}_\triangledown(\varphi), \mathrm{pd}_\triangledown(\psi)\}\\
&\mathrm{pd}_\triangledown(\neg\varphi) = \mathrm{pd}_\triangledown(F\,\varphi) = \mathrm{pd}_\triangledown(X\,\varphi) = \mathrm{pd}_\triangledown(\varphi)\\
&\mathrm{pd}_\triangledown(\exists_\triangleleft \varphi) = \mathrm{pd}_\triangledown(\exists_\triangleright \varphi) = \mathrm{pd}_\triangledown(\varphi)\\
&\mathrm{pd}_\triangledown(\exists_\triangledown \varphi) = \mathrm{pd}_\triangledown(\varphi) + 1
\end{aligned}$$

The *left-path-depth* $\mathrm{pd}_\triangleleft(\varphi)$ and the *right-path-depth* $\mathrm{pd}_\triangleright(\varphi)$ are defined in the obvious way, and the *path-depth* considers the nesting of all path quantifiers.

For example the formula

$$\exists_\triangledown\,.\, a\, U\, (\exists_\triangleleft\,.\, b\, U\, c).$$

has next/eventually depth 0, right-until-depth 0, down-until-depth 1, until-depth 1, and path depth 2.

Let $D = \{\mathrm{pd}_\triangleleft, \mathrm{pd}_\triangledown, \mathrm{pd}_\triangleright, \mathrm{ud}_\triangleleft, \mathrm{ud}_\triangledown, \mathrm{ud}_\triangleright, \mathrm{ned}\}$. For $d_1, \ldots, d_n \in D$ we denote by $\mathrm{CTL}^*_{\triangleleft\triangleright}(d_1 \le x_1, \ldots, d_n \le x_n)$ the set of $\mathrm{CTL}^*_{\triangleleft\triangleright}$ formulas φ with $d_1(\varphi) \le x_1, \ldots, d_n(\varphi) \le x_n$. In addition we use the shorthand notation $\mathrm{CTL}^*_{\triangleleft\triangleright}(d_1 \le x_1, \ldots, d_n \le x_n, \mathrm{other} \le o)$ to denote the set of $\mathrm{CTL}^*_{\triangleleft\triangleright}$ formulas φ with $d_1(\varphi) \le x_1, \ldots, d_n(\varphi) \le x_n$, and $d(\varphi) \le o$ for all $d \in D \setminus \{d_1, \ldots, d_n\}$. We denote by $\mathrm{CTL}^*_{\triangleleft\triangle\triangleright}(\mathrm{ud} \le 1)$ the set of $\mathrm{CTL}^*_{\triangleleft\triangle\triangleright}$ formulas with until-depth at most 1.

Let Σ be a label alphabet for ordered trees. We consider *first-order logic* over a signature having unary predicates $a(x)$ for every $a \in \Sigma$ as well as binary predicates for the parent/child relation, the immediate right-sibling relation, and the transitive closures of these predicates. The syntax and semantics of first-order logic is as usual [Libkin 2004]. The *quantifier depth* of a first order formula φ is – as usual – the maximal nesting depth of quantifiers in φ.

1.3. The Expressive Power of CTL*

It has been shown by Hafer and Thomas that CTL^* – $\mathrm{CTL}^*_{\triangleleft\triangleright}$ without horizontal path quantification – is equivalent to first order logic over unordered binary trees [Hafer and Thomas 1987]. Moller and Rabinovich show that CTL^* is equivalent to the bisimulation invariant fragment of first order logic over unordered trees [Moller and Rabinovich 1999]. Theorem 3.4 in [Barcelo 2005] extends this to ordered trees:

THEOREM 1.1 (BARCELO AND LIBKIN). *$CTL^*_{\triangleleft\triangleright}$ is first-order complete. That is, for every first-order sentence φ there is an $CTL^*_{\triangleleft\triangleright}$ sentence ψ such that for all ordered trees T, $T \models \varphi$ iff $T \models \psi$.*

As noted in the introduction, Theorem 3.4 of [Barcelo 2005] actually states a much stronger claim, which our Theorem 1.3 contradicts. We thus give a brief sketch of the proof of Theorem 1.1. One approach is via the composition technique of Moller and Rabinovich [Moller and Rabinovich 1999]. Indeed, this extension is implicit in the work of Moller and Rabinovich and that of Hafer and Thomas.

PROOF. Fix a finite alphabet Σ. It is known that for each k and v that there are only finitely many semantically distinct first-order formulas over Σ that have quantifier depth at most k and at most v free variables. Let Σ_k be the label set consisting of a label for each set of equivalent first-order formulas with one free variable that has quantifier depth at most k. For a node x in a Σ-labeled tree T let $\text{subtree-type}_k(x)$ be the set of first-order formulas of quantifier depth at most k true at x. For a path $\pi = x_1, \ldots, x_n$ we denote by $\text{subtree-type}_k(\pi)$ the path $\text{subtree-type}_k(x_1), \ldots, \text{subtree-type}_k(x_n)$.

Given a vertical path π in a tree T and a non-leaf node x on π we define the following: $\text{leftchildren}(\pi, x)$ is the leftward fullpath in T that starts at the unique child of x on π; $\text{rightchildren}(\pi, x)$ is defined analogously for right siblings. For all $k, l \in \mathbb{N}$, $\text{type}_{k,l}(x)$ is the pair consisting of the set of first-order sentences of quantifier depth at most l that hold of $\text{subtree-type}_k(\text{leftchildren}(\pi, x))$ and the set of first-order sentences of quantifier depth at most l that hold of $\text{subtree-type}_k(\text{rightchildren}(\pi, x))$. We let $\Sigma_{k,l}$ be the alphabet with a label for each pair of sets of sentences of quantifier depth at most l in the vocabulary for labelled strings over Σ_k, and an additional label $\perp$. $\text{type}_{k,l}(\pi)$ is the $\Sigma_{k,l}$-labelled string expanding π by labeling each interior node x with $\text{type}_{k,l}(x)$, and the leaf node of π with a special label $\perp$. Then we have:

> CLAIM 1. *For every first-order sentence ϕ, there are k and l and a first-order sentence ψ over $\Sigma_{k,l}$-labelled strings such that: $T, x \vDash \phi$ iff $\text{Type}_{k,l}(\pi) \vDash \psi$, where π is the path from the root of T to x.*

This is proved as in Theorem 3.2 of [Moller and Rabinovich 1999]. In the presence of an ordering the assumption of "wideness" used there is not needed. Hence the result then follows from:

> CLAIM 2. *For every first-order sentence ϕ over $\Sigma_{k,l}$-labelled strings, there is an $CTL^*_{\triangleleft\triangleright}$ formula ψ such that $\text{Type}_{k,l}(\pi) \vDash \phi$ iff $T, x \vDash \psi$*

This is proved using Kamp's theorem, as in Lemma 4.2 of [Moller and Rabinovich 1999]. □

$\text{CTL}^*_{\triangleleft\triangleright}$ is a large language, since it contains all of LTL as a sublanguage. Clearly, we can eliminate syntactic features of LTL that do not add expressiveness: for example, the eventually operator $F\varphi$ and the next operator $X\varphi$ can both be defined using our form of the until operator [Emerson 1990], and so both are unnecessary. What about the use of the until operator?

The following result follows directly from the work of Marx [Marx 2004; 2005a]:

THEOREM 1.2 (MARX). *$CTL^*_{\triangleleft\triangle\triangleright}(\text{ud} \leq 1)$ is first-order complete*

PROOF. The Conditional XPath language of [Marx 2004; 2005a] has been proved first-order complete in [Marx 2005a]. In [Libkin and Sirangelo 2008] it was shown that there is an easy translation from Conditional XPath filters to the language $\mathcal{X}_{\text{until}}$ defined in [Marx 2004]. $\mathcal{X}_{\text{until}}$ has quantification over partial (that is, not necessarily ending at a leaf) paths in the downward, upward, leftward, and rightward paths followed immediately by an until operator. This is analogous to the temporal logic CTL, which also restricts the sequencing of path quantifiers and LTL operators. Because the formula being checked after the quantification is an until, the partial path quantification can be replaced by quantification over fullpaths, and thus can be expressed in $\text{CTL}^*_{\triangleleft\triangle\triangleright}(\text{ud} = 1)$. $\mathcal{X}_{\text{until}}$ is a subset of $\text{CTL}^*_{\triangleleft\triangle\triangleright}(\text{ud} = 1)$, both are first-order complete. □

We thus have two ways of getting first-order completeness: Theorem 1.1 allows arbitrary LTL, downward paths and both horizontal paths, while Theorem 1.2 restricts the use of the Until operator but also allows upward paths. Can one make use of only

one horizontal operator? Can one restrict the nesting of Until operators without introducing upward axes? Our main results give negative answers to both these questions.

1.4. Main Results

We show two incompleteness results. First, $\mathrm{CTL}^*_{\triangleleft\triangleright}$ becomes incomplete if the number of $\exists_\triangleright$ quantifiers is restricted to some fixed p and the number of nested Us within $\exists_\triangleright$ quantifiers is restricted to some fixed u. Observe that despite this restriction, there still might be arbitrary LTL formulas on the leftward and downward paths, and arbitrarily many X and Fs on rightward paths.

THEOREM 1.3 (HORIZONTAL INCOMPLETENESS). *For all $p, u \in \mathbb{N}$, $CTL^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p, \mathrm{ud}_\triangleright \leq u)$ is not first-order complete on finite ordered trees. In addition, if $up > u'p'$ then $CTL^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p, \mathrm{ud}_\triangleright \leq u)$ is more expressive than $CTL^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p', \mathrm{ud}_\triangleright \leq u')$. The symmetric statements hold for leftward axes.*

It follows from our proof that $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangleleft = 0)$ is not first-order complete. This contradicts an addendum in the statement of Theorem 3.4 of [Barcelo 2005], stating that first-order queries can be rewritten without the use of backward operators in siblings – this side remark is not given a full proof in the paper. The main part of Theorem 3.4. of [Barcelo 2005] stating that unrestricted $\mathrm{CTL}^*_{\triangleleft\triangleright}$ is first-order complete, is not contradicted by our results. **Michael: For now, I have removed the statement: "However, it follows from [Barcelo 2005] that $\mathbf{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangleleft \leq 1)$ is already first order complete." since I do not see how it follows.**

Our second incompleteness result is concerned with downward axes. It states $\mathrm{CTL}^*_{\triangleleft\triangleright}$ is not first order complete if the nesting depth of U on downward paths is restricted to u – even with an arbitrary nesting depth of all other modifiers.

THEOREM 1.4 (VERTICAL INCOMPLETENESS). *On finite, ordered trees, $CTL^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq u)$ is not first-order complete for all $u \in \mathbb{N}$. In addition, the family $CTL^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq u)$ forms a strict hierarchy in expressiveness.*

We will show that this incompleteness result holds even over binary trees.

Discussion. It is known from [Etessami and Wilke 2000] that on strings, the subsets of LTL formed by restricting the until-depth to some fixed number form a strict hierarchy in expressiveness. This does not imply the corresponding result for a language with path quantification on ordered trees, since path quantification coupled with LTL operators could add more expressiveness. For example, when one restricts to trees that consist of exactly one path, $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} = 1)$ is first-order complete: arbitrary nesting of untils can be mimicked by putting each until in a path quantifier, since the until-depth only counts nesting within a given path quantification. Consider also the analogous situation when LTL is extended with "past operators" S and P, which are the duals of U and F (see [Emerson 1990] for the precise definition): Etessami and Wilke [Etessami and Wilke 2000] have shown that the subsets of LTL extended with past operators formed by restricting the nesting of both until and its dual S form a strict hierarchy. But the theorem of Marx mentioned above implies that $\mathrm{CTL}^*_{\triangleleft\triangleright}$ based on LTL-with-past but restricted to since/until-depth one is already sufficient for first-order completeness; this is because upward path quantification can be replaced by past operators within a path, using the fact that a node has a unique ancestor path.

Thus the incompleteness of query languages based on fixed until-depth when only future operators are available is not obvious. The proofs of Theorems 1.3 are quite involved and 1.4 are will take up the remainder of the paper.

2. BACKGROUND: GAMES FOR WORDS AND TREES

We now describe a game on finite labelled trees, that corresponds to $\mathrm{CTL}^*_{\triangleleft\triangleright}$. In the same way as LTL is embedded into $\mathrm{CTL}^*_{\triangleleft\triangleright}$, there is a game on words that is embedded into the game that corresponds to $\mathrm{CTL}^*_{\triangleleft\triangleright}$. This game was introduced by Etessami and Wilke in [Etessami and Wilke 2000] where they showed that it corresponds to LTL. Before we define the game on trees, we define this game on words.

2.1. Ehrenfeucht-Fraïssé Games on Words

The LTL$(\mathrm{ned} \leq e, \mathrm{ud} \leq u)$-game (or LTL$(e,u)$-game for short) is played by two players, a male *Spoiler* and a female *Duplicator*, on two words w, v. The goal of Spoiler is to show that the words are different while Duplicator tries to show that they are similar. At each stage of the game a pair w^i, v^j of suffixes of w, v is *selected*. There are different kinds of moves – X-, F-, and U-moves – in which the players alter the selected words.

We describe the play of the LTL(e,u)-game with selected words w, v. If $e = 0$ and $u = 0$, Duplicator wins the game if the roots of the selected words have same label and Spoiler wins otherwise. In this case we call w, v the *final position* of the game.

If $e > 0$ Spoiler can chooses to play either an X-, or an F-move.

> *X-move.* In an *X-move* neither player has any choice: the new selected words are w^2, v^2. Spoiler can only choose this move if one of the words has such a suffix. If one of w, v has such a suffix and the other does not, Duplicator cannot move and Spoiler wins the game. Spoiler also wins the game if the roots of the new selected words have different labels.
>
> *F-move.* In an *F-move* Spoiler picks one of the two words, say w. He then selects a proper suffix $w[i*]$ of w with $i > 1$. Again, Spoiler can only choose this move if such a suffix exists. Duplicator selects a suffix $v[j,*]$ of v with $j > 1$. Duplicator loses the game if she cannot respond with a suffix such that the roots of $w[i,*]$ and $v[j,*]$ have the same label.

If Duplicator did not lose the game in the preceding round, the players play the LTL$(e-1,u)$-game on the newly selected words. The winner of this subgame will be declared winner of the LTL(e,u)-game with w, v.

If $u > 0$ then Spoiler can also consider to play an U-move.

> *U-move.* In an *U-move* Spoiler picks one of the words and we assume he picks w. An U-move consists of two *half moves*. In the first half move Spoiler selects a proper suffix $w[i,*]i$ of w for $i > 1$. The Duplicator has to reply with a proper suffix $v[j,*]$ of v for some $j > 1$ such that the roots of $w[i,*]$ and $v[j,*]$ have the same label. Again, Spoiler can only choose this move if $w[i,*]$ exists and Duplicator looses the game if the roots of $w[i,*]$ and $v[j,*]$ have different labels.
> In the second half move Spoiler selects $v[j',*]$ for some $1 < j' \leq j$. Duplicator has to select $w[i',*]$ such that $1 < i' \leq i$ and such that the roots of $w[i',*]$ and $v[j',*]$ have the same label. In the case that the Spoiler picks $v[j',*]$ such that $j' = j$, Duplicator must choose i' to be equal to i. Again, Duplicator looses the game if she cannot select such a word.

If Duplicator survives both half rounds, the players continue playing the LTL$(e,u-1)$-game with $w[i',*], v[j',*]$. The winner of this game is the winner of the LTL(e,u)-game on w, v.

Etessami and Wilke have shown the following [Etessami and Wilke 2000]:

PROPOSITION 2.1 (ETESSAMI, WILKE). *Duplicator has a winning strategy in the $LTL(e,u)$-game on words w,v iff w,v satisfy the same LTL$(\mathrm{ned} \le e, \mathrm{ud} \le u)$ formulas.*

2.2. Ehrenfeucht-Fraïssé Games on Trees

For given $p_\triangleleft, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e \in \mathbb{N}$ we will abbreviate

$$\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \le p_\triangleleft, \mathrm{pd}_\triangledown \le p_\triangledown, \mathrm{pd}_\triangleright \le p_\triangleright, \mathrm{ud}_\triangleleft \le u_\triangleleft, \mathrm{ud}_\triangledown \le u_\triangledown, \mathrm{ud}_\triangleright \le u_\triangleright, \mathrm{ned} \le e)$$

by $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$ below. The $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$-game is played by Spoiler and Duplicator on a pair of trees T, S. Again there are different kinds of moves – $\exists_\triangleleft$-, $\exists_\triangleright$-, and $\exists_\triangledown$-moves – in which the players alter a pair of *selected nodes* x, y. Initially, the roots of T, S are selected. We describe the play of the game with x, y selected. Duplicator wins the $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$-game on T, S if $p_\triangleleft + p_\triangledown + p_\triangleright = 0$ and if x and y have the same label. Otherwise Spoiler can choose one of the following moves:

> $\exists_\triangleleft$*-move.* Spoiler can choose an $\exists_\triangleleft$*-move* if $p_\triangleleft > 0$. He picks one of the two trees T, S, say T. Spoiler then picks the leftward fullpath π that is rooted at x. Duplicator responds by picking the leftward fullpath τ that is rooted at y. Then Spoiler and Duplicator play the LTL$(e, u_\triangleleft)$-game on π, τ, where these paths are considered as words. If Spoiler wins this word-game then he wins the tree-game. In addition, at any point in the play of the LTL$(e, u_\triangleleft)$-game on on π, τ, Spoiler can choose to spawn a new game. Let x', y' be the play of this game (below we will call these "intermediate positions" of the LTL-game). Then the players play the $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft - 1, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$-game on T, S with x', y' selected. If Spoiler wins any of these subgames, he wins, and otherwise Duplicator wins.
>
> $\exists_\triangledown$*-move.* An $\exists_\triangledown$*-move* can be chosen by Spoiler if $p_\triangledown > 0$. It is played like an $\exists_\triangleleft$-move, just that the players pick some downward fullpaths π, τ instead of the leftward fullpaths and the players play the LTL$(e, u_\triangledown)$-game on π, τ instead of the LTL$(e, u_\triangleleft)$-game. Again, Spoiler wins the tree game if he wins the game in π, τ. In addition, Spoiler can choose any intermediate position of the path game, and wins if he can win the $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown - 1, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$-game from these positions.
>
> $\exists_\triangleright$*-move.* Spoiler may play an $\exists_\triangleright$*-move* if $p_\triangleright > 0$. The rules are as above, just with rightward paths on which the players play the LTL$(e, u_\triangleright)$-game. Again, Spoiler can win the tree game by winning the game on π, τ, or by winning the $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown, p_\triangleright - 1, u_\triangleleft, u_\triangledown, u_\triangleright, e)$-game on an intermediate position of the word game.

A *winning strategy* for either player from a given initial position and set of moves is defined as usual. The following lemma shows the correspondence between the tree game and the logic $\mathrm{CTL}^*_{\triangleleft\triangleright}$.

PROPOSITION 2.2. *Let T, S be finite trees. Duplicator has a winning strategy for the CTL$^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$-game on T, S iff the trees T and S agree on all CTL$^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$ node formulas.*

PROOF. (sketch) We show only the 'if' direction, the other direction is similarly straightforward. Given the hypothesis, we show that Duplicator's winning strategy has the property that if the position after a move is x, y at a stage with $(p'_\triangleleft, p'_\triangledown, p'_\triangleright)$ moves to play, then (T, x) agrees with (S, y) on $\mathrm{CTL}^*_{\triangleleft\triangleright}(p'_\triangleleft, p'_\triangledown, p'_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$ node formulas. We show this by induction on $p'_\triangleleft + p'_\triangledown + p'_\triangleright$. The base case of no moves remaining is clear.

One inductive case is when Spoiler plays an $\exists_\triangledown$-move π By induction, (T, x) satisfies the same formulas of $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown,\ p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$ as (S, y), and hence there is a path τ rooted at y such that (T, π) and (S, τ) satisfy the same path formulas of $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown - 1, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$. Duplicator responds with such a path. Consider LTL over the alphabet with propositions corresponding to $\mathrm{CTL}^*_{\triangleleft\triangleright}(p_\triangleleft, p_\triangledown - 1, p_\triangleright, u_\triangleleft, u_\triangledown, u_\triangleright, e)$ node formulas, and consider the expansion of π and τ where nodes are decorated by the formulas in the above language that they satisfy. The hypothesis on π and τ and Proposition 2.1 guarantee that there is a winning strategy for Duplicator in the $LTL(e, u_\triangledown)$-game over this alphabet. Duplicator uses this strategy in the remainder of the move.

The case of leftward and rightward paths is done similarly. □

2.3. The Until Hierarchy on Words

We will now review a winning strategy of Duplicator for a game on words. This strategy has been used by Etessami and Wilke to show that $\mathrm{LTL}(\mathrm{ud} \le u)$ is not first-order complete over finite words for all $u \in \mathbb{N}$. As we will often reuse parts of their proof in our proofs of Theorems 1.3 and 1.4, we reprove the result of Etessami and Wilke here.

THEOREM 2.3 ([ETESSAMI AND WILKE 2000]). *$\mathrm{LTL}(\mathrm{ud} \le u)$ is not first-order complete over finite words for every $u \ge 0$. In addition, for each u, $\mathrm{LTL}(\mathrm{ud} \le u + 1)$ is more expressive than $\mathrm{LTL}(\mathrm{ud} \le u)$.*

The Separating Property. To show Theorem 2.3, we define for each $u \in \mathbb{N}$ a property S_u that can be expressed in $\mathrm{LTL}(\mathrm{ud} \le u)$ but not in $\mathrm{LTL}(\mathrm{ud} < u)$ for all $u \in \mathbb{N}$.

Definition 2.4. Let S_u denote the set of words that contain a prefix satisfying the regular expression

$$a(c^* a)^u.$$

The property S_u can be encoded by the $\mathrm{LTL}(\mathrm{ud} = u)$ formula ψ_u defined recursively as follows:

$$\psi_u = \begin{cases} a & \text{if } u = 0 \\ a \wedge (c\, U\, \psi_{u-1}) & \text{if } u > 0 \end{cases}$$

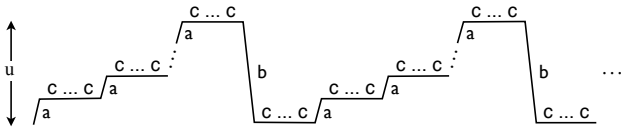
To complete the proof of Theorem 2.3, we now show that S_u cannot be expressed in $\mathrm{LTL}(\mathrm{ud} < u)$, even for arbitrary nesting depth of X and F modifiers. This follows from Lemma 2.5 below together with Proposition 2.1. We first show the lemma for infinite words, because it simplifies the presentation. The case for finite words will be a simple consequence (Corollary 2.6).

LEMMA 2.5 (ETESSAMI, WILKE). *For all $u, e > 0$ there are two infinite words $\tilde{v}$ and $\tilde{w}$ such that (i) $\tilde{v}$ satisfies S_{u+1} but $\tilde{w}$ does not, and (ii) Duplicator has a winning strategy for the $\mathrm{LTL}(\mathrm{ud} \le u, \mathrm{ned} \le e)$-game on $\tilde{v}, \tilde{w}$.*

We show the lemma in the rest of this section. We fix some given $\tilde{u}, \tilde{e} \in \mathbb{N}$ and define the infinite words

$$\tilde{w} = a\big((c^{\tilde{e}} a)^{\tilde{u}} c^{\tilde{e}} b\big)^\omega$$
$$\tilde{v} = a c^{\tilde{e}}\, \tilde{w}$$

The word $\tilde{w}$ can be visualized as the following "staircase".



It is obvious that $\tilde{v}$ satisfies S_{u+1} but $\tilde{w}$ does not. Hence part (i) of Lemma 2.5 holds. To prove part (ii) we show that Duplicator has a winning strategy for the LTL$(\mathrm{ud} \le u, \mathrm{ned} \le e)$-game on $\tilde{v}$ and $\tilde{w}$. The idea of the proof will be to show that Spoiler can force the selected positions up only one step of the staircase on each U-move. As the number of steps depends on the number of U-moves, Spoiler cannot detect that the selected positions are on different steps of the staircase in the beginning of the game.

Duplicator's Strategy. We first need some notations. Given two positions x, y of the same word we denote by (x, y) the sequence of positions between x and y, excluding x and y. We denote by $\mathrm{right}_a(x)$ the next position to the right of x that is labelled a. $\mathrm{right}_{ab}(x)$ is the next a or b labelled position to the right of x. Intuitively, the *plateau-depth* of a position x in $\tilde{w}$ or $\tilde{v}$ is the distance of x to the end of the "plateau" to the right. Formally, $\text{plateau-depth}(x)$ is the number of c-labelled positions in $(x, \mathrm{right}_{ab}(x))$. The *top-depth* of a position x in $\tilde{w}$ or $\tilde{v}$ is the number of steps between x and the top of the staircase, that is $\text{top-depth}(x)$ is the number of a positions in $(x, \mathrm{right}_b(x))$. The top-depth of a substring of $\tilde{w}$ or $\tilde{v}$ is the top-depth of its root, considered within $\tilde{w}$ (resp. $\tilde{v}$), and similarly for plateau-depth. For example the top-depth of $\tilde{w}$ is u and its plateau-depth is e. Note also that

$$\text{plateau-depth}(w) = n \qquad \text{if } w \in \Sigma \cdot c^n \cdot \Sigma \setminus \{c\} \cdot \Sigma^\omega$$
$$\text{top-depth}(w) = n \qquad \text{if } w \in \Sigma \cdot (c^* a)^n c^* b \cdot \Sigma^\omega$$

The following claim implies part (ii) of Lemma 2.5 because $\tilde{w}, \tilde{v}$ satisfy the conditions of Claim 1 if the whole paths are selected.

CLAIM 1. *Let $e \le \tilde{e}$ and $u \le \tilde{u}$. Duplicator can win the LTL$(\mathrm{ud} \le u, \mathrm{ned} \le e)$-game on suffixes w, v of the words $\tilde{w}, \tilde{v}$, if*
(1) the roots of w and v have the same label.
(2) w and v have the same plateau-depth.
(3) $|\text{top-depth}(w) - \text{top-depth}(v)| \le 1$.
(4) If w and v have different top-depths then
(a) $\text{top-depth}(w) \ge u$ and $\text{top-depth}(v) \ge u$ and
(b) if $\text{top-depth}(w) = u$ or $\text{top-depth}(v) = u$
then $\text{plateau-depth}(w) \ge e$ (and hence $\text{plateau-depth}(v) \ge e$).

Proof of Claim 1. The proof is by induction on $u + e$. The base case $u + e = 0$ holds because by Condition 1, the roots of v, w have the same label. In the induction step, we fix $e + u > 0$, assume that the claim holds for every pair e', u' with $e' + u' < u + e$, and look to show the result for e, u. We thus fix w and v satisfying the conditions above for e and u, and give a strategy for Duplicator to win the LTL$(\mathrm{ud} \le u, \mathrm{ned} \le e)$ game. We distinguish three cases, according to the kind of move that Spoiler chooses first.

X-moves. Spoiler can choose an X-move only if $e > 0$. In this case Duplicator's strategy is determined by the rules of the game: the selected suffixes after the round are $v[2, *], w[2, *]$.
We prove that Duplicator can win the LTL$(\mathrm{ud} \le u, \mathrm{ned} \le e - 1)$-game on $v[2, *], w[2, *]$. This is done by showing that the conditions of Claim 1 hold for u and $e - 1$. This implies that the roots of the newly-selected suffixes have

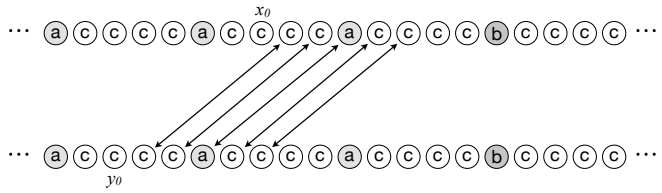


Fig. 1. Duplicator's strategy if the current position of the game is (x_0, y_0) and Spoiler skips at most $\tilde{e}+1$ positions. If Spoiler picks a position z in either w or v in his first half move, then Duplicator picks the position z' in the other word, such that there is a arrow from z to z'.

the same label, and hence the claim follows by induction. The only interesting case is when $|\text{top-depth}(w) - \text{top-depth}(v)| = 1$ and $\text{top-depth}(w) = u$. WLOG, we can assume $\text{top-depth}(v) = u+1$. In this case $\text{plateau-depth}(w) \geq e$ by Condition 4b. As by definition we have that $\text{plateau-depth}(w[2,*]) \geq \text{plateau-depth}(w) - 1$ it follows that the plateau-depth of $w[2,*]$ is greater or equal to $e-1$. A similar argument shows that $\text{plateau-depth}(v[2,*]) \geq e-1$. As $\text{plateau-depth}(w) \geq e > 0$ it also follows that the top-depth of neither word has decreased, that is $\text{top-depth}(w[2,*]) = \text{top-depth}(w) = u$ and $\text{top-depth}(w[2,*]) = \text{top-depth}(w[2,*]) = u$. Hence Condition 4 holds for $w[2,*], v[2,*]$. Condition 2 is obvious.

F-moves. The strategy on F-moves is very easy for Duplicator. It relies on the observation that as v and w are infinite and satisfy the invariant, they have the same set of proper suffixes. Therefore, given Spoiler's selection, Duplicator can select the same suffix in the other word.

U-moves. On U-moves, Duplicator cannot use the same strategy as for F-moves in her first half move: if she did, Spoiler might play on the word with the smaller top-depth and just move the selected suffix by one position. Then Duplicator might skip $\tilde{e}+1$ positions to the next isomorphic suffix. In the second half move, Spoiler can pick from $\tilde{e}+2$ positions (the positions that Duplicator skipped and the one she selected), but Duplicator can only choose the position that Spoiler selected. Hence, when Spoiler skips only few positions, Duplicator will skip the same number of positions. Only when Spoiler skips sufficiently many positions will Duplicator try to find an isomorphic suffix.
We now assume that Spoiler chooses to play on w and that he selects a suffix $w[n,*]$ of w in his first half move. If Spoiler skips at most $\tilde{e}+1$ positions (that is $n \leq \tilde{e}+1$) then Duplicator skips the same number of positions as Spoiler did, that is, she picks $v[n,*]$. Otherwise Duplicator picks the largest proper suffix $v[m,*]$ of v that is isomorphic to $w[n,*]$ (such a suffix exists due to Condition 4). The case where Spoiler plays on v is symmetric. See Figures 1 and 2 for a visualization of Duplicator's strategy.
We verify that the suffixes w', v' chosen after the first half move satisfy the conditions of Claim 1 for $e, u-1$. This implies that their roots have the same label and thus the claim follows by induction. If Duplicator has chosen a suffix that is isomorphic to Spoiler's choice, then it is clear that the conditions of Claim 1 hold. In the other case both positions have been advanced by $n \leq \tilde{e}+1$ positions. The interesting case is when $|\text{top-depth}(w) - \text{top-depth}(v)| = 1$ and $\text{top-depth}(w) = u$ (and hence $\text{top-depth}(v) = u+1$). Clearly $\text{plateau-depth}(w[n,*]) = \text{plateau-depth}(v[n,*])$ and hence Condition 2 holds for $w[n,*], v[n,*]$. To verify Condition 4 first note that $|\text{top-depth}(w[n,*]) - \text{top-depth}(v[n,*])| = 1$. There are two cases:

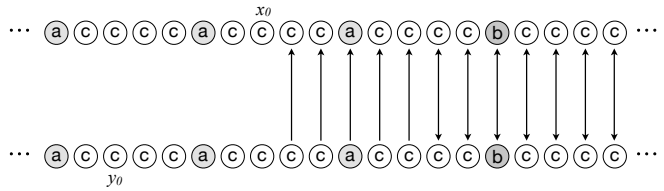


Fig. 2. Duplicator's strategy if Spoiler plays on the lower word (starting from position y_0) and skips more than $\tilde{e}+1$ positions. The meanings of the arrows are as in Figure 1.

— If the positions have not been advanced beyond the next b (formally $n \leq \text{plateau-depth}(w)+1$) then $\text{top-depth}(w[n,*]) = \text{top-depth}(w) = u > u-1$ and similarly $\text{top-depth}(v') > u-1$.
— In the other case $\text{top-depth}(w[n,*]) = u-1$ and $\text{top-depth}(v[n,*]) = u-1$, but as $n \leq e+1$, the plateau-depth of both suffixes cannot have increased. Hence $\text{plateau-depth}(w[n,*]) \geq \text{plateau-depth}(w) > e$ and similarly $\text{plateau-depth}(v[n,*]) > e$.

Assume that the suffixes $w[n,*], v[m,*]$ have been selected after the first round. In the second half move Spoiler chooses a suffix $v[m',*]$ such that $1 < m' \leq m$. It is easy to check that Duplicator can always choose $w[n',*]$ such that $n-n' = m-m'$. **Michael: I do not see this: if w had smaller top-depth then it is clear, but in the other case?** Observe that if $w[n',*]$ and $v[m',*]$ are not isomorphic, then $n', m' \leq e+1$. Hence the conditions of Claim 1 are maintained. □

How must the above construction be altered to show the result for finite words? Note that Duplicator only exploits that the words are infinite on her strategy for F-moves. Hence, if she only jumps to the *next* $(ac^e)^u bc^e$ section on each F-move, then she can win the game provided there are at least $e+1$ such sections. Thus we have:

COROLLARY 2.6 (ETESSAMI, WILKE). *For all $u, e \in \mathbb{N}$, Duplicator has a winning strategy for the LTL$(\text{ud} \leq u, \text{ned} \leq e)$-game on the finite words*

$$\tilde{w}_{\mathit{fin}} := \left((ac^e)^u bc^e\right)^{e+1}$$
$$\tilde{v}_{\mathit{fin}} := ac^e\, \tilde{w}_{\mathit{fin}}.$$

We will reuse Duplicator's winning strategy that we described in the proof from Claim 1 in Lemma 2.5 in the proofs of Theorems 1.3 and 1.4. The game on trees will be designed in such a way, that in at some point of this game, the players will select paths that are isomorphic to the words $\tilde{w}_{\text{fin}}$ and $\tilde{v}_{\text{fin}}$. Hence we will be able to use Duplicator's strategy of Claim 1 in Lemma 2.5 to show that Duplicator can win the word-game on these paths.

3. THE HORIZONTAL UNTIL HIERARCHY

We now turn to the proof of Theorem 1.3, that is our horizontal incompleteness result on trees. We start by defining a property Q_i of trees for all $i \in \mathbb{N}$. Later we will show that $\text{CTL}^*_{\triangleleft\triangleright}(\text{pd}_\triangleright \leq p_\triangleright, \text{ud}_\triangleright \leq u_\triangleright)$ can define $Q_{p_\triangleright u_\triangleright}$ but not $Q_{p_\triangleright u_\triangleright+1}$ for all $p_\triangleright, u_\triangleright \in \mathbb{N}$.

3.1. The Separating Property

The *right path* of a node x in a tree is the sequence of its right siblings.

Definition 3.1 (*Separating Property*). Let Q_i be the set of ordered trees that have a rooted downward fullpath π ending at a leaf labelled d, such that each node on π apart

from the root and the leaf has a right path with a prefix satisfying

$$a(c^*a)^i.$$

If $p_\triangleright \cdot u_\triangleright \geq i$, then Q_i can be expressed in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangleright \leq u_\triangleright)$ by the sentence 'there is a rooted downward fullpath ending on d on which every node satisfies μ_i' where μ_i is the formula 'there is a rightward path satisfying $a(c^*a)^i\,\Sigma^*$'. Formally

$$\mu_i = \begin{cases} a & \text{if } i = 0 \\ \exists_\triangleright(\lambda_{u_\triangleright,i}) & \text{if } i > 0 \end{cases}$$

$$\lambda_{j,i} = \begin{cases} \mu_i & \text{if } j = 0 \text{ or } i = 0 \\ a \wedge (c\,U\,\lambda_{j-1,i-1}) & \text{if } j > 0 \text{ and } i > 0 \end{cases}$$

The following lemma shows that $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangleright \leq u_\triangleright)$ cannot express Q_{i+1} if $p_\triangleright u_\triangleright < i$. Thus it implies Theorem 1.3.

LEMMA 3.2. *For all $p_\triangleright, u_\triangleright, o \in \mathbb{N}$, there are finite trees $\tilde{T}$ and $\tilde{S}$ such that (a) $\tilde{T}$ satisfies $Q_{p_\triangleright u_\triangleright+1}$ but $\tilde{S}$ does not and (b) Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangleright \leq u_\triangleright, \mathrm{other} \leq o)$-game on $\tilde{T}, \tilde{S}$.*

Before we prove Lemma 3.2, we note a consequence of it: Consider the case where $u_\triangleright = 0$. Then Lemma 3.2 states that there are finite trees $\tilde{T}, \tilde{S}$ such that $\tilde{T}$ satisfies Q_1 but $\tilde{S}$ does not and Duplicator wins the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangleright = 0; \mathrm{other} \leq o)$-game on $\tilde{T}, \tilde{S}$ for all $p_\triangleright, o \in \mathbb{N}$. It follows that Q_1 cannot be expressed in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangleright = 0; \mathrm{other} \leq o)$ for any $p_\triangleright, o \in \mathbb{N}$. Hence we get the following corollary, which implies that $\mathrm{ForCXPath}$ is not first-order complete.

COROLLARY 3.3. *$\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangleright = 0)$ is not first-order complete on finite ordered trees.*

We now start with the proof of Lemma 3.2. Recall that this lemma is all we need to complete the proof of Theorem 1.3.

PROOF OF LEMMA 3.2. The proof is quite long. We will define two trees $\tilde{T}, \tilde{S}$ and then we show that Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangleright \leq u_\triangleright; \mathrm{other} \leq o)$-game on $\tilde{T}, \tilde{S}$. We will first show the lemma for "wide trees" – those in which a node can have infinitely many left and right siblings, but where any two siblings have only finitely many nodes between them (i.e. the sibling order has type $\omega^* + \omega$). Later we explain how the proof must be altered for the finite case.

3.2. The Trees

To construct $\tilde{T}$ and $\tilde{S}$, we need some way to inductively define trees. We will use the concept of templates, defined in the following. A *hedge* is an ordered sequence of wide trees, where the sequence can again be infinite in both directions. A *template* F is a hedge with two sets of distinguished leaves – *positive ports* and *negative ports* – labelled by "+" and "–" respectively. A template F and two hedges $\bar{T}, \bar{S}$ can be combined to form a new hedge $F[\bar{T}, \bar{S}]$ which is obtained from F by replacing each positive port with the hedge $\bar{T}$ and each negative port with the hedge $\bar{S}$. In Figure 3 we see an example of a template F and two hedges S and T and the result of applying F to S and T.

We now fix some given $\tilde{p}_\triangleright, \tilde{u}_\triangleright, \tilde{o} \in \mathbb{N}$ for the rest of the proof of Lemma 3.2. We first define for each $k \in \mathbb{N}$ two hedges $\bar{T}^k$ and $\bar{S}^k$. These hedges will be constructed from two templates F and G which we define first. Both F and G are hedges of infinite width.

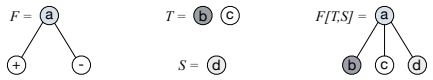


Fig. 3. Constructing the hedge $F[T, S]$ from the template F and the hedges T and S.

The sequence of roots of both F and G spell out the infinite word

$$\left((c^n a)^m\ c^n b\right)^\omega$$

where $n = \tilde{p}_\triangleright \cdot \tilde{o} + \tilde{o}^2$ and $m = \tilde{p}_\triangleright \cdot \tilde{u}_\triangleright + \tilde{o}^2 + 2$. All subtrees in F or G that have a root labelled c are singleton trees, consisting only of a root. The subtrees that have a root labelled a or b consist of a root with a single child that is either a positive or a negative port. In both templates all ports but one are negative. What distinguishes F from G is the labeling of the rightward path that starts at the unique parent of the positive port: In F this path has a label in the language $a(c^*a)^{\tilde{p}_\triangleright \tilde{u}_\triangleright + 1}\, c^* b\, \Sigma^\omega$; in G its label lies in the language $a(c^*a)^{\tilde{p}_\triangleright \tilde{u}_\triangleright} c^* b\, \Sigma^\omega$. Figure 4 shows F and G.

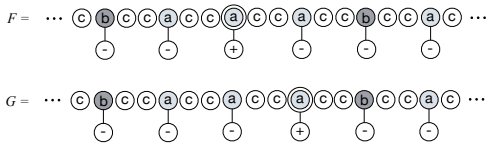


Fig. 4. The templates F and G for $\tilde{p}_\triangleright = 1$, $\tilde{o} = 1$, and $\tilde{u}_\triangleright = 0$.

We define two sequences $(\bar{T}_k)_{k\in\mathbb{N}}$ and $(\bar{S}_k)_{k\in\mathbb{N}}$ of hedges by induction on k: $\bar{T}_0$ is the single node labelled d and $\bar{S}_0$ is the single node labelled c. For $k > 0$

$$\bar{T}_k := F\left[\bar{T}_{k-1}, \bar{S}_{k-1}\right]$$
$$\bar{S}_k := G\left[\bar{T}_{k-1}, \bar{S}_{k-1}\right]$$

We say that the single node T_0 is *positive* and the single node S_0 is *negative*. For a tree of the form $F[\bar{T}, \bar{S}]$ which is obtained from F by replacing each positive port with the hedge $\bar{T}$ and each negative port with the hedge $\bar{S}$, each copy of the root of $\bar{T}$ under this replacement is said to be *positive*, and each copy of the root of $\bar{S}$ is said to be *negative*. That is, the root nodes of the substituted trees inherit the polarity of the ports in which they were substituted. The polarity of a subtree is the polarity of its root. Figure 5 shows a positive and a negative tree in $\bar{T}_3$.

Now we can define $\tilde{T}$ and $\tilde{S}$, the trees whose existence is asserted in Lemma 3.2. These are also the trees on which Spoiler and Duplicator will play the $\mathrm{CTL}^*_{\triangleleft\triangleright}$-game. Recall that we fixed $\tilde{o} \in \mathbb{N}$ at the beginning of this proof.

Definition 3.4 ($\tilde{T}$, $\tilde{S}$). Let $\tilde{T}$ be the unique positive tree in the hedge $\bar{T}_{\tilde{o}+1}$. $\tilde{S}$ is some negative tree in $\bar{S}_{\tilde{o}+1}$ whose root is labelled a (note that all trees with this property are isomorphic).

It is easy to see that $\tilde{T}$ satisfies $Q_{\tilde{p}_\triangleright \tilde{u}_\triangleright + 1}$ while $\tilde{S}$ does not: An inductive argument shows that a subtree of $\bar{T}_{\tilde{o}+1}$ or $\bar{S}_{\tilde{o}+1}$ satisfies $Q_{\tilde{p}_\triangleright \tilde{u}_\triangleright + 1}$ iff it is positive. Hence part (a) of Lemma 3.2 follows.

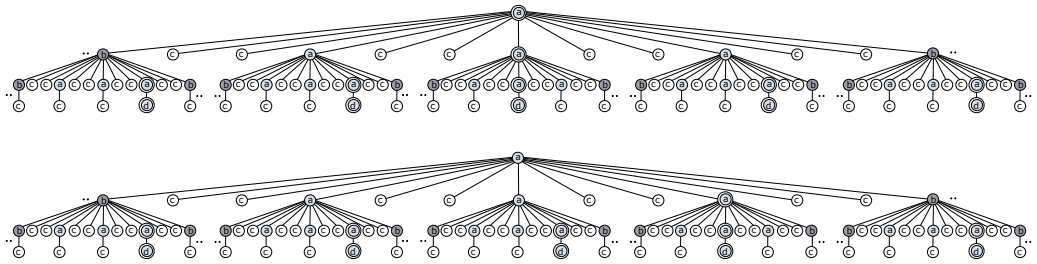


Fig. 5. A positive tree in (top) and a negative tree (bottom) in $\bar{T}_3$ for $\tilde{p}_\triangleright = 1, \tilde{u}_\triangleright = 0$ and $\tilde{o} = 0$. Roots of positive subtrees are displayed with double circles, roots of negative subtrees with single ones. In particular, the top tree satisfies Q_1, while the bottom tree does not.

3.3. Duplicator's Strategy

We now show part (b) of Lemma 3.2 for $\tilde{T}$ and $\tilde{S}$ as defined above. In fact, we show the following more general statement on hedges:

> If x is the root of a positive tree in $\bar{T}_{\tilde{o}+1}$ and y is the root of a negative tree in $\bar{S}_{\tilde{o}+1}$, then Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleright \le \tilde{p}_\triangleright, \mathrm{ud}_\triangleright \le \tilde{u}_\triangleright; \mathrm{other} \le \tilde{o})$-game on the hedges $\bar{T}_{\tilde{o}+1}$ and $\bar{S}_{\tilde{o}+1}$ starting from position x, y.

We use an extension of $\mathrm{CTL}^*_{\triangleleft\triangleright}$ for hedges: The node formula $\exists_\triangleright \phi$ is true at the root x of a tree in a hedge $\bar{T}$ if ϕ is true on the sequence of roots of $\bar{T}$ to the right of x. In a similar way, the $\mathrm{CTL}^*_{\triangleleft\triangleright}$-game can be extended to hedges, such that the players can choose the sequence of roots right of the current selection in $\exists_\triangleright$ moves. The symmetric definitions hold for leftward paths.

Notation. During the game on $\bar{T}_{\tilde{o}+1}$ and $\bar{S}_{\tilde{o}+1}$, Duplicator can maintain an invariant on the string of siblings of the selected nodes x and y. This invariant is similar to the invariant used in the word game from Section 2.3, but this time, Duplicator not only has to assure that the strings to the right of the selected nodes are similar, but also those to the left of them. We therefore define the inverse versions of top-depth and plateau-depth. The *inverse-plateau-depth* of a node x is the number of c-labelled nodes between x and the next node labelled a or b to the left of x. The *bottom-depth* of x is the number of a-labelled nodes between x and the next b to the left of x. The *plateau-depth* of a node x in $\bar{T}_k$ or $\bar{S}_k$ is the plateau-depth of x with respect to its sequence of right siblings – or on the sequence of roots of $\bar{T}_k$ or $\bar{S}_k$, if x is a root. We lift the notions of inverse-plateau-depth, the top-depth, and the bottom-depth to trees in the same way. In each sequence of siblings in either $\bar{T}_k$ or $\bar{S}_k$ we distinguish the node that is the next b-labelled node to the right of the only positive node. We will say that a node x in $\bar{S}_k$ *corresponds* to a node y in $\bar{T}_k$ (or vice versa) if x and y have the same distance and direction to the distinguished node in their respective sequence of siblings.

The following claim is sufficient for part (b) of Lemma 3.2. As we have already proven part (a) of Lemma 3.2, the proof of the following claim completes the proof of Theorem 1.3.

> CLAIM 1. *Let $p_\triangleright \le \tilde{p}_\triangleright$, and $p_\triangleleft, p_\triangledown \le \tilde{o}$. Duplicator can win the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \le p_\triangleleft, \mathrm{pd}_\triangledown \le p_\triangledown, \mathrm{pd}_\triangleright \le p_\triangleright, \mathrm{ud}_\triangleright \le \tilde{u}_\triangleright; \mathrm{other} \le \tilde{o})$-game on $\bar{S}_{\tilde{o}+1}, \bar{T}_{\tilde{o}+1}$ if the selected positions x, y are roots of trees in $\bar{S}_{\tilde{o}+1}, \bar{T}_{\tilde{o}+1}$ and satisfy:*
> (*1*) *x and y have the same label.*
> (*2*) *x and y have the same plateau-depth.*
> (*3*) $|\text{top-depth}(x) - \text{top-depth}(y)| \le 1$.
> (*4*) *If x and y have different top-depths then*

(a) $\text{top-depth}(x) \geq p_\triangleright \tilde{u}_\triangleright$ ***and*** $\text{top-depth}(y) \geq p_\triangleright \tilde{u}_\triangleright$
(b) ***if*** $\text{top-depth}(x) = p_\triangleright \tilde{u}_\triangleright$ ***or*** $\text{top-depth}(y) = p_\triangleright \tilde{u}_\triangleright$
then $\text{plateau-depth}(x) \geq p_\triangleright \tilde{o}$
(and hence $\text{plateau-depth}(y) \geq p_\triangleright \tilde{o}$***).***

(5) *If x and y have different bottom-depths then*
(a) $\text{bottom-depth}(x) \geq p_\triangleleft \tilde{o}$ ***and*** $\text{bottom-depth}(y) \geq p_\triangleleft \tilde{o}$
(b) ***if*** $\text{bottom-depth}(x) = p_\triangleleft \tilde{o}$ ***or*** $\text{bottom-depth}(y) = p_\triangleleft \tilde{o}$
then $\text{inverse-plateau-depth}(x) \geq p_\triangleleft \tilde{o}$
(and hence $\text{inverse-plateau-depth}(y) \geq p_\triangleleft \tilde{o}$***).***

Proof of Claim 1. The claim is proven by induction on $p_\triangleleft + p_\triangledown + p_\triangleright$. The base case holds as by Condition 1 the nodes x and y have the same label. Now assume that $p_\triangleleft + p_\triangledown + p_\triangleright > 0$.

Strategy for $\exists_\triangledown$-moves. Assume that Spoiler picks an infinite downward fullpath π rooted at x in either $\bar{T}_{\tilde{o}+1}$ or $\bar{S}_{\tilde{o}+1}$. As x has a downward path rooted at it, its label cannot be c. Since x and y have the same label, it follows that there is also an infinite downward-fullpath rooted at y. In fact there are several such paths, and we now describe which of these Duplicator chooses.
Let x' be the child of x on π. We first determine the child y' of y on the path τ that Duplicator chooses. Duplicator's goal is to pick y' such that she can win the $\text{CTL}^*_{\triangleleft\triangleright}(\text{pd}_\triangleleft \leq p_\triangleleft, \text{pd}_\triangledown \leq p_\triangledown - 1, \text{pd}_\triangleright \leq p_\triangleright, \text{ud}_\triangleright \leq \tilde{u}_\triangleright; \text{other} \leq \tilde{o})$-game from x', y', and to assist in this they will ensure that the trees rooted at x' and y' have the same polarity: this will make "downward moves" in the game easy to win. The easiest case is when the node that corresponds to x' in the other tree has (in the sense defined above) has the same polarity as x': in this case Duplicator will choose this node. Otherwise there are two cases: If x' has positive polarity, then Duplicator picks the single child of y that has positive polarity. The remaining case is that x' has negative polarity and its corresponding node in the other tree has positive polarity. In this case Duplicator picks the unique child y' of y who's corresponding node has positive polarity (observe that y' has negative polarity). In any case x' has the same polarity as y'. Because the polarity of children of the root of a hedge determines what subtrees are added beneath, and because x and y are both roots of $\bar{S}_{\tilde{o}+1}$ and $\bar{T}_{\tilde{o}+1}$, the subtrees rooted at x' and y' are isomorphic. Thus Duplicator can choose a path τ' starting at y' such that (a) the path has the same labeling as the suffix π' of π starting at x' and (b) the node $\pi'(i)$ has the same polarity as the node $\tau'(i)$ for all $i \leq |\pi'|$. As π and τ have the same labeling, Duplicator can play isomorphically on the path game.

Maintaining the Invariant on $\exists_\triangledown$-moves. At the end of the path game Spoiler will pick an intermediate position from which the tree game will continue. Hence we verify that all possible intermediate positions of the path game are winning positions for Duplicator for the $\text{CTL}^*_{\triangleleft\triangleright}$-game with $p_\triangledown - 1$ downward path moves. This follows from the induction hypothesis if x and y – the roots of π and τ – is the final position. The main concern is when the intermediate positions are x' and y'. Since Spoiler can play horizontal moves from these positions, we cannot appeal isomorphism of the subtrees. For the case that Spoiler chooses to continue to play from position x', y', we verify that Duplicator can win the $\text{CTL}^*_{\triangleleft\triangleright}(\text{pd}_\triangleleft \leq p_\triangleleft, \text{pd}_\triangledown \leq p_\triangledown - 1, \text{pd}_\triangleright \leq p_\triangleright, \text{ud}_\triangleright \leq \tilde{u}_\triangleright; \text{other} \leq \tilde{o})$-game from x' and y'. The interesting case is when x' and y' both have positive polarity – in the other cases, x' and y' are isomorphic within their sibling string, and hence the invariant clearly holds. Assume that the sequence of

siblings of x' is obtained from F and that the siblings of y' are obtained from G. Then Condition 2 holds because both x' and y' have positive polarity. To check Condition 3 recall that the right fullpath starting at x is in the language

$$a(c^n a)^{\tilde{p}_\triangleright \tilde{u}_\triangleright + 1} c^n b\, \Sigma^\omega$$

with $n = \tilde{p}_\triangleright \cdot \tilde{o} + \tilde{o}^2$, while the right fullpath starting at y' is in the language

$$a(c^n a)^{\tilde{p}_\triangleright \tilde{u}_\triangleright} c^n b\, \Sigma^\omega.$$

Thus Conditions 3 and 4 are maintained. For Condition 5, observe that both x' and y' are contained in a block of siblings labelled

$$b\,(c^n a)^m\ c^n b$$

where $m = \tilde{p}_\triangleright \cdot \tilde{u}_\triangleright + \tilde{o}^2 + 2$. Hence the left path from x' to its next b-labelled left sibling is labelled $a(c^n a)^{\tilde{o}^2} c^n b$ (from right to left) and the string from y' to its next b left sibling is $a(c^n a)^{\tilde{o}^2+1} c^n b$ (from right to left). As $p_\triangleleft \le \tilde{o}$ this shows that Condition 5 is maintained.
If the position that Spoiler chooses is further down in π, τ than x', y', then the selected nodes are isomorphic positions in isomorphic hedges, and therefore winning positions for any $\mathrm{CTL}^*_{\triangleleft\triangleright}$-game.

$\exists_\triangleright$*-moves.* Now assume that Spoiler picks a rightward fullpath π in either hedge. Duplicator has to choose the unique rightward fullpath τ starting at the selected node in the other tree. The players then play the $\mathrm{LTL}(\mathrm{ud} \le \tilde{u}_\triangleright, \mathrm{ned} \le \tilde{o})$ game on the selected paths. In this game, Duplicator uses the strategy described in Section 2.3. The following claim can be shown using the strategy described in the proof of Claim 1 in Lemma 2.5.

SUBCLAIM 1.1. *Let x, y be positions that satisfy the conditions of Claim 1 and let $\tilde{\pi}, \tilde{\tau}$ be the right fullpaths starting at x, y respectively. For all $u_\triangleright \le \tilde{u}_\triangleright$ and $o \le \tilde{o}$, Duplicator has a winning strategy for the LTL$(\mathrm{ud} \le u_\triangleright, \mathrm{ned} \le o)$-game on suffixes π, τ of $\tilde{\pi}, \tilde{\tau}$ if*
(1) *π and τ have the same plateau-depth.*
(2) *$|\text{top-depth}(\pi) - \text{top-depth}(\tau)| - 1$.*
(3) *If π and τ have different top-depths then*
  (a) *$\text{top-depth}(\pi) \ge (p_\triangleright - 1)\tilde{u}_\triangleright + u_\triangleright$ and $\text{top-depth}(\tau) \ge (p_\triangleright - 1)\tilde{u}_\triangleright + u_\triangleright$*
  (b) *if $\text{top-depth}(\pi) = (p_\triangleright - 1)\tilde{u}_\triangleright + u_\triangleright$ or $\text{top-depth}(\tau) = (p_\triangleright - 1)\tilde{u}_\triangleright + u_\triangleright$ then $\text{plateau-depth}(\pi) \ge (p_\triangleright - 1)\tilde{o} + o$ (and hence $\text{plateau-depth}(\tau) \ge (p_\triangleright - 1)\tilde{o} + o$).*

In addition, every intermediate position of this LTL-game is a winning position for Duplicator in the CTL$^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \le p_\triangleleft, \mathrm{pd}_\triangledown \le p_\triangledown, \mathrm{pd}_\triangleright \le p_\triangleright - 1, \mathrm{ud}_\triangleright \le \tilde{u}_\triangleright; \mathrm{other} \le \tilde{o})$-game.

It is easy to check that the conditions of Subclaim 1.1 imply the conditions of Claim 1 for $(p_\triangleright, p_\triangleleft - 1, p_\triangledown)$ moves left to play.

$\exists_\triangleleft$*-moves.* Duplicator's strategy for leftward path moves is symmetric to her strategy on right paths moves. The proof that this strategy maintains the invariant follows the same lines as above. This concludes the proof of Claim 1. □

This concludes the proof of Theorem 1.3 for infinite trees.

3.4. The Finite Case

We now describe what needs to be changed, in order to prove Theorem 1.3 for finite trees. Recall that roots of both F and G spell out the infinite word

$$\left((c^n a)^m\ c^n b\right)^\omega$$

where $n = \tilde{p}_\triangleright \cdot \tilde{o} + \tilde{o}^2$ and $m = \tilde{p}_\triangleright \cdot \tilde{u}_\triangleright + \tilde{o}^2 + 2$. Consider a template F^{fin} that is obtained from F by pruning in such a way that the sequence of its roots spells out the word

$$\left((c^n a)^m\ c^n b\right)^{2(\tilde{o}^2 + \tilde{o}\tilde{p}_\triangleright)}$$

and the right-path starting at the unique root with a positive child is labelled $a(c^n a)^{\tilde{p}_\triangleright \tilde{u}_\triangleright + 1}\, c^n b\, w^{2op_\triangleright}$ for $w = (c^n a)^m\, c^n b$. The finite template G^{fin} is obtained from G in a similar way, with the right-path starting at the parent of the unique positive port being in the language $a(c^* a)^{\tilde{p}_\triangleright \tilde{u}_\triangleright} c^* b\, w^{2op_\triangleright}$. Consider the finite hedges $\bar{T}_k^{\text{fin}}, \bar{S}_k^{\text{fin}}$ that are constructed like the hedges $\bar{T}_k$ and $\bar{S}_k$ but from templates F^{fin} and G^{fin} instead of from the templates F and G. We argue that Duplicator can win the $\text{CTL}^*_{\triangleleft\triangleright}(\text{pd}_\triangleleft \le p_\triangleleft, \text{pd}_\triangledown \le p_\triangledown, \text{pd}_\triangleright \le p_\triangleright, \text{ud}_\triangleright \le \tilde{u}_\triangleright; \text{other} \le \tilde{o})$-game on $\bar{S}^{\text{fin}}_{\tilde{o}+1}$ and $\bar{T}^{\text{fin}}_{\tilde{o}+1}$. In fact, one can show that if the current position is x and y, then Duplicator can maintain the invariant of Claim 1 together with the additional condition:

(6) If the number of b-labelled nodes to the right x is not equal to the number of b-labelled nodes to the right y, then there are at least $2\tilde{o}\tilde{p}_\triangleright$ b-labelled nodes to the right of both x and y and there are at least $2\tilde{o}^2$ b-labelled nodes to the left of both x and y.

On path moves, Duplicator can reuse her strategy from the infinite case. In the path game, Duplicator can use the same strategy as in the infinite case on downward paths, and the Etessami-Wilke strategy from Corollary 2.6 on horizontal paths. It is easy to check that this strategy preserves the invariant. This concludes the proof of Theorem 1.3 for finite trees.

4. THE VERTICAL UNTIL HIERARCHY

We now show Theorem 1.4. The structure of the proof is similar to the structure of the proof of Theorem 1.3: For each $u_\triangledown \in \mathbb{N}$ we define a property $P_{u_\triangledown}$ that can be expressed in $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le u_\triangledown)$ but not in $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le u_\triangledown - 1)$. Again, we first show Theorem 1.4 for infinite trees and then we discuss what needs to be changed for the finite case.

4.1. The Separating Property

For $u_\triangledown \in \mathbb{N}$ we define $P_{u_\triangledown}$ to be the set of ordered trees which have a fullpath π ending at d such that each suffix of π that starts with b satisfies the regular expression

$$b(c^* a)^{u_\triangledown}\, \Sigma^\omega.$$

$P_{u_\triangledown}$ can be expressed in $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le u_\triangledown)$ by the formula 'there is a downward fullpath ending at d on which every node satisfies $b \to \mu_{u_\triangledown}$' where $\mu_{u_\triangledown}$ is defined by

$$\mu_i = \begin{cases} \text{true} & \text{if } i = 0 \\ c\, U\, (a \wedge \mu_{i-1}) & \text{if } i > 0 \end{cases}$$

The following lemma shows that $P_{u_\triangledown+1}$ cannot be expressed in $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le u_\triangledown)$.

LEMMA 4.1. *For all $u_\triangledown, o \in \mathbb{N}$ there are finite trees $\tilde{T}$ and $\tilde{S}$ such that (i) $\tilde{T}$ satisfies $P_{u_\triangledown+1}$ but $\tilde{S}$ does not and (ii) Duplicator has a winning strategy for the $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le u_\triangledown; other \le o)$ game on $\tilde{T}, \tilde{S}$.*

Before we prove Lemma 4.1 we show how it implies Theorem 1.4.

PROOF OF THEOREM 1.4. Assume towards a contradiction that there is a number $u_\triangledown \in \mathbb{N}$ such that $P_{u_\triangledown+1}$ can be expressed in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le u_\triangledown)$. Then there is a formula ϕ in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le u_\triangledown)$ that expresses P_u. In particular, ϕ is in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le u_\triangledown; \mathrm{other} \le o)$ for some $o \in \mathbb{N}$.

For the numbers $u_\triangledown$ and o there are, according to Lemma 4.1, two trees $\tilde{T}$ and $\tilde{S}$ on which Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le u_\triangledown; \mathrm{other} \le o)$-game. Thus, by Proposition 2.2, no $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le u_\triangledown; \mathrm{other} \le o)$ formula can distinguish $\tilde{T}$ and $\tilde{S}$. Therefore ϕ either holds on both $\tilde{T}$ and $\tilde{S}$ or on neither of them. As $\tilde{T}$ satisfies $P_{u_\triangledown+1}$ but $\tilde{S}$ does not, it follows that ϕ does not define $P_{u_\triangledown+1}$. This is a contradiction of the assumption that ϕ defines P_u. □

The proof of Lemma 4.1 is quite involved and will take up the rest of this section. In Section 4.2 we construct two trees $\tilde{T}$ and $\tilde{S}$ on which the $\mathrm{CTL}^*_{\triangleleft\triangleright}$-game will be played. Before we define Duplicator's strategy formally, we explain its intuition in Section 4.3. Section 4.4 describes Duplicator's strategy in detail, and we show that it is a winning strategy. To simplify the presentation, Sections 4.2 – 4.4 prove the lemma for infinite trees. Section 4.5 describes which modifications are necessary for the proof on finite trees.

4.2. The Trees

We now fix two numbers $\tilde{u}_\triangledown, \tilde{o} \in \mathbb{N}$ for the rest of the proof of Lemma 4.1. We construct two families of infinite binary trees $(T_k)_{k\in\mathbb{N}}$ and $(S_k)_{k\in\mathbb{N}}$ from the templates F and G shown in Figure 6. The only distinction between F and G lies in the node labelled "±": it is a positive port in F and a negative port in G. In each of the two templates, the leftmost branch consists of the root labelled b followed by infinitely many $c^{\tilde{o}}a$ blocks. The final c node in any $c^{\tilde{o}}$ sequence has two children: the left child is labelled a and the right child is a positive or negative port. In both F and G the topmost $\tilde{u}_\triangledown$ ports are negative. In F the next two ports are positive, while in G only the next port is positive. All other ports are negative.

We define the sequences $(T_k)_{k\in\mathbb{N}}$ and $(S_k)_{k\in\mathbb{N}}$ by induction on k. T_0 consists of a single node labelled d, while S_0 consists of a single node labelled c. For $k \ge 1$:

$$T_k := F\left[T_{k-1}, S_{k-1}\right]$$
$$S_k := G\left[T_{k-1}, S_{k-1}\right]$$

For trees T, T' we write $T \equiv T'$ iff T and T' agree in all $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$ formulas. It is easy to see that $\equiv$ is an equivalence relation. The next lemma shows that $\equiv$ has a finite number of equivalence classes.

LEMMA 4.2. *The relation $\equiv$ has at most $\mathcal{O}(\Sigma^{\mathcal{O}(2^m)})$ equivalence classes where $m = \max(\tilde{o}^2, \tilde{u}_\triangledown\tilde{o})$.*

PROOF. We show that there are at most $\mathcal{O}(\Sigma^{\mathcal{O}(2^m)})$, $m = \max(\tilde{o}^2, \tilde{u}_\triangledown\tilde{o})$ different syntax trees of formulas in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$. Note that each such syntax tree is a binary tree that has depth at most $m = \max(\tilde{o}^2, \tilde{u}_\triangledown\tilde{o})$. Each node is labelled by either one of the symbols $\{U, X, F, \wedge, \vee, \neg\}$ or by a predicate in Σ. As a binary tree of depth m has at most $2(2^m)$ nodes, it follows that there are at most $(6 + |\Sigma|)^{2(2^m)} = \mathcal{O}(\Sigma^{\mathcal{O}(2^m)})$ syntax trees of formulas in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$. □

Definition 4.3 (*Period*). A pair of numbers (μ, λ) is a *period* of the sequences $(T_k)_{k\in\mathbb{N}}$ and $(S_k)_{k\in\mathbb{N}}$ if for all $k \ge \mu$, $T_k \equiv T_{k+\lambda}$ and $S_k \equiv S_{k+\lambda}$.

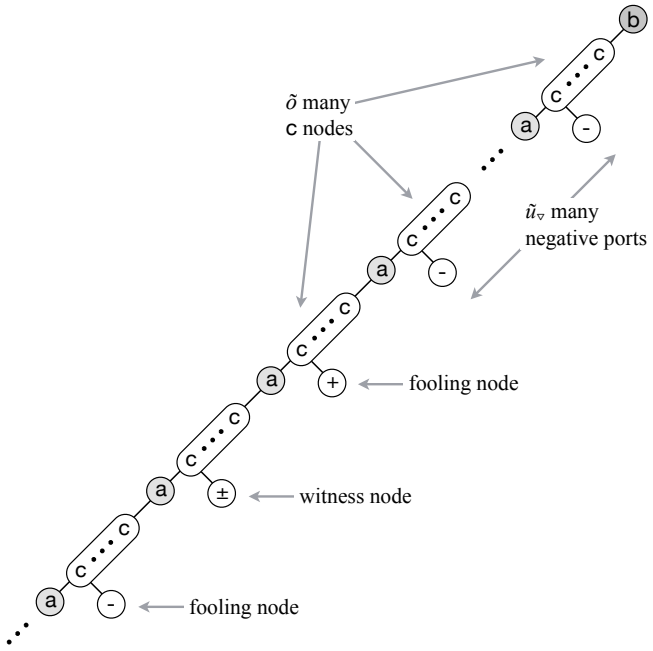


Fig. 6. The templates F and G

LEMMA 4.4. *$(T_k)_{k\in\mathbb{N}}$ and $(S_k)_{k\in\mathbb{N}}$ have a period (μ, λ).*

PROOF. Let E_k be the pair of $\equiv$-classes of T_k and S_k. Since the number of equivalence classes of $\equiv$ is finite, there must be μ and λ such that $E_{\mu+\lambda} = E_\mu$, and we claim that this μ and λ suffice. Note that the $\equiv$-class of $F[T_k, S_k]$ (and $G[T_k, S_k]$) depends only on the $\equiv$-classes of T_k and S_k for all $k \in \mathbb{N}$. This follows by induction using the usual composition technique for trees (see e.g. [Moller and Rabinovich 1999]). Hence from $T_k \equiv T_{k+\lambda}$ and $S_k \equiv S_{k+\lambda}$ we can conclude (applying F to both equivalence classes) that $T_{k+1} \equiv T_{k+\lambda+1}$, and that $S_{k+1} \equiv S_{k+\lambda+1}$ (applying G to both equivalence classes). The result now follows by induction. □

Fix a period (μ, λ) of $(T_k)_{k\in\mathbb{N}}$ and $(S_k)_{k\in\mathbb{N}}$ for the rest of this proof. We can now define the trees $\tilde{T}, \tilde{S}$ on which the $\mathrm{CTL}^*_{\triangleleft\triangleright}$-game will be played.

Definition 4.5. Let $k = 3\tilde{o}^2\lambda + \mu + 1$. Then we define

$$\tilde{T} := T_k \qquad\qquad \tilde{S} := S_k$$

Notation. We call a $b(c^*a)^\omega$ labelled path within T_k or S_k a *stem*. Observe that a stem is isomorphic to F (or G) without ports. A node that is connected to the root of a stem by a path labelled $b(c^*a)^i c^* b$ is called a *witness node* if $i = \tilde{u}_\triangledown + 1$ and *fooling node* if $i = \tilde{u}_\triangledown$ or $i = \tilde{u}_\triangledown + 2$. A path that always departs from the stem on the witness node is a *witness path*. Observe that T_k contains a single witness path that starts at the root, and both T_k and S_k contain several witness paths that start at intermediate nodes. The *enclosing subtree* of a node x in $T \in \{T_k, S_k\}$ is the smallest subtree of T that contains x and that is isomorphic to either T_j or S_j for some $j \le k$. A subtree T of T_k or S_k has *positive polarity* if it is isomorphic to T_j for some $j \le k$ and T has *negative polarity* otherwise. The *polarity* of a node x is the polarity of its enclosing subtree.

We now show part (i) of Lemma 4.1.

PROPOSITION 4.6. *For all $k \in \mathbb{N}$, T_k satisfies $P_{\tilde{u}_\triangledown+1}$ but S_k does not.*

PROOF. The proof is by induction on k. The proposition obviously holds for the trees T_0 and S_0. Now let $k > 0$. Note that because the witness path departs from the topmost stem on the witness node, it starts with a sequence labelled $b(c^*a)^{\tilde{u}_\triangledown+1}c^*b$. Hence the first b is followed by sufficiently many a nodes. In addition, it departs from the topmost stem into a subtree isomorphic to T_{k-1}. Since T_{k-1} satisfies $P_{\tilde{u}_\triangledown}$ by induction, we have that T_k satisfies $P_{\tilde{u}_\triangledown}$.

We now show that S_k does not satisfy $P_{\tilde{u}_\triangledown}$. We know by induction that any strict subtree S of S_k of negative polarity does not contain a path witnessing that $S \in P_{\tilde{u}_\triangledown+1}$. Hence a path witnessing $S_k \in P_{\tilde{u}_\triangledown+1}$ must contain the upper fooling node of the topmost stem. But any path that contains this node starts with a sequence labelled $b(c^*a)^{\tilde{u}_\triangledown}c^*b$. □

4.3. The Strategy: Challenges for Duplicator

To prove part (ii) of Lemma 4.1 we will need to show that Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le u_\triangledown; other \le o)$ game on $\tilde{T}, \tilde{S}$. We start with some intuition about the strategy of Duplicator. The idea is that $\tilde{T}$ and $\tilde{S}$ consist of several "similar levels" of subtrees. We will show that there is a number λ such subtrees on levels that are λ apart and of the same polarity are winning positions for Duplicator. The game will start on similar levels, but on trees of different polarity. As Spoiler eventually wins on trees of non-similar levels, Duplicator must assure that the selected nodes stay on similar levels throughout the game. Duplicator cannot force the game to a position with both similar levels and of the same polarity, thus her strategy is to maintain the position on similar levels regardless of the polarities. As Duplicator has to keep the position on similar levels, Spoiler can force the game down a fixed number of levels on each move. Thus Duplicator can only win the game on trees with very many similar levels. Thereby she must assure that the selected nodes are high up in the tree if the polarities of their enclosing subtrees differ.

Basically, Duplicator will have to disguise that the witness path in $\tilde{T}$ ends on a d while the witness path in $\tilde{S}$ ends in a c. Clearly, Duplicator must have a remedy when Spoiler plays the witness path in $\tilde{T}$. In fact, Duplicator's strategy depends on the place where Spoiler's path departs from the witness path. Assume Spoiler picks a path $\pi\pi'$ that departs from the witness path on the root of π'. Then Duplicator's response depends on the length on π.

The case where π is short is "easy" for Duplicator: she picks a path $\tau\tau'$ such that τ is isomorphic to π. To choose τ', observe that there are two possibilities for path $\pi\pi'$ to depart from the witness path. If $\pi\pi'$ departs above the witness node then Duplicator can choose the root of τ' such that the roots π' and τ' have the same polarity. As Duplicator maintained similar levels throughout the game the roots of π' and τ' are on similar levels and hence winning positions for Duplicator. She can use her winning strategy to determine the rest of τ'. If $\pi\pi'$ does not depart from the witness path above the witness node then it departs on the (a-labelled) sibling of the witness node. We will see that in this case Duplicator has an "obvious" strategy to determine τ'. It will be easy to see in either of these cases that Duplicator wins the path game if the paths are chosen in this way. There are two cases for its final position: If the final position is in π, τ, the Duplicator has achieved her goal to keep the trees big, and therefore she wins by induction. If the final position is in π', τ' then the selected nodes are on similar levels and of the same polarity – and Duplicator wins by the definition of similar levels.

The case that π is long is more threatening for Duplicator. If Duplicator uses the strategy for "short" moves described above then the final position of the path game might be in small subtrees of different polarity. In this case it is not guaranteed that there are sufficiently many levels below the selected nodes for Duplicator to use her strategy. Therefore Duplicator will respond to such a "long" move of Spoiler by picking a path that moves off of a stem at a different point some place down the tree – Duplicator has some flexibility as to where to do this "fooling", which we will exploit.

But given that Duplicator has played a fooling path, the first cause for concern is that Spoiler may try to detect a distinction in the paths by moving to the "fooling point" where the two paths are first distinguished – the point in which one path departs from a stem at a different point from the other path. Note that on the witness path, the number of c^*a blocks between the root of a stem and the departure point is $u_\triangledown + 1$. Hence Spoiler will be unable to use only until moves to force the play to this point on the critical path, since his until moves are limited to $u_\triangledown$. But one must still worry that Spoiler can try to push the play down to this point using eventually moves, which he has in some abundance. The response of the Duplicator to these threatening eventually moves will be to jump down to a lower stem. This is analogous to the strategy used by the Duplicator in the linear case of the until-depth hierarchy theorem of Etessami and Wilke ([Etessami and Wilke 2000], Theorem 2.3); there, the Duplicator responds to eventually moves of the Spoiler by jumping to next $b(c^*a)^*c^*b$ block in the word.

However, this "jumping response" of Duplicator cannot be done so naively in the setting of ordered trees. If Duplicator jumps so that the position is only one level off from the position of Spoiler, then the two nodes are on non-similar levels. In particular the selected nodes are in enclosing subtrees T_i and S_j where i and j have different parities; Spoiler can detect this difference in parity of i and j by playing paths that alternate in the way they jump from stem to stem: e.g. by playing a path that will depart after two a's on even levels and after one a on odd levels. This method of detecting differences in trees of different depths goes back to Potthoff [Potthoff 1995]. The general problem is that two distinct depths of the tree could have cardinalities with different properties, and this difference can be exposed by further path moves.

Duplicator will remedy this problem by making not a small jump down one stem, but an "exaggerated jump" that moves down λ stems to a place that looks locally (on its stem) isomorphic to the place where Duplicator has played. How do we ensure that a locally similar place exists? Duplicator will make sure that in all cases where Spoiler can execute this strategy, the currently-played paths below the fooling point begin with a large segment of the witness path. Duplicator can guarantee this on path moves because if π is not long, there is no need to perform fooling at all. On the other hand, if π is long, Duplicator can play a path that has a long regular structure at the top, which allows her to perform the exaggerated jump.

4.4. Duplicator's Strategy in Detail

We now formalize Duplicator's winning strategy in order to prove part (ii) of Lemma 4.1. We start with the notion of 'similar levels'.

Definition 4.7. Let (μ, λ) be the period of $(T_k)_{k\in\mathbb{N}}$ and $(S_k)_{k\in\mathbb{N}}$. We define $\doteq \subseteq \mathbb{N}\times\mathbb{N}$ by

$$n \doteq m \quad \text{iff} \quad n, m \geq \mu \text{ and } n = m + \lambda i \text{ for some } i \in \mathbb{Z}.$$

The following is an immediate consequence of the definition of $\doteq$ and Proposition 2.2.

FACT 4.8. *Duplicator has a winning strategy for the* $\mathbf{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} \leq \tilde{u}_\triangledown, \mathrm{other} \leq \tilde{o})$*-game on* T_i, T_j *if* $i \doteq j$ *and the roots are selected. The same holds for* S_i, S_j.

Notation. A node x in $\tilde{T}$ or $\tilde{S}$ is on *level* i if its enclosing subtree is isomorphic to either T_i or S_i. Two nodes x, y in $\tilde{T}$ or $\tilde{S}$ are on *similar levels* if x is on level i, y is on level j, and $i \doteq j$. In this case we write $x \doteq y$. We denote by $\text{plateau-depth}(x)$ the plateau-depth of x on the stem that contains x. Given a tree T, we denote by $T[x]$ the subtree of T rooted at x.

We noted in the previous section that Duplicator will have to maintain similar positions throughout the game. If she can, in addition, achieve a position in which the selected nodes are locally isomorphic on their stems, and both nodes are not on the witness path then she can win the $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le u_\triangledown; \text{other} \le o)$-game if it proceeds strictly downwards.

CLAIM 1. *Let x and y be nodes in $\tilde{T}$ and $\tilde{S}$ respectively, that are on similar levels, have the same plateau depth, the same label, and are both not on witness paths (as defined in the paragraph below Definition 4.5) . Then Duplicator can win the $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud} \le \tilde{u}_\triangledown, \text{other} \le \tilde{o})$-game on the subtrees $\tilde{T}[x], \tilde{S}[y]$ of $\tilde{T}, \tilde{S}$ rooted at x, y respectively.*

Proof of Claim 1. Without loss of generality we assume that x is positive and y is negative. Hence the enclosing subtree of x is T_i for some i and the enclosing subtree of y is S_j for some j. There is a unique node x' in S_i such that the path from the root of S_i to x' is isomorphic to the path from the root of T_i to x. Note that as x is not on any witness path, the subtree $S_i[x']$ of S_i rooted at x' is isomorphic to the subtree $T_i[x]$ of T_i rooted at x. In addition, Duplicator has a winning strategy for the $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud} \le \tilde{u}_\triangledown, \text{other} \le \tilde{o})$ game on $S_i[x']$ and $S_j[y]$ by Fact 4.8. The result follows because $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud} \le \tilde{u}_\triangledown, \text{other} \le \tilde{o})$ is closed under isomorphism. □

We have seen that Duplicator has a simple winning strategy from certain positions in $\tilde{T}$ and $\tilde{S}$. Unfortunately, if the roots of $\tilde{T}$ and $\tilde{S}$ are selected, neither Fact 4.8 nor Claim 1 applies. The following claim is concerned with this situation – that is, it implies that Duplicator has a winning strategy for the $\text{CTL}^*_{\triangleleft\triangleright}(\text{ud}_\triangledown \le \tilde{u}_\triangledown; other \le \tilde{o})$ game on $\tilde{T}, \tilde{S}$ if the roots are selected. Hence it proves part (ii) of Lemma 4.1.

CLAIM 2. *Let $p_\triangleleft, p_\triangledown, p_\triangleright \le \tilde{o}$. Duplicator has a winning strategy for the $\text{CTL}^*_{\triangleleft\triangleright}(\text{pd}_\triangleleft \le p_\triangleleft, \text{pd}_\triangledown \le p_\triangledown, \text{pd}_\triangleright \le p_\triangleright, \text{ud}_\triangledown \le \tilde{u}_\triangledown; \text{other} \le \tilde{o})$-game on $\tilde{T}, \tilde{S}$ if the selected nodes x, y satisfy*
(1) x, y have the same label,
(2) x, y are on similar levels,
(3) x, y have the same plateau-depth,
(4) $\text{level}(x) > k$ and $\text{level}(y) > k$ where $k = (\lambda(\tilde{u}_\triangledown + \tilde{o}) + 1)(p_\triangleleft + p_\triangledown + p_\triangleright) + \mu$.

Proof of Claim 2. We prove the claim by induction on $p_\triangleleft + p_\triangledown + p_\triangleright$. The base case $p_\triangleleft + p_\triangledown + p_\triangleright = 0$ holds because x and y have the same label by Condition 1. For the induction step, we fix $p_\triangleleft, p_\triangledown, p_\triangleright$ such that $p_\triangleleft + p_\triangledown + p_\triangleright > 0$. If Spoiler plays a horizontal move, then Duplicator's strategy is trivial. For downward moves we distinguish several cases.

Case 1. Spoiler plays a downward move on $\tilde{T}$. Assume that Spoiler chooses a downward path π that is rooted at x in $\tilde{T}$. As noted above, Duplicator's strategy depends on the length of the prefix of π that is on a witness path. Therefore let $\pi_1, \ldots, \pi_n$ be a partition of π such that $\pi_1 \ldots \pi_{n-1}$ is a maximal prefix of π on a witness path, and for each $i \le n-1$, π_i is contained in exactly one stem. We will call Spoiler's move a *short move* if $n \le \lambda(\tilde{u}_\triangledown + \tilde{o}) + 1$ and a

long move otherwise. Duplicator's strategy is different for the two kinds of moves.

Case 1.1. Short downward move on $\tilde{T}$. As described in the previous subsection, Duplicator has an easy strategy in this case. Duplicator first chooses a prefix $\tau_1 \cdot \ldots \cdot \tau_{n-1}$ of the full path $\tau = \tau_1 \cdot \ldots \cdot \tau_n$ that she will choose in the game. She picks this prefix such that τ_i has the same labeling as π_i for $1 \leq i \leq n-1$ (note that any downward path in $\tilde{T}$ and $\tilde{S}$ is determined by its labeling). The τ_i exist, because by Condition 4, there are enough levels below to accommodate a path of this length, and because by construction a path not ending at a leaf in one tree can be mimiced in the other tree.
To determine τ_n, recall that π departs from the witness path on the root of π_n. There are two ways in which a path can depart from a witness path: above the witness node or on the sibling of the witness node. In the first case the root x_n of π_n is labelled b and in the second case x_n is labelled a. In both cases we define the root y_n of τ_n to be the child of the leaf of τ_{n-1} that has the same label x_n.

— If π departs from the witness path on the sibling of the witness node, then both x_n and y_n are not on *any* witness path. Hence it follows from Claim 1 that Duplicator wins the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$-game on $\tilde{T}[x_n], \tilde{S}[y_n]$.
— If π departs from the witness path above the witness node, then the subtrees rooted at x_n and y_n have the same polarity. As x_n and y_n are on similar levels, Duplicator can win the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$ on $\tilde{T}[x_n], \tilde{S}[y_n]$ by Fact 4.8.

In both cases, Duplicator can use her winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$-game on $\tilde{T}[x_n], \tilde{S}[y_n]$ to determine a path τ_n rooted at y_n such that she has a winning strategy for the $\mathrm{LTL}(\mathrm{ud} \leq \tilde{u}_\triangledown, \mathrm{ned} \leq \tilde{o})$-game on π_n, τ_n.

The LTL-game after short downward moves on $\tilde{T}$. We now describe Duplicator's strategy for the LTL game on π, τ. Duplicator can the play isomorphically on $\pi_0 \ldots \pi_{n-1}$ and $\tau_0 \ldots \tau_{n-1}$ as these paths are isomorphic (that is, she picks the n-th node whenever Spoiler picks the n-th node in the other path). If Spoiler chooses a node in π_n or τ_n then Duplicator can use her winning strategy for the $\mathrm{LTL}(\mathrm{ud} \leq \tilde{u}_\triangledown, \mathrm{ned} \leq \tilde{o})$ on π_n, τ_n. These strategies can be composed to derive a winning strategy for π, τ. This composed strategy has the property that if π', τ' is an intermediate position of the path-game, then either the root of π' is in $\pi_0 \cdot \ldots \cdot \pi_{n-1}$ and the root of τ' is in $\tau_0 \cdot \ldots \cdot \tau_{n-1}$ or the root of π' in π_n and the root of τ' is in τ_n.

It remains to show that any intermediate position π', τ' of the path-game on π, τ is a winning position for Duplicator in the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \leq p_\triangleleft, \mathrm{pd}_\triangledown \leq p_\triangledown - 1, \mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$-game. For positions where the root of π' is in $\pi_0 \cdot \ldots \cdot \pi_{n-1}$ and the root of τ' is in $\tau_0 \cdot \ldots \cdot \tau_{n-1}$ it is easy to see that the Conditions 2, 3 and 1 of Claim 2 are true. Condition 4 is satisfied as $\pi_0 \cdot \ldots \cdot \pi_{n-1}$ and $\tau_0 \cdot \ldots \cdot \tau_{n-1}$ are 'short' (that is, $n \leq \lambda(u_\triangledown + o) + 1$) and hence the levels of the roots of both π' and τ' are sufficiently high up in the trees. Thus Duplicator wins on these positions by induction. The same argument applies if the roots of π_n and τ_n are selected as the new position. Now consider the case that the root of π' is a descendant of x_n and the root of τ' is a descendant of y_n. In this case the subsequent game can never leave the trees $\tilde{T}[x_n], \tilde{S}[y_n]$ (this is because the game can only move horizontally

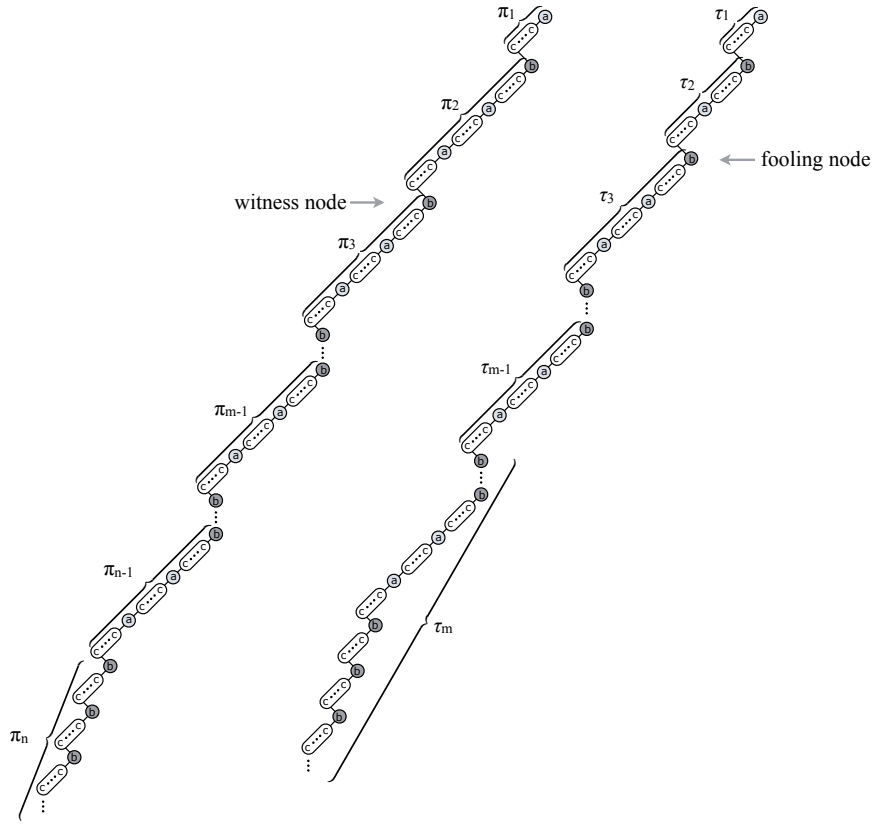


Fig. 7. The paths chosen by Spoiler and Duplicator in Case 1.2.

and downwards – but never upwards). Then the new position is a winning position for Duplicator because by construction, she has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$-game on $\tilde{T}[x_n], \tilde{S}[y_n]$.

Case 1.2. Long downward move on $\tilde{T}$. Recall that in this case Spoiler chooses a path $\pi = \pi_1 \dots \pi_n$ in $\tilde{T}$ such that $n > \lambda(u_\triangledown + o) + 1$ and $\pi_1 \dots \pi_{n-1}$ is a maximal prefix of π on a witness path and each π_i with $i \le n-1$ is contained in exactly one stem. The idea is that Duplicator picks a path $\tau = \tau_1 \dots \tau_m$ for some large m that is smaller than n such that τ_1 is isomorphic to π_1, τ_2 is similar to but different from π_2 – different in a way that is undetectable by Spoiler in the game (as we show below). The path $\tau_3 \dots \tau_{m-1}$ is a long sequence of segments on the witness path that is isomorphic to $\pi_3 \dots \pi_{m-1}$, and τ_m is chosen by induction in such a way that it is indistinguishable from the path $\pi_m \dots \pi_n$ in a suitable LTL game (see Figure 7). Formally

— τ_1 is isomorphic to π_1.
— τ_2 is a prefix of a stem that departs from its stem on the upper fooling node. Note that τ_2 is labelled $bc^{\tilde{o}}(ac^{\tilde{o}})^{u_\triangledown - 1}$. This is Duplicator's 'fooling' move.
— $m = \lambda(u_\triangledown + o) + 1$.
— τ_i is isomorphic to π_i for all $3 \le i \le m-1$.

The τ_i with $i \leq m$ exist by Condition 4.
We still need to specify τ_m to complete the description of Duplicator's path move. Let x_m be the root of π_m and let y_m be the child of the leaf of τ_{m-1} that has the same label as x_m. Observe that due to Duplicator's fooling move, the roots of x_3 and y_3 of π_3 and τ_3 have positive polarity. As $\tau_3 \dots \tau_{m-1}$ is isomorphic to $\pi_3 \dots \pi_{m-1}$ it follows that x_m and y_m have the same polarity. In addition, as the positions x, y at the beginning of the current round were on similar levels (Condition 2) it follows that x_m and y_m are on similar levels. Finally, as both positions are the roots of their respective stems, they have the same plateau depth. Hence it follows from Fact 4.8 that Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$-game on $\tilde{T}[x_m], \tilde{S}[y_m]$. This allows Duplicator to determine a path τ' rooted at y_m such that (a) she wins the $\mathrm{LTL}(\mathrm{ud} \leq \tilde{u}_\triangledown, \mathrm{ned} \leq \tilde{o})$-game on $\pi_m \dots \pi_n$ and τ' and (b) every intermediate position of the LTL-game is a winning position for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$-game on $\tilde{T}[x_m], \tilde{S}[y_m]$. Duplicator chooses $\tau_m = \tau'$, which completes the description of $\tau = \tau_1 \dots \tau_m$. Figure 7 shows the paths π and τ.

The LTL game after long downward moves on $\tilde{T}$. Let π, τ be the paths chosen in the current round as described above and let $p_\triangleleft, p_\triangledown, p_\triangleright$ be the numbers fixed at the beginning of the proof of Claim 2. Then the following subclaim implies Claim 2.

> SUBCLAIM 2.1. *Duplicator can win the LTL$(\mathrm{ud} \leq \tilde{u}_\triangledown, \mathrm{ned} \leq \tilde{o})$-game on π and τ. In addition she can play in such a way that any intermediate position is a winning position for her in the CTL$^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \leq p_\triangleleft, \mathrm{pd}_\triangledown \leq p_\triangledown - 1, \mathrm{pd}_\triangleright \leq p_\triangleright, \mathrm{ud}_\triangledown \leq \tilde{u}_\triangledown; \mathrm{other} \leq \tilde{o})$-game on $\tilde{T}, \tilde{S}$.*

We will prove Subclaim 2.1 by defining winning strategies for Duplicator in three $\mathrm{LTL}(\mathrm{ud} \leq \tilde{u}_\triangledown, \mathrm{ned} \leq \tilde{o})$-games: one on π_1 and τ_1, one on $\pi_2 \dots \pi_{m-1}$ and $\tau_2 \dots \tau_{m-1}$, and one on $\pi_m \dots \pi_n$ and τ_m. All these strategies will have the property that intermediate positions are winning positions for Duplicator in a suitable restriction of the $\mathrm{CTL}^*_{\triangleleft\triangleright}$-game. As in Case 1.1. these strategies can be combined to prove Subclaim 2.1.
The strategy for the game π_1, τ_1 is easy: as the paths are isomorphic, Duplicator will play isomorphically (that is, if Spoiler picks the i-th node in either path, then Duplicator will pick the i-th node in the other path). It follows from the hypothesis of Claim 2 that any intermediate position satisfies all the Conditions of Claim 2 for $p_\triangleleft, p_\triangledown - 1, p_\triangleright, \tilde{u}_\triangledown, \tilde{o}$.
A suitable strategy for Duplicator in the game on $\pi_m \dots \pi_n$ and τ_m exists by definition of τ_m.
It remains to show that Duplicator has a winning strategy for the game on $\pi_2 \dots \pi_{m-1}$ and $\tau_2 \dots \tau_{m-1}$. Observe that the paths $\pi_2 \dots \pi_{m-1}$ and $\tau_2 \dots \tau_{m-1}$ are 'long' versions of the paths used in Corollary 2.6. A stem corresponds to a staircase, and hence the *top-depth* of a node corresponds to its distance to the place where the path leaves the stem. A block of the form $c^{\tilde{o}}a$ within a stem corresponds to a plateau, and hence the *plateau-depth* is the position within such a block. Let the *end-depth* of a path π be the number of b labelled nodes on π – that is, the number of stems remaining in the path.
We show that Duplicator can maintain an invariant similar to the one in Claim 1 in the proof of Claim 2.5. There are two differences. First, the paths $\pi_1 \dots \pi_m$ and $\tau_1 \dots \tau_m$ are finite, and hence Duplicator must make sure that Spoiler cannot exploit that the end of the paths might have different dis-

tances from the selected nodes. In addition, the position at the end of the path game must satisfy the conditions of Claim 2. In particular they must be on similar levels. This will have an impact on Duplicator's strategy on F- and U-moves: if Duplicator were to play according to the strategy described in Claim 1 in the proof of Lemma 2.5, then the positions could end up being exactly one stem apart, and hence on *non*-similar levels. Therefore, Duplicator will jump to the next position that is locally isomorphic *and* on a similar level to the position that Spoiler chose (whereas in the proof of Claim 1 of Lemma 2.5 it was sufficient to jump to the next locally isomorphic position). After this 'exaggerated jump', Duplicator will be able to play according to a simpler strategy. The invariant of Duplicator's strategy is given in the following subclaim:

SUBCLAIM 2.1.1. *[Duplicator Strategy] Let* $u_\triangledown \le \tilde{u}_\triangledown$ *and* $o \le \tilde{o}$*. Then Duplicator has a winning strategy for the LTL*$(\mathrm{ud} \le u_\triangledown, \mathrm{ned} \le o)$*-game on* $\pi_2 \dots \pi_{m-1}$ *and* $\tau_2 \dots \tau_{m-1}$*, if the selected suffixes* $\hat{\pi}, \hat{\tau}$ *satisfy:*

(*1*) *the roots of* $\hat{\pi}$ *and* $\hat{\tau}$ *have the same label.*
(*2*) $\hat{\pi}$ *and* $\hat{\tau}$ *have the same plateau-depth.*
(*3*) $|\text{top-depth}(\hat{\pi}) - \text{top-depth}(\hat{\tau})| \le 1$.
(*4*) *If* $\hat{\pi}$ *and* $\hat{\tau}$ *have different top-depths then*
  (*a*) $\text{top-depth}(\hat{\pi}) \ge u_\triangledown$ *and* $\text{top-depth}(\hat{\tau}) \ge u_\triangledown$ *and*
  (*b*) *if* $\text{top-depth}(\hat{\pi}) = u_\triangledown$ *or* $\text{top-depth}(\hat{\tau}) = u_\triangledown$
  then $\text{plateau-depth}(\hat{\pi}) > o$ *(and hence* $\text{plateau-depth}(\hat{\tau}) > o$*).*
(*5*) $\hat{\pi}$ *and* $\hat{\tau}$ *are on similar levels.*
(*6*) *Either* $\hat{\pi}$ *and* $\hat{\tau}$ *have the same end-depth or*
  $|\text{end-depth}(\hat{\pi}) - \text{end-depth}(\hat{\tau})| = \lambda$*,* $\text{end-depth}(\hat{\pi}) \ge \lambda(u_\triangledown + o)$*, and* $\text{end-depth}(\hat{\tau}) \ge \lambda(u_\triangledown + o)$.

In addition, Duplicator can play in such a way that each intermediate position is a winning position for her in the $CTL^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \le p_\triangleleft, \mathrm{pd}_\triangledown \le p_\triangledown - 1, \mathrm{pd}_\triangleright \le p_\triangleright, \mathrm{ud}_\triangledown \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$*-game.*

Proof of Subclaim 2.1.1. The proof is by induction on $u_\triangledown + o$. The base case $u_\triangledown + o = 0$ follows from the conditions 2 and 4. We now assume that $u_\triangledown + o > 0$. On X-moves and U-moves, Duplicator uses exactly the Etessami-Wilke strategy from Claim 1 of Lemma 2.5.

Comparing the condition of Claim 1 in Lemma 2.5 and Subclaim 2.1.2 one can see that Conditions 1-4 of both claims are exactly the same. The words in Claim 1 are prefixes of the words obtained by appending a "b" at the beginning of the two paths being considered here – the two paths have additional $(c^*a)^*c^*b$ suffixes at the end.

On X- and U-moves, Duplicator uses exactly the Etessami-Wilke strategy of Claim 1. We have shown in Claim 1 that this strategy preserves Conditions 1-4 of Subclaim 2.1.2. Conditions 5-6 are preserved by X- and U-moves because if Spoiler moves his position a certain number of stems downwards Duplicator will move her position the same number of stems downwards. On F-moves Duplicator uses her strategy for F-moves from the finite Etessami-Wilke game, with the modification that she jumps to the next stem on a similar level instead of the very next stem. As Conditions 1-4 of Subclaim 2.1.2 only talk about properties within a stem, Claim 1 already proves that these are preserved during F-moves. Condition 5 is preserved because Duplicator jumps to a similar level, while Condition 6 is preserved because similar levels are at most $\lambda \cdot (u + o)$ stems apart.

In order to show that all intermediate positions of the game are winning for the appropriate $\mathrm{CTL}^*_{\triangleleft\triangleright}$ game, it suffices to show that they satisfy the conditions of Claim 2. Conditions 1,2, and 3 of Claim 2 follow directly from the conditions in Subclaim 2.1.2. Condition 2 of Claim 2 follows from Condition 6 of Subclaim 2.1.2 and the fact that m=$\lambda(u+o)+1$. □

Case 2. Downward move on $\tilde{S}$. This case is very similar to Case 1. The only difference is that if Spoiler plays a long move, then Duplicator's "fooling" move is to play a path that departs from the witness path on the lower fooling node on the second stem.
More specifically, the strategy on short downward moves is exactly the same as in $\tilde{T}$. This works because this move involves no "fooling" - Duplicator exits from the witness path on the same node as Spoiler does.
If Spoiler plays a long move then Duplicator is forced to perform his fooling move. In this fooling move Duplicator can depart from the stem in a slightly different position than Spoiler, but she must make sure that she exits from the witness path into a subtree that has the same polarity asrthe root of the stem on the witness path in the other tree. If Spoiler plays in $\tilde{(S)}$, then roots of stems on the witness paths have positive polarity - hence Duplicator uses the upper fooling node. As roots of stems on the witness path have negative polarity on $\tilde{(T)}$, Duplicator must perform a fooling move that exits from the witness path into a subtree of negative polarity. Thus she chooses the lower fooling node (see Figure 6).
□

4.5. The Finite Case

Part (ii) of Lemma 4.1 can also be shown for finite trees. We define trees $\tilde{T}^{\mathrm{fin}}$ and $\tilde{S}^{\mathrm{fin}}$ to be obtained from $\tilde{T}$ and $\tilde{S}$ by pruning the stems at some point. In particular, each stem in $\tilde{T}^{\mathrm{fin}}$ and $\tilde{S}^{\mathrm{fin}}$ spells out the finite word $b(c^*a)^{\tilde{o}(\tilde{u}+\tilde{o})+\tilde{u}}$. The witness node and the fooling nodes are as in $\tilde{T}$ and $\tilde{S}$.

The *stem-depth* of a node x is the number of a labelled nodes on a downward fullpath rooted at x that contains no right siblings. The stem-depth of a path π is the stem-depth of its root.

Lemma 4.1 for finite trees trees follows from the following claim. It is basically the finite tree version of Claim 2 in the proof of Lemma 4.1.

CLAIM 3. *Let $p_\triangleleft, p_\triangledown, p_\triangleright \le \tilde{o}$. Duplicator has a winning strategy for the $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \le p_\triangleleft, \mathrm{pd}_\triangledown \le p_\triangledown, \mathrm{pd}_\triangleright \le p_\triangleright, \mathrm{ud}_\triangledown \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$-game on $\tilde{T}^{\mathrm{fin}}, \tilde{S}^{\mathrm{fin}}$ if the selected nodes x, y satisfy*
(1) x, y have the same label,
(2) x, y have the same plateau-depth,
(3) x, y are on similar levels,
(4) $\mathrm{level}(x) > k$ and $\mathrm{level}(y) > k$ where $k = (\lambda(\tilde{u}_\triangledown + \tilde{o})+1)(p_\triangleleft + p_\triangledown + p_\triangleright) + \mu$,
(5) $\mathrm{stem\text{-}depth}(x) \ge p_\triangledown(\tilde{u}_\triangledown + \tilde{o})$ and $\mathrm{stem\text{-}depth}(y) \ge p_\triangledown(\tilde{u}_\triangledown + \tilde{o})$.

Proof of Claim 3. Duplicator can use a similar strategy as in the infinite case. In particular, whenever Spoiler picks a path that departs from the stem, then Duplicator can use the strategy described in Claim 2 of Lemma 4.1 – using this strategy she can preserve that conditions of Claim 3. But Spoiler might try to expose that the stem depths of x and y are different. To do so, he might select the path π that consists only only of the stem. In this case,

Duplicator will also choose the path τ in the other tree that only consists of the stem. The following Subclaim shows that in this case Duplicator can win the LTL$(\mathrm{ud} \le \tilde{u}_\triangledown, \mathrm{ned} \le \tilde{o})$-game on π and τ, while maintaining the conditions of Claim 3. The proof of Subclaim 1 concludes the proof of Claim 3, and hence the proof of Theorem 1.3 for finite trees.

SUBCLAIM 1. *Let $u_\triangledown \le \tilde{u}_\triangledown$ and $o \le \tilde{o}$. Duplicator can win the LTL$(\mathrm{ud} \le u, \mathrm{ned} \le o)$-game on π and τ if the selected suffixes $\tilde{\pi}$ and $\tilde{\tau}$ of π and τ satisfy:*

(1) *the roots of $\tilde{\pi}$ and $\tilde{\tau}$ have the same label.*
(2) *$\tilde{\pi}$ and $\tilde{\tau}$ have the same plateau-depth.*
(3) $|\text{top-depth}(\tilde{\pi}) - \text{top-depth}(\tilde{\tau})| \le 1$.
(4) *If $\tilde{\pi}$ and $\tilde{\pi}$ have different top-depths then*
  (a) $\text{top-depth}(\tilde{\pi}) \ge (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$ *and* $\text{top-depth}(\tilde{\tau}) \ge (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$
  (b) *if* $\text{top-depth}(\tilde{\pi}) = (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$ *or* $\text{top-depth}(\tilde{\tau}) = (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$
  then $\text{plateau-depth}(\tilde{\pi}) \ge (p_\triangledown - 1)\tilde{o} + o$
  (and hence $\text{plateau-depth}(\tilde{\tau}) \ge (p_\triangledown - 1)\tilde{o} + o$*).*
(5) *If $\tilde{\pi}$ and $\tilde{\pi}$ have different bottom-depths then*
  (a) $\text{bottom-depth}(\tilde{\pi}) \ge (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$
  and $\text{bottom-depth}(\tilde{\tau}) \ge (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$
  (b) *if* $\text{bottom-depth}(\tilde{\pi}) = (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$
  or $\text{bottom-depth}(\tilde{\tau}) = (p_\triangledown - 1)\tilde{u}_\triangledown + u_\triangledown$
  then $\text{inverse-plateau-depth}(\tilde{\pi}) \ge (p_\triangledown - 1)\tilde{o} + o$
  (and hence $\text{inverse-plateau-depth}(\tilde{\tau}) \ge (p_\triangledown - 1)\tilde{o} + o$*).*
(6) $\text{stem-depth}(x) \ge p_\triangledown(\tilde{u}_\triangledown + \tilde{o})$ *and* $\text{stem-depth}(y) \ge p_\triangledown(\tilde{u}_\triangledown + \tilde{o})$.

In addition every intermediate position of the LTL$(\mathrm{ud} \le u, \mathrm{ned} \le o)$-game on π and τ is a winning position for Duplicator for the CTL$^*_{\triangleleft\triangleright}(\mathrm{pd}_\triangleleft \le p_\triangleleft, \mathrm{pd}_\triangledown \le p_\triangledown, \mathrm{pd}_\triangleright \le p_\triangleright, \mathrm{ud}_\triangledown \le \tilde{u}_\triangledown; \mathrm{other} \le \tilde{o})$-game on $\tilde{T}^{\mathrm{fin}}, \tilde{S}^{\mathrm{fin}}$.

Proof of Subclaim 1. The proof is similar to the proof of Claim 1 in Lemma 2.5. We use an induction on $u_\triangledown + o$. If $u_\triangledown + o = 0$ then the claim follows as by Condition 1 the roots of $\tilde{\pi}$ and $\tilde{\tau}$ have the same label. For the inductive case we assume that $u_\triangledown + o > 0$. Duplicator's strategy depends on the kind of move that Spoiler plays first.

X-move. On X-moves Duplicator's strategy is determined by the rules of the game. We omit the calculations that show that X-moves preserve the invariant.

F-move. Assume that Spoiler plays an F-move on $\tilde{\pi}$, in which he selects a suffix $\tilde{\pi}'$ of $\tilde{\pi}$. If $\tilde{\tau}$ contains a suffix $\tilde{\tau}'$ that is isomorphic to $\tilde{\pi}'$ then Duplicator selects $\tilde{\tau}'$. Otherwise $\tilde{\pi}'$ must contain $\tilde{\tau}$ as a suffix. In this case it is Duplicator's goal to jump the fewest number of positions while still preserving the invariant. Hence she selects the largest suffix $\tilde{\tau}'$ of $\tilde{\tau}$ such that $\tilde{\pi}'$ and $\tilde{\tau}'$ have the same plateau-depth and top-depth. Again it is easy to verify that the conditions of Claim 3 are preserved.

U-moves. Finally assume that Spoiler plays an U-move. In the first half-move, Duplicator's strategy is similar to her strategy on F-moves. However, as in the proof of Claim 1 in Lemma 2.5, Duplicator must worry that Spoiler plays a very small move on the path with the smaller top-depth. If Duplicator were to use the strategy from

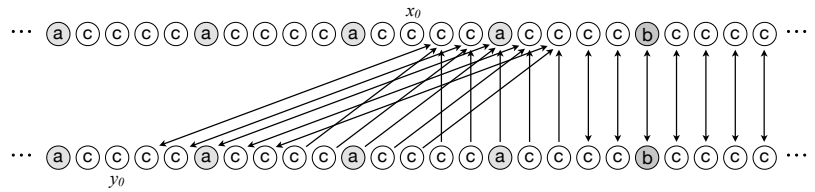


Fig. 8. Duplicator's strategy if Spoiler plays an U-move. If the current position of the game is (x_0, y_0) and Spoiler picks a position z in either path in his first half move, then Duplicator picks the position z' in the other word, such that there is a arrow from z to z'.

the F-move then she might skip more positions than Spoiler did in his first half-move, and in the second half-move she might not have a suitable position to jump to (see the description of Duplicator's U-move in Claim 1 in Lemma 2.5 for the discussion of this problem). Hence Duplicator will skip as many positions as Spoiler did if Spoiler skips at most $\tilde{o} + 1$ positions. Otherwise she will follow her strategy for F-moves in the first half-move. It is easy to see that in the second half-move, Duplicator can always find a position, such that the conditions of Claim 3 are maintained. Duplicator's strategy is shown in Figure 8. □

This concludes the proof of Claim 3, and hence the proof of Theorem 1.3 for finite trees. □

5. CONCLUSIONS AND FUTURE WORK

In this work we have investigated what direction-restricted XML query languages can be first-order complete. We began with the language $\mathrm{CTL}^*_{\triangleleft\triangleright}$ based on the whole of LTL going downwards and sideways, and we have shown that one cannot make due either with no untils in one of the horizontal directions or with a restriction on the number of untils vertically.

In future work we intend to characterise the precise expressiveness of the languages $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} = k)$, in terms of fragments of first-order logic. We also need to investigate more thoroughly the relationship of the languages $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} = k)$ with the queries of "bounded operator depth" mentioned by Bojańczyk [Bojańczyk 2008]. For the moment we note the following distinction: [Bojańczyk 2008] states that the queries of bounded operator depth cannot capture all languages of the form:

$$Q_n := \exists_{\triangledown}(a^n b)^*$$

In contrast, all these Q_n are contained at the lowest level of our hierarchy.

Thérien and Wilke [Thérien and Wilke 2004] have given an algebraic characterisation of the LTL formulas of fixed until-depth on words, and have used this to show how to decide whether a formula is of a given until-depth. We do not know whether one can decide membership in $\mathrm{CTL}^*_{\triangleleft\triangleright}(\mathrm{ud} = k)$ (or in $\mathrm{CTL}^*(\mathrm{ud} = k)$).

Acknowledgements. We thank the ICDT referees for invaluable comments on the submission. We also thank Mikołaj Bojańczyk for many suggestions and corrections, and for providing the construction that underlies Theorem 1.3.

Benedikt is supported in part by EPSRC EP/H017690/1 and EP/G004021/1 (the Engineering and Physical Sciences Research Council, UK)

We also acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the

European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599.

REFERENCES

BARCELO, P. 2005. Temporal logics over unranked trees. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, Washington, DC, USA, 31–40.

BENEDIKT, M. AND JEFFREY, A. 2007. Efficient and expressive tree filters. In *Proceedings of the 27th international conference on Foundations of software technology and theoretical computer science*. FSTTCS'07. Springer-Verlag, Berlin, Heidelberg, 461–472.

BOJAŃCZYK, M. 2008. Effective characterizations of tree logics. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '08. ACM, New York, NY, USA, 53–66.

EMERSON, E. A. 1990. Temporal and modal logic. In *Handbook of theoretical computer science (vol. B)*, J. van Leeuwen, Ed. MIT Press, Cambridge, MA, USA, 995–1072.

ETESSAMI, K. AND WILKE, T. 2000. An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. *Information and Computation 160*, 88–108.

GABBAY, D., PNUELI, A., SHELAH, S., AND STAVI, J. 1980. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. POPL '80. ACM, New York, NY, USA, 163–173.

HAFER, T. AND THOMAS, W. 1987. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *Proceedings of the 14th International Colloquium, on Automata, Languages and Programming*. ICALP '87. Springer-Verlag, London, UK, 269–279.

KAMP, H. 1968. Tense logic and the theory of linear order. Ph.D. thesis, University of California, Los Angeles.

LEY, C. AND BENEDIKT, M. 2009. How big must complete xml query languages be? In *Proceedings of the 12th International Conference on Database Theory*. ICDT '09. ACM, New York, NY, USA, 183–200.

LIBKIN, L. 2004. *Elements of Finite Model Theory*. Springer, Heidelberg.

LIBKIN, L. AND SIRANGELO, C. 2008. Reasoning about XML with temporal logics and automata. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*. LPAR '08. Springer-Verlag, Berlin, Heidelberg, 97–112.

MARX, M. 2004. Conditional XPath, the first-order complete XPath dialect. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '04. ACM, New York, NY, USA, 13–22.

MARX, M. 2005a. Conditional XPath. *ACM Trans. Database Syst. 30*, 929–959.

MARX, M. 2005b. First-order paths in ordered trees. In *International Conference on Database Theory*, T. Eiter and L. Libkin, Eds. Vol. 3363. Springer Verlag, Heidelberg, 114–128.

MARX, M. AND DE RIJKE, M. 2005. Semantic characterizations of navigational XPath. *SIGMOD Rec. 34*, 41–46.

MOLLER, F. AND RABINOVICH, A. 1999. On the expressive power of ctl. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*. LICS '99. IEEE Computer Society, Washington, DC, USA, 360–369.

POTTHOFF, A. 1995. First-order logic on finite trees. In *Proceedings of the 6th International Joint Conference CAAP/FASE on Theory and Practice of Software Development*. TAPSOFT '95. Springer-Verlag, London, UK, 125–139.

RABINOVICH, A. 2008. Personal communciation.

RABINOVICH, A. M. 2002. Expressive power of temporal logics. In *Proceedings of the 13th International Conference on Concurrency Theory*. CONCUR '02. Springer-Verlag, London, UK, 57–75.

RABINOVICH, A. M. AND MAOZ, S. 2000. Why so many temporal logics climb up the trees? In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science*. MFCS '00. Springer-Verlag, London, UK, 629–639.

THÉRIEN, D. AND WILKE, T. 2004. Nesting until and since in linear temporal logic. *Theory of Computing Systems 37*, 111–131. 10.1007/s00224-003-1109-3.

WORLD WIDE WEB CONSORTIUM. 1999. XML Path Language (XPath) Recommendation. http://www.w3c.org/tr/xpath.