How Big Must Complete XML Query Languages Be?

Clemens Ley Michael Benedikt

Computing Laboratory
Oxford University
Parks Rd, Oxford OX13QD, UK
{clemens.ley, michael.benedikt}@comlab.ox.ac.uk

ABSTRACT

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 XML Boolean queries. 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.

Introduction

XPath [21] is the w3c standard for querying XML documents; the navigational core of XPath is a query language on finite labeled ordered trees. Marx and de Rijke [14] 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 [13] that such closure for an extension of XPath is equivalent to first-order completeness.

How can one extend XPath to get a first-order complete language? One answer is given by Marx in [12], who defines a first-order complete query language *Conditional XPath* (CXPath). Roughly speaking, CXPath extends XPath by an "until operator" $\phi \,\mathsf{U}\, \psi$: 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 has the 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 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 ForCXPath complete for arbitrary ordered trees? It follows from results of [2] that this language is first-order complete over finite ordered trees of any fixed depth. In this paper, we determine whether ForCXPath is first-order complete over arbitrary ordered trees.

There is an alternative approach to obtain a first-order complete language on ordered trees, building on the existing work for word and (unordered) tree languages. On words, first order complete languages can be defined using Linear Temporal Logic (LTL). LTL 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)[6]: $\varphi \,\mathsf{U}\, \psi$ 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

ICDT 2009, March 23–25, 2009, Saint Petersburg, Russia.

properly containing β. A refinement of Kamp's Theorem [8] shows that LTL is first-order complete over words.

This can be lifted to the setting of finite ordered trees by considering a variant of the temporal logic CTL* [4]. 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 labelled 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 [7]. Related results on completeness up to bisimulation equivalence can be found in [15]. 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 [1]). Indeed, we will consider a variant CTL$^{*\leftrightarrow}$ 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. CTL$^{*\leftrightarrow}$ 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. 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 fragment F of LTL, we let CTL$^{*\leftrightarrow}(F)$ be the ordered tree language that restricts CTL$^{*\leftrightarrow}$ 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 formulas of F. For example, the language ForCXPath is contained in CTL$^{*\leftrightarrow}(F)$ where F contains only the formula $p\,\mathsf{U}\,q$. Our question can now be formalised as: for which fragments F is CTL$^{*\leftrightarrow}(F)$ first-order complete?

We show that if F is any fragment of LTL with bounded "Until-Depth", then CTL$^{*\leftrightarrow}(F)$ is not first-order complete. In fact, our results show that languages CTL$^{*\leftrightarrow}(F_n)$ where 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 [19]. They consider languages of the form CTL$^*(F)$ where F is a fragment of first-order logic on ω-words. The main result of [19] is that 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 [17] an extension is announced, stating that CTL$^*(AD_n)$ is incomplete, where AD_n is the set of formulas of bounded quantifier-alternation depth.

There has been work also in the ordered case. Barcelo and Libkin consider several logics on ordered trees; 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 [3], 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 [3] 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 [5]. 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 [5] 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 CTL$^{*\leftrightarrow}$ where the use of U is limited in one of the horizontal directions is incomplete. This contradicts an earlier claim from Barcelo and Libkin ([1], again Theorem 3.4: the contradiction is only with the "Moreover" addendum).

Our second main result is that the languages CTL$^{*\leftrightarrow}(UD_k)$ 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 The proof technique used 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 CTL$^*(UD_k)$ are complete on unordered trees. This in turn implies the incompleteness theorem of [19], 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 [18] that formulas of Until-depth k have bounded alternation-depth. Combining this with the result on incompleteness of bounded alternation-depth claimed in [17] implies the restriction of our result to unordered trees. When restricted to unordered trees, our results are incomparable to those announced by Bojańczyk in [3], since each of our sets CTL$^{*\leftrightarrow}(UD_i)$ contains formulas of unbounded Operator Depth.

We note that the case of ordered trees does present new difficulties, as explained in Section 4.

In [11], Marx claims a "separation theorem", stating that

CondXPath formulas can be split as a Boolean combination of pure future and pure past parts – this would be a tree analog of the separation theorem of Gabbay for temporal logics on words [6]. In particular, this would imply that ForCXPath is first-order complete for nodes at the root (i.e. for Boolean queries). The argument in [11] has a flaw, and indeed our main result disproves this claim. This flaw does not impact the main results of Marx in [11], which are reproven by other means in [12].

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 2.1 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. Remaining details are given in the appendix.

1. FIRST ORDER COMPLETE QUERY LANGUAGES

Let Σ be a finite alphabet of labels. An *ordered tree* consists of: a parent/child relation that is acyclic and such that every node having in-degree at most one and a unique node having in-degree 0; a right-sibling relation which is the successor relation of a linear order on the children of any given node; and a labelling function assigning elements 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 will only deal with ordered trees where each node has a finite number of ancestors.

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

$$\phi = a \mid \neg\phi \mid \phi \wedge \phi \mid \mathsf{X}\,\phi \mid \mathsf{F}\,\phi \mid \phi\,\mathsf{U}\,\phi$$

where $a \in \mathsf{Prop}$.

The semantics of LTL is generally given with respect to infinite words. We will give a variant for a finite labelled linear order: that is, a finite linear order $(D, <)$ with a labelling function $\mathsf{lab}()$ that maps each node of the order to a subset of Prop, where Prop is a finite set of propositions. For a labelled linear order σ, we let $|\sigma|$ be the number of nodes in σ. For $i \leq |\sigma|$ we let $\sigma(i)$ be the i^{th} node in σ and σ^i the labelled linear order induced by restricting σ to the subset $\sigma(i) \ldots \sigma(|\sigma|)$.

Then we define:

- $\sigma \models a$ iff $a \in \mathsf{lab}(\sigma(1))$.
- $\sigma \models \mathsf{X}\,\phi$ iff $|\sigma| > 1$ and $\sigma^2 \models \phi$.
- $\sigma \models \mathsf{F}\,\phi$ iff there is a j with $1 < j \leq |\sigma|$ and $\sigma^j \models \phi$.
- $\sigma \models \phi\,\mathsf{U}\,\phi'$ iff there is a j with $j \leq |\sigma|$ and $\sigma^j \models \phi'$ and for all j', if $1 < j' < j$ then $\sigma^{j'} \models \phi$.

Here we use the "strong variant" of U, in which $\phi\,\mathsf{U}\,\phi'$ asserts the existence of a node satisfying ϕ'. 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 below the first such satisfy ϕ [4]. Using this variant will make our negative results stronger, but will not impact our positive results.

We define the *Until-Depth* $\mathsf{ud}(\phi)$ of an LTL formula ϕ by induction on the Path-Depth;

$$\begin{aligned}
&\mathsf{ud}(a) = 0\\
&\mathsf{ud}(\phi \wedge \phi') = max\{\mathsf{ud}(\phi), \mathsf{ud}(\phi')\}\\
&\mathsf{ud}(\neg\phi) = \mathsf{ud}(\mathsf{F}\,\phi) = \mathsf{ud}(\mathsf{X}\,\phi) = \mathsf{ud}(\phi)\\
&\mathsf{ud}(\phi\,\mathsf{U}\,\phi') = max\{\mathsf{ud}(\phi), \mathsf{ud}(\phi')\} + 1
\end{aligned}$$

This is precisely the definition of [5], restricted to LTL formulas with only future operators. Note that there are infinitely many formulas of any fixed Until-Depth.

We define the Next/Eventually-depth ned of an LTL formula similarly:

$$\begin{aligned}
&\mathsf{ned}(a) = 0\\
&\mathsf{ned}(\phi \wedge \phi') = \mathsf{ned}(\phi\,\mathsf{U}\,\phi') = max\{\mathsf{ned}(\phi), \mathsf{ned}(\phi')\}\\
&\mathsf{ned}(\neg\phi) = \mathsf{ned}(\phi)\\
&\mathsf{ned}(\mathsf{F}\,\phi) = \mathsf{ned}(\mathsf{X}\,\phi) = \mathsf{ned}(\phi) + 1
\end{aligned}$$

A *downward path* in a finite ordered tree is a set of nodes that is linearly-ordered by the child relation of the tree. A *downward fullpath* is a downward path that contains a leaf node. Similarly, a *rightward path* is a set of nodes linearly-ordered by the right-sibling relation of the tree, 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 Σ a set of labels, and $k \geq 0$, we inductively define the sets P_k and N_k of $\mathrm{CTL}^{*\leftrightarrow}$ *path formulas* and *node formulas*.

- The symbols in Σ are the only formulas in N_0.
- Any LTL formula over propositions in N_k is in P_k.
- If $\phi \in P_k$ then $\exists_\downarrow\phi$, $\exists_\leftarrow\phi$, and $\exists_\rightarrow\phi$ are in N_{k+1}.
- Both N_k and P_k are closed under Boolean operations.

$\mathrm{CTL}^{*\leftrightarrow}$ is the union over k of the formulas in N_k and P_k.

The semantics of $\mathrm{CTL}^{*\leftrightarrow}$ is defined by induction: $t, \alpha \models a$ iff α is labelled with a in t. $t, p \models \phi$, for $\phi \in P_k$ iff $N_k(p) \models \phi$, where $N_k(p)$ is the labelled linear order formed by labelling each node with the formulas of N_k that it satisfies in t. $t, \alpha \models \exists_\downarrow\phi$ iff there is a downward fullpath p starting at α such that $t, p \models \phi$, and similarly for $\exists_\leftarrow$ and $\exists_\rightarrow$. For a node formula ϕ, we write $t \models \phi$ iff $t, \mathrm{root}(t) \models \phi$.

We can also consider the language $\mathrm{CTL}^{*\leftrightarrow\uparrow}$ formed by allowing a quantifier $\exists_\uparrow$; that is, extending the rules for node formulas to say that if $\phi \in P_k$ then $\exists_\uparrow\phi$ in N_{k+1}. The semantics is analogous to that of $\mathrm{CTL}^{*\leftrightarrow}$, but the quantifier $\exists_\uparrow$ quantifies over *upward fullpaths*, originating at a given node and extending to the root, with the linear ordering being the

ancestor ordering rather than the descendant ordering. We denote by $\mathrm{CTL}^{*\leftrightarrow\uparrow}_{\mathsf{ud}=1}$ the set of $\mathrm{CTL}^{*\leftrightarrow\uparrow}$ with Until-Depth 1.

The *Down-Closure* $\mathsf{dcl}(\phi)$ of a $\mathrm{CTL}^{*\leftrightarrow}$ formula is the set of LTL formulas that are directly nested within downward path quantification. That is

$$\begin{aligned}
&\mathsf{dcl}(a) = \emptyset\\
&\mathsf{dcl}(\phi \wedge \phi') = \mathsf{dcl}(\phi\,\mathsf{U}\,\phi') = \mathsf{dcl}(\phi) \cup \mathsf{dcl}(\phi')\\
&\mathsf{dcl}(\neg\phi) = \mathsf{dcl}(\,\mathsf{F}\,\phi) = \mathsf{dcl}(\,\mathsf{X}\,\phi) = \mathsf{dcl}(\phi)\\
&\mathsf{dcl}(\exists_{\leftarrow}\phi) = \mathsf{dcl}(\exists_{\rightarrow}\phi) = \mathsf{dcl}(\phi)\\
&\mathsf{dcl}(\exists_{\downarrow}\phi) = \mathsf{dcl}(\phi) \cup \{\phi\}
\end{aligned}$$

The *Left-Closure* $\mathsf{lcl}(\phi)$ and *Right-Closure* $\mathsf{rcl}(\phi)$ of ϕ are defined similar for leftward and rightward quantification. The *Closure* $\mathsf{cl}(\phi)$ of ϕ is the union of $\mathsf{dcl}(\phi)$, $\mathsf{lcl}(\phi)$, and $\mathsf{rcl}(\phi)$.

The *Next/Eventually-depth* $\mathsf{ned}(\phi)$ of ϕ is the maximal ned of an element in the closure of ϕ. The *Down-Until-Depth* $\mathsf{dud}(\phi)$ of ϕ is the maximal Until-Depth of any element within the Down-Closure of ϕ. *Left-Until-Depth* $\mathsf{lud}(\phi)$ and *Right-Until-Depth* $\mathsf{rud}(\phi)$ are defined alike with respect to the Left- and Right-Closure.

The *Down-Path-Depth* $\mathsf{dpd}(\phi)$ is the nesting depth of $\exists_{\downarrow}$-quantifiers within ϕ. Formally

$$\begin{aligned}
&\mathsf{dpd}(a) = \emptyset\\
&\mathsf{dpd}(\phi \wedge \phi') = \mathsf{dpd}(\phi\,\mathsf{U}\,\phi') = \mathsf{dpd}(\phi) \cup \mathsf{dpd}(\phi')\\
&\mathsf{dpd}(\neg\phi) = \mathsf{dpd}(\,\mathsf{F}\,\phi) = \mathsf{dpd}(\,\mathsf{X}\,\phi) = \mathsf{dpd}(\phi)\\
&\mathsf{dpd}(\exists_{\leftarrow}\phi) = \mathsf{dpd}(\exists_{\rightarrow}\phi) = \mathsf{dpd}(\phi)\\
&\mathsf{dpd}(\exists_{\downarrow}\phi) = \mathsf{dpd}(\phi) + 1
\end{aligned}$$

The *Left-Path-Depth* $\mathsf{lpd}(\phi)$ and the *Right-Path-Depth* $\mathsf{rpd}(\phi)$ are defined in the obvious way. For example the formula

$$\exists_{\downarrow}((\exists_{\leftarrow}(\mathsf{a}\,\mathsf{U}\,\mathsf{b}))\,\mathsf{U}\,\mathsf{d}).$$

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

Let $D = \{\mathsf{lpd}, \mathsf{dpd}, \mathsf{rpd}, \mathsf{lud}, \mathsf{dud}, \mathsf{rud}, \mathsf{ned}\}$. For $d_1, \ldots, d_n \in D$ we denote by $\mathrm{CTL}^{*\leftrightarrow}(d_1 \leq x_1, \ldots, d_n < x_n)$ the set of $\mathrm{CTL}^{*\leftrightarrow}$ formulas ϕ with $d_1(\phi) \leq x_1, \ldots, d_n(\phi) \leq x_n$. In addition we use the shorthand notation $\mathrm{CTL}^{*\leftrightarrow}(d_1 \leq x_1, \ldots, d_n < x_n, \mathsf{other} \leq o)$ to denote the set of $\mathrm{CTL}^{*\leftrightarrow}$ formulas ϕ with $d_1(\phi) \leq x_1, \ldots, d_n(\phi) \leq x_n$, and $d(\phi) \leq o$ for all $d \in D \setminus \{d_1, \ldots, d_n\}$.

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 [9]. The *quantifier depth* of a first order formula φ is – as usual – the maximal nesting depth of quantifiers in φ.

It has been shown by Hafer and Thomas that CTL^* – $\mathrm{CTL}^{*\leftrightarrow}$ without sibling axes – is equivalent to first order logic over unordered binary trees [7]. In [15] Moller and Rabinovich show that CTL^* is equivalent to the bisimulation invariant fragment of first order logic over unordered trees. Theorem 3.4 in [1] extends this to ordered trees:

THEOREM 1 (BARCELO AND LIBKIN). *$CTL^{*\leftrightarrow}$ is first-order complete. That is, for every finite alphabet Σ, for every first-order sentence ϕ, there is an $CTL^{*\leftrightarrow}$ query ψ such that for every ordered tree t labelled in Σ, $t \models \phi \leftrightarrow t \models \psi$.*

As noted in the introduction, Theorem 3.4 of [1] actually states a much stronger claim, which our Theorem 5 contradicts. We thus give a brief sketch of the proof. One approach is via the composition technique of Moller and Rabinovich [15]. Indeed, this extension is almost implicit in the work of Moller and Rabinovich and that of Hafer and Thomas.

Fix an alphabet Σ. For a node α in a Σ-labeled tree t, let: $\mathrm{subtree}_k(\alpha)$ be the set of first-order formulas of quantifier rank at most k true at α. Let Σ_k be the label set consisting of a label for each set of first-order formulas of quantifier rank at most k in one variable (say, in prenex normal form with propositionally redundant or inconsistent clauses removed from the quantifier-free part; clearly there are then only finitely many such formulas).

Given a vertical path p in t and a node α that is on p but is not a leaf of p,

- Let $\mathrm{leftchild}_k(p, \alpha)$ be the Σ_k-labelled string $\mathrm{subtree}_k(\alpha_i) : i < n$, where α_0 is the unique child of α on p and $\alpha_i : 1 \leq i < n$ is the sequence of left-siblings of α_0, ordered from right to left, where each node in $\mathrm{leftchild}_k(p, \alpha)$ is labelled with $\mathrm{subtree}_k(\alpha)$.
- Let $\mathrm{leftchild}_k(p, \alpha)$ be defined analogously for right siblings.
- For k, l numbers let $\mathrm{child}_{k,l}(\alpha)$ be the pair consisting of the set of first-order sentences of quantifier rank at most l that hold of $\mathrm{leftchild}_k(p, \alpha)$ and the set of first-order sentences of quantifier rank at most l that hold of $\mathrm{rightchild}_k(p, \alpha)$.
- Let $\Sigma_{k,l}$ be the alphabet with a label for each pair of sets of (normalised) sentences of quantifier rank at most l in the vocabulary for labelled strings over Σ_k, and an additional label $\perp$.
- Let $\mathrm{Type}_{k,l}(p)$ be the $\Sigma_{k,l}$-labelled string expanding p by labelling each interior node α with $child_{k,l}(\alpha)$, and the leaf node of p with a special label $\perp$.

Then we have:

LEMMA 2. *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: $(t, \alpha) \models \phi \leftrightarrow \mathrm{Type}_{k,l}(p) \models \psi$, where p is the path from the root of t to α.*

This is proved as in Theorem 3.2 of [15]. In the presence of an ordering the assumption of "wideness" used there is not needed.

The result then follows from:

LEMMA 3. *For every first-order sentence ϕ over $\Sigma_{k,l}$-labelled strings, there is an CTL$^{*\leftrightarrow}$ formula such that* $\mathrm{Type}_{k,l}(p) \models \phi \leftrightarrow (t, \alpha) \models \phi$

This is proved using Kamp's theorem, as in Lemma 4.2 of [15].

CTL$^{*\leftrightarrow}$ 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 $\mathsf{F}\,\phi$ and the next operator $\mathsf{X}\,\phi$ can both be defined using our form of the until operator [4], and so both are unnecessary. What about the use of the until operator?

The following result follows directly from the work of Marx [11, 12]:

THEOREM 4 (MARX). *CTL$^{*\leftrightarrow\uparrow}_{\mathsf{ud}=1}$ is first-order complete*

PROOF. The Conditional XPath language of [11, 12] has been proved first-order complete in [12]. In [10] it was shown that there is an easy translation from Conditional XPath filters to the language $\mathcal{X}_{\mathrm{until}}$ defined in [11]. $\mathcal{X}_{\mathrm{until}}$ has quantification over partial 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). As $\mathcal{X}_{\mathrm{until}}$ is a subset of CTL$^{*\leftrightarrow\uparrow}_{\mathsf{ud}=1}$, both are first-order complete. □

We thus have two ways of getting first-order completeness: Theorem 1 allows arbitrary LTL, downward paths and both horizontal paths, while Theorem 4 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.

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

THEOREM 5 (HORIZONTAL INCOMPLETENESS). *For all $p, u \in \mathbb{N}$, CTL$^{*\leftrightarrow}(\mathsf{rpd} \le p, \mathsf{rud} \le u)$ is not first-order complete on finite ordered trees. In addition, if $up > u'p'$ then CTL$^{*\leftrightarrow}(\mathsf{rpd} \le p, \mathsf{rud} \le u)$ is more expressive than CTL$^{*\leftrightarrow}(\mathsf{rpd} \le p', \mathsf{rud} \le u')$. The symmetric statements hold for leftward axes.*

It follows from our proof that CTL$^{*\leftrightarrow}(\mathsf{lud} = 0)$is not first order complete. This contradicts a statement made in [1]. It follows from [1] that CTL$^{*\leftrightarrow}(\mathsf{lud} = 1)$ is already first order complete.

Our second incompleteness result is concerned with downward axes. It states CTL$^{*\leftrightarrow}$ 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 6 (VERTICAL INCOMPLETENESS). *On finite, ordered trees, CTL$^{*\leftrightarrow}(\mathsf{dud} \le u)$ is not first-order complete for all $u \in \mathbb{N}$. In addition, the family CTL$^{*\leftrightarrow}(\mathsf{dud} \le u)$ forms a strict hierarchy in expressiveness.*

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

Discussion. It is known from [5] 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 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, CTL$^{*\leftrightarrow}(\mathsf{ud} = 1)$ 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 [4] for the precise definition): Etessami and Wilke [5] 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 CTL$^{*\leftrightarrow}$ based on LTL-with-past but restricted to Since/Until-Depth one is already sufficient for first-order completeness.

Thus the incompleteness of query languages based on fixed Until-Depth when only future operators are available is not obvious. *The proofs of Theorems 5 and 6 are quite complex, and will take up the remainder of the paper.*

2. BACKGROUND: GAMES FOR WORDS AND TREES

In this section we define an Ehrenfeucht-Fraïssé game that corresponds to CTL$^{*\leftrightarrow}$. Then we recall an incompleteness result on strings by Etessami and Wilke.

2.1 Ehrenfeucht-Fraïssé Games for Bounded Until-Depth

Before we define a game that corresponds to CTL$^{*\leftrightarrow}$ on trees, we define a game on paths that is used as a sub-game within the tree game. This game was introduced by Etessami and Wilke in [5] where they showed that it corresponds to LTL.

The Path Game. The LTL$(\mathsf{ned} \le e, \mathsf{ud} \le u)$-game (or LTL$(e, u)$-game for short) is played by two players, a male *Spoiler* and a female *Duplicator*, on two paths p, q. The goal of Spoiler is to show that the paths are different while Duplicator tries to show that they are similar. At each stage of the game a pair p^i, q^j of suffixes of p, q is *selected*. There

are different kinds of moves – X-, F-, and U-moves – in which the players alter the selected paths. When the game is played, we track the number of moves of each kind remaining in the game, which can be thought of as an additional part of the game state. Initially, there are e X- and F-, and u U-moves left to play (denoted (e, u) moves to play). p, q are selected initially.

We describe the play of the LTL(e, u)-game with selected paths p, q and e', u' moves to play, for $e' \leq e$ and $u' \leq u$. If (e', u') is $(0, 0)$ Duplicator wins if the roots of the selected paths have same label and Spoiler wins otherwise. We call p, q the *final position* of the game. If $e' > 0$ Spoiler can chooses to play either an X-, or an F-move.

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

If Duplicator did not lose the round, the players play the LTL(e, u)-game on the newly selected paths with $(e' - 1, u')$ moves to play.

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

- In an U-*move* Spoiler picks one of the paths and we assume he picks p. An U-move consists of two *half moves*. In the first half move the Spoiler selects a proper suffix p^i of p for $i > 1$. The Duplicator has to reply with a proper suffix q^j of q for some $j > 1$ such that the roots of p^i and q^j have the same label. Again, Spoiler can only choose this move if p^i exists and Duplicator looses if the roots of p^i and q^j have different labels.

  In the second half move Spoiler selects $q^{j'}$ such that $1 < j' \leq j$. Duplicator has to select $p^{i'}$ such that $1 < i' \leq i$ and such that the roots of $p^{i'}$ and $q^{j'}$ have the same label. In the case that the Spoiler picks $q^{j'}$ such that $j' = j$, i' must be equal to i. Again, Duplicator looses if she cannot select such a path.

If Duplicator survives both half rounds, the players continue playing the LTL(e, u)-game with $p^{i'}, q^{j'}$ selected with $(e', u' - 1)$ moves left to play.

Etessami and Wilke [5] have shown the following:

Proposition 7 (Etessami, Wilke). *Two paths satisfy the same LTL$(\mathsf{ned} \leq e, \mathsf{ud} \leq u)$ formulas iff the Duplicator has a winning strategy in the LTL(e, u)-game.*

The Tree Game. The CTL$^{*\leftrightarrow}(\mathsf{lpd} \leq l_p, \mathsf{dpd} \leq d_p, \mathsf{rpd} \leq r_p, \mathsf{lud} \leq l_u, \mathsf{dud} \leq d_u, \mathsf{rud} \leq r_u, \mathsf{ned} \leq e)$-game (CTL$^{*\leftrightarrow}(l_p, d_p, r_p, l_u, d_u, r_u, e)$-game for short) is played by Spoiler and Duplicator on a pair of trees t, s. Again there are different kinds of moves – $\exists_{\leftarrow}$-, $\exists_{\rightarrow}$-, and $\exists_{\downarrow}$-moves – in which the players alter a pair of *selected nodes* α, β. Initially, the roots of p, q are selected.

The state of a game consists of the pair of selected nodes α, β and a *move tuple* (l, d, r). The intuition is that there are l left, d down-, and r right-path-moves left to play. Therefore we often say that there are (l, d, r) *moves left*. Initially the roots of t and s are selected and there are (l_p, d_p, r_p) moves left.

We describe the play of the game with α, β selected and (l, d, r) moves left for $l \leq l_p$, $d \leq d_p$, and $r \leq r_p$. The Duplicator wins if the move tuple is $(0, 0, 0)$ and α and β have the same label. Otherwise Spoiler can choose one of the following moves:

- Spoiler can choose an $\exists_{\leftarrow}$-*move* if $l > 0$. He picks one of the two trees t, s, say t. Spoiler then picks a leftward fullpath p that is rooted at α. Duplicator responds by picking a leftward fullpath q that is rooted at β. Then Spoiler and Duplicator play the LTL(e, l_u)-game on p, q. If Spoiler wins this path-game then he wins the tree-game. Otherwise Spoiler chooses the roots α', β' of some intermediate position of the LTL-game. The players play the tree game with $(l - 1, d, r)$ moves left and α', β' selected.
- An $\exists_{\downarrow}$-*move* can be chosen by Spoiler if $d > 0$. It is played like an $\exists_{\leftarrow}$-move, just that the players pick downward fullpaths p, q instead of leftward fullpaths and the players play the LTL(e, d_u)-game on p, q instead of the LTL(e, l_u)-game. Again, Spoiler wins the tree game if he wins the game in p, q. Otherwise the players proceed to play the tree game with $(l, d - 1, r)$ moves left and the roots of some intermediate position of the path game selected.
- Spoiler may play an $\exists_{\rightarrow}$-*move* if $r > 0$. The rules are as above, just with rightward-paths on which the players play the LTL(e, r_u)-game. Again, Spoiler can win the tree game by winning the game on p, q. Otherwise the game proceeds with the tree game on an intermediate position of the path game with $(l, d, r - 1)$ moves left.

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

Proposition 8. *Duplicator has a winning strategy for the CTL$^{*\leftrightarrow}(l_p, d_p, r_p, l_u, d_u, r_u, e)$-game on t, s iff t and s agree on all CTL$^{*\leftrightarrow}(l_p, d_p, r_p, l_u, d_u, r_u, e)$ 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 α, β at a stage with (l, d, r) moves to play, then (t, α) agrees with (s, β) on $\mathrm{CTL}^{*\leftrightarrow}(l_p, d_p, r_p, l_u, d_u, r_u, e)$ node formulas. We show this by upward induction on $l+d+p$. The base case of no moves remaining is clear.

One inductive case is when Spoiler plays an $\exists_\downarrow$ move p By induction, (t, α) satisfies the same formulas of $\mathrm{CTL}^{*\leftrightarrow}(l_p, d_p, r_p, l_u, d_u, r_u, e)$ as (s, β), and hence there is a path q rooted at β such that (t, p) and (s, q) satisfy the same path formulas of $\mathrm{CTL}^{*\leftrightarrow}(l_p, d_p - 1, r_p, l_u, d_u, r_u, e)$. Duplicator respond with such a path. Consider LTL over the alphabet with propositions form $\mathrm{CTL}^{*\leftrightarrow}(l_p, d_p - 1, r_p, l_u, d_u, r_u, e)$ node formulas, and consider the expansion of p and q where nodes are decorated by the formulas in the above language that they satisfy. The hypothesis on p and q and Proposition 7 guarantee that there is a winning strategy for Duplicator in the $LTL(e, d_u)$-game over this alphabet. Duplicator uses this strategy in the remainder of the move.

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

2.2 The Until Hierarchy on Words

The proof of Theorems 5 and 6 makes use of a hierarchy Theorem on words by Etessami and Wilke [5]. As we reuse their proof for our incompleteness results on trees, we reprove this theorem here.

THEOREM 9 (ETESSAMI, WILKE). *$LTL(\mathsf{ud} \leq u)$ is not first order complete over finite words for every $u \geq 0$. In addition, for each u $LTL(\mathsf{ud} \leq u+1)$ is more expressive than $LTL(\mathsf{ud} \leq u)$.*

To show Theorem 9, we define for each $u \in \mathbb{N}$ a property S_u such that S_u that can be expressed in $\mathrm{LTL}(\mathsf{ud} \leq u)$ but not in $\mathrm{LTL}(\mathsf{ud} \leq u - 1)$. Let S_u denote the set of paths that satisfy the regular expression

$$\mathsf{a}(\mathsf{c}^*\mathsf{a})^x\Sigma^*.$$

where Σ is the alphabet.

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

$$\psi_0 := \mathsf{a} \qquad \psi_x := \mathsf{a} \wedge (\mathsf{c}\,\mathsf{U}\,\psi_{x-1})$$

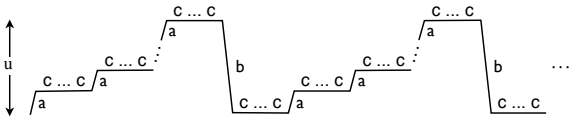
Together with Proposition 8, the following Lemma shows that S_u can not be expressed in $\mathrm{LTL}(\mathsf{ud} \leq u - 1, \mathsf{ned} \leq e)$, even for any nesting depth e of X and F modifiers. This completes the proof of Theorem 9. It simplifies the presentation to show the lemma for infinite paths. We will later show how the proof can be altered for the finite case.

LEMMA 10 (ETESSAMI, WILKE). *For each $u \geq 1, e \geq 0$ there are two infinite paths $v_{u,e}$ and $w_{u,e}$ such that (i) every $v_{u,e}$ satisfies S_u no $w_{u,e}$ does, and (ii) Duplicator has a winning strategy for the $LTL(\mathsf{ud} \leq u - 1, \mathsf{ned} \leq e)$-game on $v_{u,e}, w_{u,e}$.*

For all $u, e \in \mathbb{N}$ we define the infinite paths

$$v_{u,e} = (\mathsf{ac}^e)^{u+1}\mathsf{bc}^e\ w_{u,e}$$
$$w_{u,e} = ((\mathsf{ac}^e)^u\mathsf{bc}^e)^\omega$$

Observe that $w_{u,e}$ can be visualised as the following "staircase".



The following proposition is obvious.

PROPOSITION 11. *For all $u, e \in \mathbb{N}$, $v_{u,e}$ satisfies S_u while $w_{u,e}$ does not.*

We now fix u, e and write v instead of $v_{u,e}$ and w instead of w. It remains to show that Duplicator has a winning strategy for the $\mathrm{LTL}(\mathsf{ud} \leq u-1, \mathsf{ned} \leq e)$-game on v and w.

The idea of the proof will be to show that Spoiler can force the selected nodes up only one step of the staircase on each U-move. As the number of steps depends on the number of U-moves, Spoiler can not detect that the selected nodes are on different steps of the staircase in the beginning of the game.

To describe Duplicator's winning strategy, we need some notation. Give two nodes α, β on the same string we denote by (α, β) the sequence of nodes between α and β, excluding α and β. We denote by $\mathsf{right\text{-}b}(\alpha)$ the next node to the right of α that is labelled b. $\mathsf{right\text{-}ab}(\gamma)$ is the next a or b labelled right sibling of α. The *Plateau-Depth* $\mathsf{pd}(\alpha)$ of a node α in v or w is the distance of α to the end of the its "plateau" to the right. Formally, $\mathsf{pd}(\alpha)$ is the number of cs in $(\alpha, \mathsf{right\text{-}ab}(\alpha))$. The *Top-Depth* $\mathsf{td}(\alpha)$ of α in v or w is the number of steps between α and the top of the staircase, that is $\mathsf{td}(\alpha)$ is the number of a nodes in $(\alpha, \mathsf{right\text{-}b}(\alpha))$. If p is a path the we write $\mathsf{td}(p)$ for $\mathsf{td}(\mathsf{root}(p))$ and similarly for pd. We also write $\alpha =_{\mathsf{pd}} \beta$ if $\mathsf{pd}(\alpha) = \mathsf{pd}(\beta)$ and similar for td.

Part (ii) of Lemma 10 follows from the following claim.

CLAIM 1. *Duplicator can play the $LTL(\mathsf{ud} \leq u-1, \mathsf{ned} \leq e)$ game on v and w so that if there are $e' \leq e$ X- or F-moves and $u' \leq u-1$ U-moves left to play then the roots α, β of the selected paths in v, w maintain the following invariant:*

1. $\alpha =_{\mathsf{pd}} \beta$.
2. $|\mathsf{td}(\alpha) - \mathsf{td}(\beta)| \leq 1$.
3. *If $\alpha \neq_{\mathsf{td}} \beta$ then*
   (a) $\mathsf{td}(\alpha) \geq u'$ *and* $\mathsf{td}(\beta) \geq u'$
   (b) *if $\mathsf{td}(\alpha) = u'$ or $\mathsf{td}(\alpha) = u'$ then $\mathsf{pd}(\alpha) = \mathsf{pd}(\beta) \geq e'$.*

PROOF. On X-moves, Duplicator's strategy is determined by the rules of the game. Basically, condition 3b assures that Spoiler can not use X moves to falsify condition 3a.

The strategy on F-moves is very easy for Duplicator. It relies on the observation that if v, w satisfy the invariant, then v and w have the same set of proper suffixes. Therefore, given Spoilers selection, Duplicator can select the same suffix in the other path.

On U-moves, Duplicator can not use the same strategy as for F-moves in her first half move: if she did, Spoiler might play on the path with the smaller td and just move the selected suffix by one node. Then Duplicator might skip $e+1$ nodes to the next isomorphic suffix. Then in the second half move, Spoiler can pick from $e+2$ nodes (the nodes that Duplicator skipped and the one she selected), but Duplicator can only choose the node that Spoiler selected. Hence, when Spoiler skips only few nodes, Duplicator will skip the same number of nodes. Only when Spoiler skips sufficiently many nodes will Duplicator try to find an isomorphic suffix.

Assume that Spoiler picks a path $p \in \{v, w\}$ and a suffix p' of p in his first half move. Duplicator will respond with a suffix q' of the other path q. Let $\gamma, \gamma', \delta, \delta'$ be the roots of p, p', q, q' respectively. There are two cases. If Spoiler skips at most e nodes (that is $|(\gamma, \gamma')| \leq e$), then Duplicator skips the same number of nodes as Spoiler did. That is, she picks δ' such that $(\gamma, \gamma') = (\delta, \delta')$. Otherwise Duplicator picks the *next* q' such that p' and q' are isomorphic (formally $(\gamma', \mathsf{right\text{-}b}(\gamma')) = (\delta', \mathsf{right\text{-}b}(\delta'))$ and $|(\gamma, \gamma') - (\delta, \delta')| \leq e + 1$). In the second half move Spoiler chooses some δ'' in (δ', δ''). It is easy to check that Duplicator can always choose γ'' such that $|(\gamma', \gamma'') - (\delta', \delta'')| \leq o + 1$.

Observe that if p' and q' are not isomorphic, then the roots of the selected paths have been moved to the right at most $e + 1$ nodes. It follows that the invariant is maintained. □

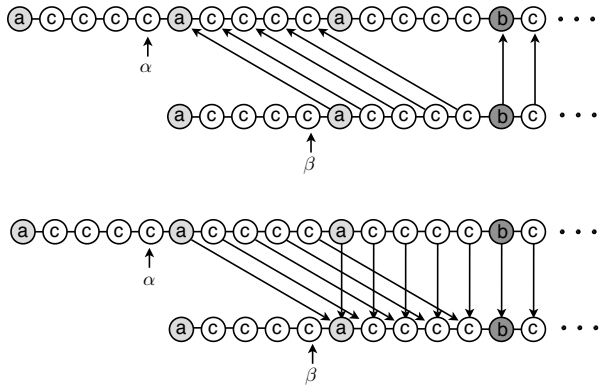


Figure 1: Duplicator's strategy on U-moves on $v_{2,4}$ and $w_{2,4}$.

Figure 1 shows the position of the LTL($\mathsf{ud} \leq 1, \mathsf{ned} \leq 4$)-game on $v_{2,4}, w_{2,4}$ after four X-moves. The first and the third path shown are $v_{2,4}$, while the second and fourth path are prefixes of $w_{2,4}$. The roots of the selected paths are marked with α and β. The arrows indicate Duplicator's strategy on U-moves: if Spoiler picks a node γ' in either $v_{2,4}$ or $w_{2,4}$ in his first half move, then Duplicator picks the node δ' in the other path, such that there is a arrow from γ' to δ' in Figure 1. Observe that the position α, β is a winning position for Duplicator if there are no X-moves, and at most one U-move left to play.

We now turn to the finite case. How must the construction be altered to show the result for finite words? If Duplicator only jumps to the *next* $(\mathsf{ac}^e)^u\mathsf{bc}^e$ section on each eventually move, then $e + 1$ such sections are obviously sufficient for Duplicator to win the game. Hence

COROLLARY 12 (ETESSAMI, WILKE). *Duplicator has a winnig strategy for the LTL($\mathsf{ud} \leq u - 1, \mathsf{ned} \leq e$)-game on the finite words $v^{\mathit{fin}}_{u,e}$ and $w^{\mathit{fin}}_{u,e}$ defined as*

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

3. THE HORIZONTAL UNTIL HIERARCHY

In this section we show Theorem 5. The argument is based on a construction of Mikołaj Bojańczyk. We define a property Q_x such that CTL$^{*\leftrightarrow}(\mathsf{lpd} \leq p, \mathsf{lud} \leq u)$ can express Q_{pu} but not Q_{pu+1}.

Let the *right path* of a node α in a tree be the sequence of its right siblings. Q_x is the set of ordered trees that have a full path p ending at a leaf labelled d, and each node on p apart from the root and the leaf has a right path that satisfies the regular expression

$$\mathsf{a}(\mathsf{c}^*\mathsf{a})^x\Sigma^*$$

where Σ is the alphabet.

If $pu \geq x$ then Q_x can be expressed in CTL$^{*\leftrightarrow}(\mathsf{lpd} \leq p, \mathsf{lud} \leq u)$ by $\exists_\downarrow(\mu_x\, \mathsf{U}\, (\mathsf{d} \wedge \mathsf{X}\,\mathsf{false}))$ where μ_x says "there is a rightward path satisfying $\mathsf{a}(\mathsf{c}^*\mathsf{a})^x\Sigma^*$". Formally

$$\begin{aligned}
\mu_0 &= \mathsf{a} \\
\mu_y &= \exists_\rightarrow(\lambda_{u,y-1}) && \text{if } y > 0 \\
\lambda_{0,y} &= \mu_y \\
\lambda_{x,y} &= \mathsf{a} \wedge (\mathsf{c}\,\mathsf{U}\,\lambda_{x-1,y}) && \text{if } x > 0
\end{aligned}$$

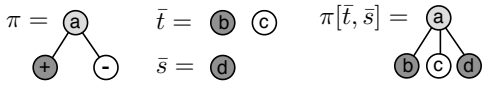
The following lemma shows that CTL$^{*\leftrightarrow}(\mathsf{rpd} \leq p, \mathsf{rud} \leq u)$ can not express Q_{x+1} if $pu \leq x$. Theorem 5 immediately follows from this lemma.

LEMMA 13. *For each $p, u, o \in \mathbb{N}$, there are trees $t_{p,u,o}$ and $s_{p,u,o}$ such that (i) $t_{p,u,o}$ satisfies Q_{pu+1} but $s_{p,u,o}$ does not and (ii) Duplicator has a winning strategy for the CTL$^{*\leftrightarrow}(\mathsf{rpd} \leq p, \mathsf{rud} \leq u; \mathsf{other} \leq o)$-game on $t_{p,u,o}, s_{p,u,o}$.*

We begin by showing 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 $\omega^* + \omega$). To show the lemma by induction we show a more general statement

for *hedges* – ordered sequences of wide trees, where the sequence can again be infinite in both directions. Later we will explain how the proof must be altered for the finite case.

To construct $t_{p,u,o}$ and $s_{p,u,o}$, we need some machinery. A *template* π is a hedge with two sets of distinguished nodes – *positive ports* and *negative ports* – labelled by "+" and "−" respectively. A template π and two hedges $\bar{t}, \bar{s}$ can be combined to form a new hedge $\pi[\bar{t}, \bar{s}]$ which is obtained from π by replacing each positive port with the hedge $\bar{t}$ and each negative port with the hedge $\bar{s}$.



We define two hedges $\bar{t}^k_{p,u,o}$ and $\bar{s}^k_{p,u,o}$ by induction on k. These hedges will be constructed from two templates $\pi = \pi_{p,u,o}$ and $\tau = \tau_{p,u,o}$ which we define first. Both π and τ are hedges of infinite width. Each tree in π or τ consists of a root with a single child that is either a positive or a negative port. For a word q, we denote by q^{rev} the reverse of q. The sequence of roots of both π and τ spell out the infinite word $(p^\omega)^{\mathsf{rev}} \cdot \mathsf{b} \cdot p^\omega$ where

$$p := (\bar{\mathsf{c}}\mathsf{a})^{pu+o^2+2}\bar{\mathsf{c}}\mathsf{b}$$

and where $\bar{\mathsf{c}}$ abbreviates $\mathsf{c}^{\max(po,o^2)}$. Now fix in both π and τ some node labelled b. We will refer to this node as the *center* of π and τ respectively. In both templates all ports but one are negative. What distinguishes π from τ is the location of the positive port with respect to the center: In π the sequence of roots from the parent of the positive port to the center is labelled $\mathsf{a}(\bar{\mathsf{c}}\mathsf{a})^{pu+1}\bar{\mathsf{c}}\mathsf{b}$; in τ this sequence is labelled $\mathsf{a}(\bar{\mathsf{c}}\mathsf{a})^{pu}\bar{\mathsf{c}}\mathsf{b}$. Figure 2 shows π and τ for $p = 1, u = 0$ and $o = 0$.

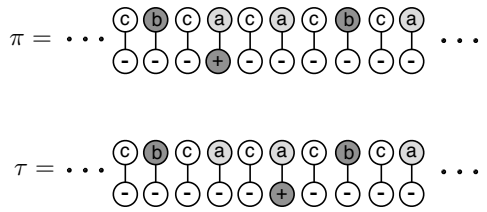


Figure 2: The templates π and τ for $p = 1, u = 0$ and $o = 0$. The right b node is the center of each template.

For fixed $p, u, o \in \mathbb{N}$ we define two sequences $\bar{t}^k_{p,u,o}$ and $\bar{s}^k_{p,u,o}$ of hedges by induction on k: $\bar{t}^0_{p,u,o}$ is the single node labelled d and $\bar{s}^0_{p,u,o}$ is the single node labelled c. For $k > 0$

$$\bar{t}^k_{p,u,o} := \pi_{p,u,o}\left[\bar{t}^{k-1}_{p,u,o}, \bar{s}^{k-1}_{p,u,o}\right]$$
$$\bar{s}^k_{p,u,o} := \tau_{p,u,o}\left[\bar{t}^{k-1}_{p,u,o}, \bar{s}^{k-1}_{p,u,o}\right]$$

We say that a node α in $\bar{t}^k_{p,u,o}$ *corresponds* to a node β in $\bar{s}^k_{p,u,o}$ (or vice versa) if α and β have the same distance and direction to the center of their respective sequence of siblings. A root α of $\bar{t}^k_{p,u,o}$ or $\bar{s}^k_{p,u,o}$ has *positive polarity*, if the sequence of its children forms $\bar{t}^{k-1}_{p,u,o}$ and α has *negative polarity* otherwise. Observe that α has positive polarity iff it is obtained from the parent of a positive port in the inductive construction of $\bar{t}^k_{p,u,o}$ or $\bar{s}^k_{p,u,o}$.

Finally, $t_{p,u,o}$ is the tree with positive polarity in $\bar{t}^{o+1}_{p,u,o}$ and $s_{p,u,o}$ is a tree with root labelled a and negative polarity in the same hedge. Figure 3 shows t and s for $p = 1, u = 0$ and $o = 0$.

Part (i) of Lemma 13 is obvious:

PROPOSITION 14. *For all $p, o, u \in \mathbb{N}$, $t_{p,u,o}$ satisfies Q_{pu+1} but $s_{p,u,o}$ does not.*

We now fix p, u, o and omit p, u, o as a subscript.

We proceed to show that Duplicator has a winning strategy for the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{rpd} \le p, \mathsf{rud} \le u; \mathsf{other} \le o)$-game on $\bar{t}^{o+1}$ and $\bar{s}^{o+1}$. We use the obvious extension of $\mathrm{CTL}^{*\leftrightarrow}$ to hedges: The node formula $\exists_\rightarrow \phi$ is true at the root α of a tree in a hedge $\bar{t}$ if ϕ is true on the sequence of roots of $\bar{t}$ to the right of α. In a similar way, the $\mathrm{CTL}^{*\leftrightarrow}$-game can be extended to hedges, such that the players can choose the sequence of roots right of the current selection in $\exists_\rightarrow$ moves. The symmetric definitions hold for leftward paths.

During the game $\bar{t}^{o+1}$ and $\bar{s}^{o+1}$, Duplicator can maintain an invariant on the string of siblings of the selected nodes α, β. This invariant is similar to the invariant used in the word game from Section 2.2. 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* $\mathsf{ipd}(\alpha)$ of a node α is the number of cs between α and the next node labelled a or b to the left of α. Then *Bottom-Depth* $\mathsf{bd}(\alpha)$ of α is the number as between α and the next b to the left of α. For a node α in $\bar{t}^k$ or $\bar{s}^k$ we denote by $\mathsf{pd}(\alpha)$ the Plateau-Depth of α with respect to the sequence of siblings of α or on the sequence of roots of $\bar{t}^k$ or $\bar{s}^k$, if α is the root. We use the same notation for $\mathsf{ipd}, \mathsf{td}$, and bd.

The following claim is sufficient for part (ii) of Lemma 13.

CLAIM 2. *Let $r \le p$ and $d, l \le o$. Assume that α is a root in $\bar{t}^{d+1}$ and β is a root in $\bar{s}^{d+1}$ such that α, β satisfy the conditions 1 to 4 below. Then Duplicator can win the $CTL^{*\leftrightarrow}(\mathsf{rpd} \le p, \mathsf{rud} \le u; \mathsf{other} \le o)$-game with α, β selected and (l, d, r) moves left to play.*

1. $\alpha =_{\mathsf{pd}} \beta$.
2. $|\mathsf{td}(\alpha) - \mathsf{td}(\beta)| \le 1$.
3. *If $\alpha \neq_{\mathsf{td}} \beta$ then*
   (a) $\mathsf{td}(\alpha) \ge ru$ *and* $\mathsf{td}(\beta) \ge ru$
   (b) *if $\mathsf{td}(\alpha) = ru$ or $\mathsf{td}(\beta) = ru$ then $\mathsf{pd}(\alpha) = \mathsf{pd}(\beta) \ge ro$.*

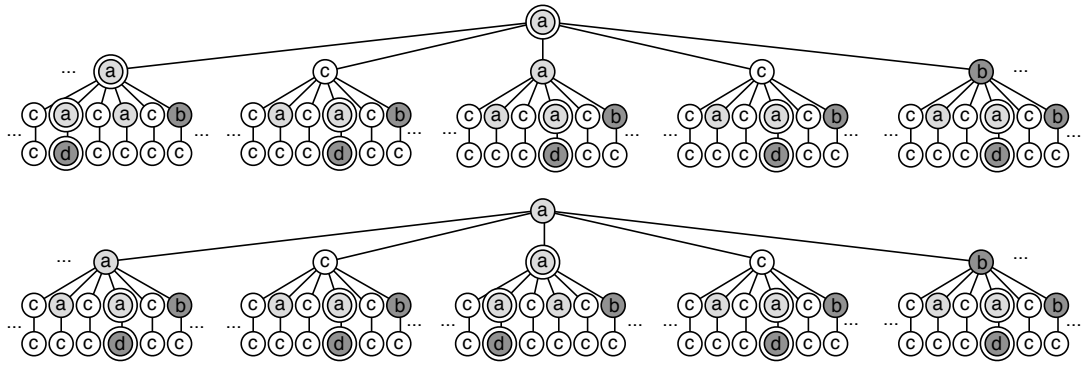


Figure 3: The trees t^2 (top) and s^2 (bottom) for $p = 1, u = 0$, and $o = 0$. Roots of positive subtrees are displayed with double circles, roots of negative subtrees with single ones.

4. *If* $\alpha \neq_{\mathsf{bd}} \beta$ *then*
   (a) $\mathsf{bd}(\alpha) \geq lo$ *and* $\mathsf{bd}(\beta) \geq lo$
   (b) *if* $\mathsf{bd}(\alpha) = lo$ *or* $\mathsf{bd}(\beta) = lo$ *then* $\mathsf{ipd}(\alpha) = \mathsf{ipd}(\beta) \geq lo$.

Proof. The lemma is proven by induction on $l + d + r$. The base case holds as it follows from the invariant that α and β have the same label.

For the induction step, we first consider downward moves. Assume that Spoiler picks a downward path p. Let α' be the child of α on p. We first determine the child β' of β on the path q that Duplicator chooses. Duplicator's goal is to pick β' such that α', β' satisfies the invariant, for $(l, d-1, r)$ moves left and such that the trees rooted at α' and β' have the same polarity. The easiest case is when the node γ that corresponds to α' has the same polarity as α': in this case Duplicator can choose β' to be γ. Otherwise there are two cases: If α' has positive polarity, then Duplicator picks the single child of β that has positive polarity. If β has negative polarity (and hence its corresponding node has positive polarity), the Duplicator picks the single child of β who's corresponding node has positive polarity. Observe in any case α' has the same polarity as β'. Thus the trees rooted at α' and β' are isomorphic, and Duplicator can choose a path starting at β' isomorphic the suffix of p starting at α'.

Chosen this way, p and q have the same labelling. Therefore Duplicator can play isomorphically on the path game. It remains to verify that all possible intermediate positions of the path game are winning positions for Duplicator. This is obvious if α, β is the final position. For the case that α', β' is an intermediate position of the path game, we verify that this position maintains the invariant for $(l, d-1, r)$. The interesting case is when α' and β' both have positive polarity. Assume that the sequence of siblings of α' is obtained from π and that the siblings of β' are obtained from τ. We now check the invariant. Conditions 1 and 2 are obvious. To check Condition 3 observe that the string from α' to the next right sibling labelled with b forms the string $\mathsf{a}(\bar{\mathsf{c}}\mathsf{a})^{pu+1}\bar{\mathsf{c}}\mathsf{b}$ and the string from β' to its next b right sibling is labelled $\mathsf{a}(\bar{\mathsf{c}}\mathsf{a})^{pu}\bar{\mathsf{c}}\mathsf{b}$ where $\bar{\mathsf{c}}$ abbreviates $\mathsf{c}^{\max(po,o^2)}$. As $r \leq p$ this shows that Condition 3 is maintained. For Condition 4, observe that both α' and β' are contained in a block of siblings forming the string

$$\mathsf{b}\bar{\mathsf{c}}(\mathsf{a}\bar{\mathsf{c}})^{pu+o^2+2}\mathsf{b}$$

Hence the string from α' to its next b left sibling is $\mathsf{a}(\bar{\mathsf{c}}\mathsf{a})^{o^2}\bar{\mathsf{c}}\mathsf{b}$ and the string from β' to its next b left sibling is $\mathsf{a}(\bar{\mathsf{c}}\mathsf{a})^{o^2+1}\bar{\mathsf{c}}\mathsf{b}$. As $l \leq o$ this shows that Condition 4 is maintained. If the final position of the path game is further down in p, q than α', β', then the selected nodes are isomorphic positions in isomorphic hedges, and therefore winning positions for any number of moves left to play.

Now assume that Spoiler picks a rightward fullpath p in either hedge. Duplicator has to choose the unique rightward fullpath q starting at the selected node in the other hedge. The players then play the LTL($\mathsf{ud} \leq u, \mathsf{ned} \leq o$) game on the selected paths. In this game, Duplicator uses the strategy described in Section 2.2. The following claim can be shown using the strategy described under Claim 1.

Claim 3. *In the setting above, assume Duplicator uses the Etessami-Wilke strategy. Let* α', β' *be the roots of the selected paths* p', q' *after* $u' \leq u$ U- *and* $o' \leq o$ F- *and* X- *moves of the path game. Then either* p' *is isomorphic to* q' *or*

1. $\alpha =_{\mathsf{pd}} \beta$.
2. $|\mathsf{td}(\alpha) - \mathsf{td}(\beta)| \leq 1$.
3. *If* $\alpha \neq_{\mathsf{td}} \beta$ *then*
   (a) $\mathsf{td}(\alpha) \geq (r-1)u + u'$ *and* $\mathsf{td}(\beta) \geq (r-1)u + u'$
   (b) *if* $\mathsf{td}(\alpha) = (r-1)u + u'$ *or* $\mathsf{td}(\beta) = (r-1)u + u'$ *then* $\mathsf{pd}(\alpha) = \mathsf{pd}(\beta) \geq (r-1)o + o'$.

It is easy to check that at any intermediate position of the path game satisfies the invariant of Claim 2.

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. □

As in the case of the word game, it is easy to see that Duplicator can still win the game if the templates are pruned to $(p^{o+1})^{\mathsf{rev}} \cdot \mathsf{b} \cdot p^o$: i.e. that Lemma 13 holds for finite trees.

Now consider the special case of Lemma 13 where $u = 0$. In this case the lemma reads that each $t_{p,0,o}$ satisfies Q_1 but no $s_{p,0,o}$ does. In addition Duplicator wins the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{rpd} \leq p, \mathsf{rud} = 0; \mathsf{other} \leq o)$-game on $t_{p,0,o}, s_{p,0,o}$. Hence Q_1 can not be expressed in $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{rpd} \leq p, \mathsf{rud} = 0; \mathsf{other} \leq o)$ for any p, o. Hence

COROLLARY 15. *$CTL^{*\leftrightarrow}(\mathsf{rud} = 0)$ is not first order complete on ordered trees of infinite width*

4. THE VERTICAL UNTIL HIERARCHY

We now show Theorem 6. We define a property P_u that can be expressed in $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u)$ but not in $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u-1)$. It simplifies our presentation to first show the theorem on infinite trees. Later, we will extend the proof to finite trees.

For all $x \in \mathbb{N}$ we let P_x be the set of ordered trees which have a fullpath p ending at d such that each suffix of p that starts with b satisfies the regular expression

$$\mathsf{b}(\mathsf{c}^*\mathsf{a})^x\Sigma^*.$$

P_u can be expressed in $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u)$ by

$$\exists_\downarrow (\mathsf{G}((\mathsf{b} \rightarrow \mu_u) \wedge (\mathsf{X}\,\mathsf{false} \rightarrow \mathsf{d})))$$

where μ_u is defined recursively by

$$\mu_0 = \mathsf{true} \qquad \mu_x = \mathsf{c}\,\mathsf{U}\,(\mathsf{a} \wedge \mu_{x-1})$$

We show that P_u can not be expressed in $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u-1)$ for every u. By Proposition 8, the following lemma is sufficient.

LEMMA 16. *For all $u \geq 1$, $o \geq 0$ there are two finite trees $t_{u,o}$ and $s_{u,o}$ such that (i) each $t_{u,o}$ satisfies P_u but no $s_{u,o}$ does and (ii) Duplicator has a winning strategy for the $CTL(\mathsf{dud} \leq u-1; other \leq o)$ game on $t_{u,o}, s_{u,o}$.*

We first describe how to construct $t_{p,o}$ and $s_{p,o}$. Figure 4 shows two templates $\pi_{u,o}$ and $\tau_{u,o}$ that will be of interest to us. Positive and negative ports are denoted with "+" and "−" respectively. The node labelled "±" represents the only distinction between $\pi_{u,o}$ and $\tau_{u,o}$: it is a positive port in $\pi_{u,o}$ and a negative port in $\tau_{u,o}$

In each of the two templates, the leftmost branch consists of the root labelled b followed by infinitely many $\mathsf{c}^o\mathsf{a}$ blocks. The final c node in any c^o sequence has two children: the left child is labelled a and the right child is a positive or negative port. In both $\pi_{u,o}$ and $\tau_{u,o}$ the topmost $u-1$ ports are negative. In $\pi_{u,o}$ the next two ports are positive, while in $\tau_{u,o}$ only the next port is positive. All other ports are negative.

Fixing $u \geq 1$, $o \geq 0$, we will define two sequences of trees $t^k_{u,o}$ and $s^k_{u,o}$ by induction on k. $t^0_{u,o}$ consists of a single node

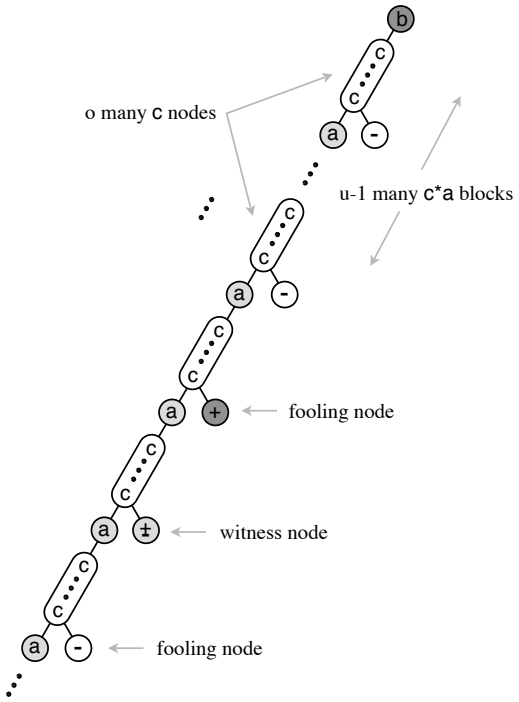


Figure 4: The templates $\pi_{u,o}$ and $\tau_{u,o}$

labelled d, while $s^0_{u,o}$ consists of a single node labelled c. For $k \geq 1$:

$$t^k_{u,o} := \pi_{u,o}\left[t^{k-1}_{u,o}, s^{k-1}_{u,o}\right]$$
$$s^k_{u,o} := \tau_{u,o}\left[t^{k-1}_{u,o}, s^{k-1}_{u,o}\right]$$

We will later define $t_{u,o}$ (and $s_{u,o}$) to be $t^k_{u,o}$ ($s^k_{u,o}$ respectively) for some large k that depends on u and o.

We call a $\mathsf{b}(\mathsf{c}^*\mathsf{a})^\omega$ labelled path within $t^k_{u,o}$ or $s^k_{u,o}$ a *stem*. Observe that a stem is isomorphic to $\pi_{u,o}$ (or $\tau_{u,o}$) without ports. A node that is connected to the root of a stem by a path labelled $\mathsf{b}(\mathsf{c}^*\mathsf{a})^x\mathsf{c}^*\mathsf{b}$ is called a *witness node* if $x = u$ and *fooling node* if $x = u-1$ or $x = u+1$. A path that always departs from the stem on the witness node is a *witness path*. Observe that both t^k and s^k contain several witness paths. The *enclosing subtree* of a node α in $t \in \{t^k_{u,o}, s^k_{u,o}\}$ is the smallest subtree of t that contains α and that is isomorphic to either $t^{k'}_{u,o}$ or $s^{k'}_{u,o}$ for some $k' \leq k$. A Subtree t of $t^k_{u,o}$ or $s^k_{u,o}$ has *positive polarity* if it is isomorphic to $t^{k'}_{u,o}$ for some $k' \leq k$ and t has *negative polarity* otherwise. The *polarity* $\mathsf{pol}(\alpha)$ of a node α is the polarity of its enclosing subtree.

We first note:

PROPOSITION 17. *Let $u \geq 1$ and $k, o \geq 0$. Then $t^k_{u,o}$ satisfies P_u but $s^k_{u,o}$ does not.*

PROOF. The proposition obviously holds for the trees $t^0_{u,o}$ and $s^0_{u,o}$. We show by induction on k that the witness path that contains the root of $t^k_{u,o}$ witnesses that $t^k_{u,o} \in P_u$: As the

witness path departs from the topmost stem on the witness node it starts with a sequence labelled $\mathsf{b}(\mathsf{c}^*\mathsf{a})^u\mathsf{c}^*\mathsf{b}$. Hence the first b is followed by sufficiently many a nodes. In addition, it departs from the topmost stem into a subtree isomorphic to $t^{k-1}_{u,o}$. Within $t^{k-1}_{u,o}$ we can conclude by induction that the witness path containing the root witnesses that $t^k_{u,o} \in P_u$.

For $s^k_{u,o}$ we know by induction that any subtree of $s^k_{u,o}$ of negative polarity does not contain a path witnessing that $s^k_{u,o} \in P_u$. Hence a path witnessing $s^k_{u,o} \in P_u$ must contain the upper fooling node of the topmost stem. But any path that contains this node starts with a sequence labelled $\mathsf{b}(\mathsf{c}^*\mathsf{a})^{u-1}\mathsf{c}^*\mathsf{b}$. □

We now turn to part (ii) in Lemma 16. Before we delve into the details of Duplicator's strategy, we describe this strategy at a higher level.

4.1 The Strategy: Challenges for Duplicator

Fixing $u \geq 1$, $o \geq 0$ we now refer to just t^k and s^k omitting the subscripts.

Our goal will be to show that there is a k such that Duplicator has a winning strategy for the CTL$^{*\leftrightarrow}(\mathsf{dud} \leq u-1, \mathsf{other} \leq o)$ game on t^k, s^k.

We start with some intuition about the strategy of Duplicator. The idea is that t^k and s^k 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 can not 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 t^k ends on a d while the witness path in s^k ends in a c. Clearly, Duplicator must have a remedy when Spoiler plays the witness path in t^k. In fact, Duplicator's strategy depends on the place where Spoilers path departs from the witness path. Assume Spoiler picks a path pp' that departs from the witness path on the root of p'. Then Duplicator's response depends on the length on p.

The case where p is short is "easy" for Duplicator: she picks a path qq' such that q is isomorphic to p. To choose q', observe that there are two possibilities for path pp' to depart from the witness path. If pp' departs above the witness node then Duplicator can choose the root of q' such that the roots p' and q' have the same polarity. As Duplicator maintained similar levels throughout the game the roots of p' and q' are on similar levels and hence winning positions for Duplicator. She can use her winning strategy to determine the rest of q'. If pp' 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 "easy" strategy to determine q'. It is easy to see 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 p, q, the Duplicator has achieved her goal to keep the trees big, and therefore she wins by induction. If the final position is in p', q' 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 p 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 guarantied that there are sufficiently may 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 $\mathsf{c}^*\mathsf{a}$ blocks between the root of a stem and the departure point is u. 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 $u-1$. 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 ([5], Theorem 9); there, the Duplicator responds to eventually moves of the Spoiler by jumping to next $\mathsf{b}(\mathsf{c}^*\mathsf{a})^*\mathsf{c}^*\mathsf{b}$ block in the word.

However, this "jumping response" of Duplicator can not 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 [16]. 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 p is not long, there is no need to perform fooling at all. On the other hand, if p is long, Duplicator can play a path that has a long regular structure at the top, which allows him to perform the exaggerated jump.

We now formalise the notion of "similar levels". The next lemma states that if k is sufficiently big, then t^k and s^k contain many levels of subtrees that can not be distinguished by CTL$^{*\leftrightarrow}(\mathsf{ud} \leq u, \mathsf{ned} \leq o+1)$ formulas.

LEMMA 18. *For all $u, o \in \mathbb{N}$ there are $\mu_{u,o}, \lambda_{u,o} \in \mathbb{N}$ such that $\lambda_{u,o} \geq 2$ and for all $n \geq \mu_{u,o}$*

$$t^n_{u,o} \equiv_{u,o+1} t^{n+\lambda_{u,o}}_{u,o} \text{ and } s^n_{u,o} \equiv_{u,o+1} s^{n+\lambda_{u,o}}_{u,o}$$

As we fixed u, o above, we abbreviate $\mu_{u,o}$ by μ and $\lambda_{u,o}$ by λ.

PROOF. Let E_k be the pair of $\equiv_{u,o+1}$-classes of $t^k_{u,o}$ and $s^k_{u,o}$. Since the number of equivalence classes is finite for each fixed u, o, there must be μ and λ such that $E_{\mu+\lambda} = E_\mu$, and we claim that this μ and λ suffice. We first note that for a fixed template q the $\equiv_{u,o+1}$-class of $q[t, s]$ for ordered trees t, s depends only on the $\equiv_{u,o+1}$-classes of t and s. This follows by induction using the usual composition technique for trees (see e.g. [15]). Hence from $t^n_{u,o} \equiv_{u,o+1} t^{n+\lambda}_{u,o}$ and $s^n_{u,o} \equiv_{u,o+1} s^{n+\lambda}_{u,o}$ we can conclude (applying $\pi_{u,o}$ to both equivalence classes) $t^{n+1}_{u,o} \equiv_{u,o+1} t^{n+1+\lambda}_{u,o}$, and $s^{n+1}_{u,o} \equiv_{u,o+1} s^{n+1+\lambda}_{u,o}$ (applying $\tau_{u,o}$ to both equivalence classes). The result now follows by induction. □

DEFINITION 19. *For all $u, o \in \mathbb{N}$ we define the relation $\doteq_{u,o} \subseteq \mathbb{N} \times \mathbb{N}$ by*

$$m \doteq_{u,o} n \text{ iff } n, m \geq \mu \text{ and } n = m + \lambda$$

Again, we write $\doteq$ for $\doteq_{o,u}$. Observe that if $n, m \geq \mu$ then $n \doteq m$ iff $n + k \doteq m + k$ for any $k \in \mathbb{N}$.

A node α is on level i if its enclosing subtree is isomorphic to either t^i or s^i. Two nodes α, β in t^k or s^k are on *similar* levels if α is on level i, β is on level j, and $i \doteq_{u,o} j$. In this case we write $\alpha \doteq \beta$.

We now define

$$t_{u,o} := t^{f(u,o)}_{u,o}$$
$$s_{u,o} := s^{f(u,o)}_{u,o}$$

where $f(u, o) = 3o^2\lambda + \mu + 1$.

We now turn to the notion of an "easy game". For many positions α, β in the game played on t^k and s^k, there is an "easy win" for Duplicator in the CTL$^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o)$ game starting at α, β; this is the case whenever α, β are in stems of similar (i.e. $\doteq$) depth, they are locally isomorphic on their stems, and both nodes are not on the witness path.

We denote by $\mathsf{pd}(\alpha)$ the Plateau-Depth (pd) of α on the stem that contains α.

LEMMA 20. *Let $k \in \mathbb{N}$. Let α be a node in t^k and let β be a node in s^k such that*

- $\alpha \doteq \beta$,
- $\alpha =_{\mathsf{pd}} \beta$, *and*
- α *and* β *are both not on witness paths.*

Then Duplicator has a winning strategy for the CTL$^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$ game on t^k, s^k with α, β selected.

PROOF. The definition of $\doteq$ implies that Duplicator has a winning strategy for the CTL$^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$ game played from the root of any two enclosing trees of the same polarity (i.e. both t^i's or both s^i's) that are λ apart. Duplicator can derive a winning strategy for the CTL$^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$ game starting at α, β from this: in response to a path move p of Spoiler, she plays a fullpath that concatenates a path isomorphic to q on the uppermost stem with a fullpath that is a given by the winning strategy above below the uppermost stem. The latter winning strategy exists because the stems below the uppermost ones must also differ in b-depth by λ, and must have the same polarity. Subsequent next, eventually, and until moves can be handled by breaking up into cases: for example, an eventually move in the uppermost stem is answered isomorphically, while an eventually move outside the uppermost stem is done according to the winning strategy. □

4.2 The Strategy in Detail

In this section we show a claim that is sufficient for part (ii) of Lemma 16. As in the game arguments given in the previous sections, we show that Duplicator can maintain an invariant throughout the game. Recall that we denote by $\mathsf{pd}(\alpha)$ the Plateau-Depth of α on the stem that contains α.

CLAIM 4. *Let $d, l, r \leq o$ and let $k > \lambda(u+o)(l+d+r) + \mu$. Assume that p, q are paths in t^k, s^k that satisfy the Conditions 1 to 4 below. Then Duplicator can win the CTL$^{*\leftrightarrow}(\mathsf{dud} \leq u-1; \mathsf{other} \leq o)$-game with (l, d, r) path moves left to play and p, q selected.*

1. $\mathsf{lab}(\mathsf{root}(p)) = \mathsf{lab}(\mathsf{root}((q))$.
2. $p \doteq q$.
3. $p =_{\mathsf{pd}} q$.
4. *If $\mathsf{pol}(\mathsf{root}(p)) \neq \mathsf{pol}(\mathsf{root}(q))$ then $\mathsf{level}(\mathsf{root}(p)) > m$ and $\mathsf{level}(\mathsf{root}(q)) > m$ where $m = (\lambda(u+o)+1)(l+d+r)+\mu$.*

PROOF. If $\mathsf{pol}(\mathsf{root}(p)) = \mathsf{pol}(\mathsf{root}(q))$ then Duplicator has a winning strategy by definition of $\doteq$. Hence we prove the claim by induction on $l+d+r$ assuming that $\mathsf{pol}(\mathsf{root}(p)) \neq \mathsf{pol}(\mathsf{root}(q))$.

The base case follows from Condition 1.

In the induction step, the cases where Duplicator chooses a horizontal path are obvious.

For downward moves we distinguish several cases.

Case 1. Spoiler chooses a downward path p in t^k. As noted above, Duplicator's strategy depends on the length of the prefix of p that is on a witness path. Therefore let $p_1, \ldots, p_n$ be a partition of p such that the concatenation $p_1 \cdot \ldots \cdot p_{n-1}$ is a maximal prefix of p that is a witness path and each p_i $(i \leq n-1)$ is contained in exactly one stem.

Below we write $r =_{\mathsf{lab}} s$ if the path r has the same labels as the path s. Observe that the labelling determines a downward path in t^k or s^k as any node has at most one child with each label.

Case 1.1. $n \leq \lambda(u+o)+1$. That is, Spoiler has made a "short move". As described in the previous subsection, Duplicator has an easy strategy. Duplicator first chooses a prefix $q_1 \cdot \ldots \cdot q_{n-1}$ of the full path $q = q_1 \cdot \ldots \cdot q_n$ that she will choose in the game. She picks this prefix such that $q_i =_{\mathsf{lab}} p_i$ for $i \leq n-1$. The q_i exists by Condition 4 and our assumption that $\mathsf{pol}(\alpha) \neq \mathsf{pol}(\beta)$. To determine q_n, recall that p departs from the witness path on the root of p_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 α_n of p_n is labelled b and in the second case α_n is labelled a. In both cases we define the root β_n of q_n to be the child of the leaf of q_{n-1} that has the same label α_n. If α_n and β_n are labelled a then both nodes are not on *any* witness path. Hence it follows from Lemma 20 that Duplicator wins the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$-game if α_n, β_n are selected. If α_n and β_n are labelled b then the subtrees rooted at α_n and β_n have the same polarity. As α_n and β_n are also on $\doteq$ levels, it follows from the definition of $\doteq$ that Duplicator can win the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$ if α_n, β_n are selected. In both cases, Duplicator can use her winning strategy for the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$-game to determine a path q_n rooted at β_n such that she has a winning strategy for the $\mathrm{LTL}(\mathsf{ud} \leq o, \mathsf{ned} \leq o+1)$ on p_n, q_n and each intermediate position of this game is a winning position for the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o)$-game.

We now describe Duplicator's strategy for the LTL game on p, q. Duplicator can the play isomorphically on $p_1 \cdot \ldots \cdot p_{n-1}$ and $q_1 \cdot \ldots \cdot q_{n-1}$ as these paths are isomorphic. If Spoiler chooses a node in p_n or q_n then Duplicator can use her winning strategy for the $\mathrm{LTL}(\mathsf{ud} \leq u-1, \mathsf{ned} \leq o+1)$ on p_n, q_n. These strategies can be composed to derive a winning strategy for p, q (as described in Lemma 20). This composed strategy has the property that if r, s is an intermediate position of the path-game, then either the root of r is in $p_1 \cdot \ldots \cdot p_{n-1}$ and the root of s is in $q_1 \cdot \ldots \cdot q_{n-1}$ or the root of r in p_n and the root of s is in q_n.

It remains to show that any intermediate position r, s of the path-game on p, q is a winning position for Duplicator in the $\mathrm{CTL}^{*\leftrightarrow}$ $(\mathsf{dud} \leq u-1; \mathsf{other} \leq o)$-game. For positions where the root of r is in $p_1 \cdot \ldots \cdot p_{n-1}$ and the root of s is in $q_1 \cdot \ldots \cdot q_{n-1}$ it is easy to see that Conditions 1 to 3 of Claim 4 are true. Condition 4 is satisfied as $p_1 \cdot \ldots \cdot p_{n-1}$ and $q_1 \cdot \ldots \cdot q_{n-1}$ are "short" $(n \leq (u+o)\lambda)$ and hence the levels of the roots of both r and s are sufficiently high up in the trees. Thus Duplicator wins on these position by induction. If the root of r is in p_n and the root of s is in q_n then γ, δ is a winning positions for Duplicator by construction.

Case 1.2. $n > \lambda(u+o)+1$. That is, Spoiler has played a "long move".

We first describe how Duplicator picks a path in response. She will first pick a prefix $q_1 \cdot \ldots \cdot q_{\lambda(u+o)+1}$ of her response $q = q_1 \cdot \ldots \cdot q_{\lambda(u+o)+2}$ in the game. The q_i with $i \leq \lambda(u+o)+1$ have the following properties:

- $q_1 =_{\mathsf{lab}} p_1$
- q_2 is labelled $\mathsf{b}(\mathsf{c}^o\mathsf{a})^{u-1}\mathsf{c}^o$. That is q_2 is a prefix of a stem that departs from its stem on the upper fooling node. This is Duplicator's "fooling" move.
- $q_i =_{\mathsf{lab}} p_i$ for $3 \leq i \leq \lambda(u+o)+1$.

The q_i $(i \leq \lambda(u+o)+1)$ exist by Condition 4.

$q_{\lambda(u+o)+2}$ can be obtained as in the short move. Let α' be the root of $p_{\lambda(u+o)+2}$ and let β' be the child of the leaf of $q_{\lambda(u+o)+1}$ that has the same label as α'. Observe that both the root of p_3 and the root of q_3 are roots of positive subtrees. Hence α' and β' have the same polarity and Duplicator wins the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o+1)$ game if α', β' are selected. This allows Duplicator to determine $q_{(u+o)\lambda+2}$ rooted at β' such that she wins the $\mathrm{LTL}(\mathsf{ud} \leq u, \mathsf{ned} \leq o)$-game on $p_{\lambda(u+o)+2} \cdot \ldots \cdot p_n$ and $q_{\lambda(u+o)+2}$, and each intermediate position of this game is a winning position for the $\mathrm{CTL}^{*\leftrightarrow}(\mathsf{dud} \leq u; \mathsf{other} \leq o)$ game. Duplicator chooses this path, which completes the construction of q.

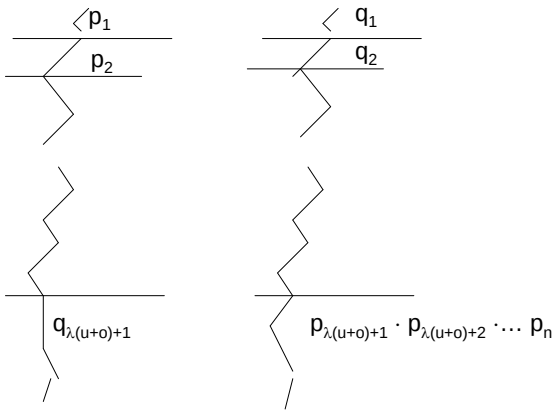


Figure 5: The paths chosen by Spoiler and Duplicator in Case 1.2

In summary, we have two paths that consist of: isomorphic segments on the first stem, similar (but undetectably different) segments on the second stem, isomorphic segments lying on the witness path for a large number of subsequent stems, and then terminating with suffixes that we know are indistinguishable in a suitable LTL game (compare with the informal description in Subsection 4.1). See Figure 4.2

We will show that Duplicator can win the LTL($\mathsf{ud} \leq u-1, \mathsf{ned} \leq o$) game on $p_2 \dots p_{\lambda(u+o)+1}$ and $q_2 \dots q_{\lambda(u+o)+1}$, and that any intermediate position of this game is a winning position for Duplicator in the CTL$^{*\leftrightarrow}$ ($\mathsf{dud} \leq u-1; \mathsf{other} \leq o$)-game. As in Case 1.1. this strategy can be combined with the obvious strategies for p_1, q_1 and $p_{\lambda(u+o)+2} \cdot \dots \cdot p_n, q_{\lambda(u+o)+2}$ to get a strategy for p, q.

Observe that the paths $p_2 \dots p_{\lambda(u+o)+1}$ and $q_2 \dots q_{\lambda(u+o)+1}$ are "long" versions of the paths used in Lemma 10. 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 $\mathsf{c}^o\mathsf{a}$ within a stem corresponds to a plateau, and hence the Plateau-Depth is the position within such a block. We show that Duplicator can maintain an invariant similar to the one in Claim 1. Let the *End-Depth* $\mathsf{ed}(r)$ of a path r is the number of b labelled nodes on r — that is, the number of stems remaining in the path.

Claim 5. *Duplicator can play the LTL($\mathsf{ud} \leq u-1, \mathsf{ned} \leq o$) game on $p_2 \cdot \dots \cdot p_{\lambda(u+o)+1}$ and $q_2 \cdot \dots \cdot q_{\lambda(u+o)+1}$ so that if there are $o' \leq o$ X- or F-moves and $u' \leq u-1$ U-moves left to play then the selected path p', q' maintain the conditions below.*

1. $p' =_{\mathsf{pd}} q'$.
2. $|\mathsf{td}(p') - \mathsf{td}(q')| \leq 1$.
3. *If* $p' \neq_{\mathsf{td}} q'$ *then*
   (a) $\mathsf{td}(p') \geq u'$ *and* $\mathsf{td}(q') \geq u'$
   (b) *if* $\mathsf{td}(p') = u'$ *or* $\mathsf{td}(q') = u'$ *then* $\mathsf{pd}(p') = \mathsf{pd}(q') \geq e'$.
4. *If* $p' \neq_{\mathsf{ed}} q'$ *or* $p' \neq_{\mathsf{td}} q'$ *then* $\mathsf{ed}(p') \geq \lambda(u'+o')$ *and* $\mathsf{ed}(q') \geq \lambda(u'+o')$
5. $p' \doteq q'$

Observe that Conditions 1 to 3 are identical with Conditions 1 to 3 of Claim 1. Duplicator's strategy is an extension of her strategy explained in the proof of Claim 1.

Fix p' and q' satisfying the invariant. If $u'+o'=0$ then it follows from the invariant that p' and q' have the same label and Duplicator wins. Now let $u'+o'>0$.

It is easy to check that X-moves can not falsify Conditions 1 to 5. Now assume that Spoiler plays an F move and he selects a suffix r^i of some $r \in \{p', q'\}$. Duplicator needs to choose a suffix s^j of the other path s.

Recall Duplicator's strategy for F-moves from Claim 1. There Duplicator selects s^j such that r^i is isomorphic to the suffix s^j that Spoiler selected. This is not possible in our current setting for two reasons. First, the paths p' and q' are finite, and hence there might not be a suffix of s that is isomorphic to r^i. Second, Duplicator must make sure that the roots of the selected paths are $\doteq$.

Therefore Duplicator distinguishes two cases:

1. If Spoiler picks a suffix r^i that is a suffix of s then Duplicator picks s^i.
2. Otherwise, s must be a suffix of r^i, since the initial paths were in a suffix relationship, and hence suffixes must be comparable. Duplicator picks the largest suffix of s such that $r^i =_{\mathsf{pd}} s^j$, $r^i =_{\mathsf{td}} s^j$, and $r^i \doteq s^j$. This is Duplicator's "exaggerated jump", moving up to λ many $\mathsf{b}(\mathsf{c}^o\mathsf{a})^u\mathsf{c}^o$ blocks (that is up to λ many stems) to pick a suffix with the desired properties.

Observe that in the first case the roots of p^i, q^j are $\doteq$ as the leafs of p^i, q^j are $\doteq$ and both paths span the same number of stems. The other conditions are easy to check. In the second case the chosen suffix of s will always be within the next λ many $\mathsf{b}(\mathsf{c}^o\mathsf{a})^u\mathsf{c}^o$ blocks. This shows that Condition 5 is maintained. The other Conditions are obviously maintained.

We now describe Duplicator's strategy for U-moves. Assume that Spoiler picks a suffix r^i of $r \in \{p', q'\}$. Duplicator will choose a suffix s^j of the other path s. We let (γ, γ') denote the subpath of r between γ and γ', and similarly for (δ, δ').

Observe that if $r =_{\mathsf{ed}} s$ and $r =_{\mathsf{td}} s$ then it follows from Condition 1 that r and s are isomorphic, and hence Duplicator wins. Therefore we can assume that $r \neq_{\mathsf{ed}} s$ or $r \neq_{\mathsf{td}} s$. Hence it follows from Condition 4 that $\mathsf{ed}(s) \geq \lambda(u'+o')$. In particular, $|s| \geq \lambda(u'+o')(u+1)(o+1)$ since each $\mathsf{b}(\mathsf{c}^o\mathsf{a})^u\mathsf{c}^o$ block contains exactly $(u+1)(o+1)$ many nodes. This will be sufficient for the existence of the suffixes s^j selected by Duplicator in the first half move.

We distinguish several subcases.

- **Subcase 1.** Spoiler selects a suffix r^i of r such that $i \leq o+2$; s^i exists because the cardinality of $|s| \geq \lambda(u'+o')(u+1)(o+1)$ Then Duplicator chooses s^i. Figure 6 shows a visualisation of this strategy.

  Condition 3 is maintained since the td of both r and s can decrease at most by one on such a move. The other conditions are obvious.

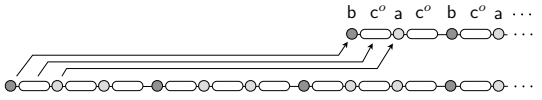


Figure 6: Duplicator's strategy on the first half move in Subcase 1. Duplicator selects s^j such that there is an arrow from the root of r^i to the root of s^j.

- **Subcase 2.** Spoiler selects a suffix r^i such that $i > o+2$ and either $i \leq \lambda(o+1)(u+1)$ or r^i is not a suffix of s.

  Duplicator selects the largest suffix s^j of s such that $s^j =_{\mathsf{pd}} r^i$, $s^j =_{\mathsf{td}} r^i$ and $s^j \doteq r^i$. The intuition is that if r^i is in the same stem as the root of r, then Duplicator picks an appropriate node within the current stem. This is possible, since the td of the roots of r and s were off by at most one, and Spoiler has moved at least $o+2$ (i.e. at least one $\mathsf{c}^o\mathsf{a}$ block). If r^i is not in the same stem as r but within the next λ stems, then Duplicator can find such a suffix by skipping the same number of stems as Spoiler, and then finding the appropriate node within that stem (see Figure 7). If $\mathsf{level}(r) - \mathsf{level}(r^i) = k$ and $k \geq \lambda$, that is r^i is more than λ stems away from r, then Duplicator skips k mod λ stems to find the appropriate suffix s^j.

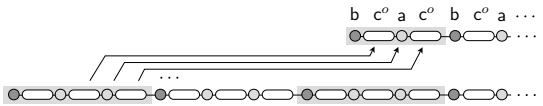


Figure 7: Duplicator's strategy for Subcase 2. when Spoiler chooses r^i such that $i \leq \lambda(o+1)(u+1)$. Duplicator selects s^j such that there is an arrow from the root of r^i to the root of s^j. All nodes in grey boxes are $\doteq$.

  Duplicator can do this because the suffix s^j can be found within the next $\lambda(o+1)(u+1)$ nodes: Observe that for any suffix of the witness path w it holds that $w =_{\mathsf{pd}} w^{o+2}$, provided that w^{o+2} exists. Hence there is a suffix s^k of s such that $s^k =_{\mathsf{pd}} r^i$ and $k \leq o+1$. A similar argument shows that a s^k with $s^k =_{\mathsf{pd}} r^i$ and $s^k =_{\mathsf{td}} r^i$ can be found within the next $(o+1)(u+1)$ nodes. If in addition $s^k \doteq r^i$ has to be satisfied then $\lambda(o+1)(u+1)$ nodes are sufficient. This shows that s^j exists, as $|s| \geq \lambda(u'+o')(u+1)(o+1)$.

  In the second half move, Spoiler selects a suffix $s^{j'}$ such that $1 < j' \leq j$. Then Duplicator selects $r^{i'}$ where

$$i' = \begin{cases} j' & \text{if } j' \leq o+2 \text{ and } \mathsf{td}(r) < \mathsf{td}(s) \\ i-(j-j') & \text{otherwise.} \end{cases}$$

  That is, if Spoiler selected a node close to the root of s then Duplicator mimics the distance from the root of s to the root of $s^{j'}$. Otherwise Duplicator mimics the distance from the end of s to the end of $s^{j'}$.

  We show that $1 < i' \leq i$. This is obvious if i' is defined by the first line of its definition. Now assume that i' is defined by the second line. Then $j' > o+2$ or $\mathsf{td}(r) \geq \mathsf{td}(s)$. First assume $\mathsf{td}(r) \geq \mathsf{td}(s)$. As r and s satisfy the conditions of Claim 5, r and s are equivalent under $=_{\mathsf{pd}}$ and $\doteq$. By Duplicator's strategy for the first half move r^i and s^j are equivalent under $=_{\mathsf{pd}}$, $=_{\mathsf{td}}$, and $\doteq$. It follows from Condition 2 that there is an $l \in \mathbb{N}$ such that either $i = j+m$ or $i = j+o+1+m$ where $m = l\lambda(o+1)(u+1)$. Substituting this into definition of i' shows that either $i' = j' + m > 0$ or $i' = j' + o + 1 + m > 1$. Now assume $j' > o+2$ and $\mathsf{td}(r) < \mathsf{td}(s)$. Then $i = j - (o+1) + m$ and thus $i' = j' - (o+1) + m$. As $j' > o+2$ it follows that $i' > 1$. It holds that $i' \leq i$ since $j' \leq j$.

  It is easy to check that the conditions are maintained if i' is defined by the first line of its definition. For the other case observe that if r^i and s^j are equivalent under $=_{\mathsf{pd}}$, $=_{\mathsf{td}}$, and $\doteq$ then r^{i-1} and s^{j-1} are equivalent under $=_{\mathsf{pd}}$, $=_{\mathsf{td}}$, and $\doteq$. Hence $r^{i'}$ and $s^{j'}$ are equivalent under $=_{\mathsf{pd}}$, $=_{\mathsf{td}}$, and $\doteq$ and thus Conditions 1, 2, 3, and 5 are true for $r^{i'}$ and $s^{j'}$.

  To check Condition 4, recall that we can assume that either $r \neq_{\mathsf{ed}} s$ or $r \neq_{\mathsf{td}} s$ (see the paragraph before the subcases). Therefore by Condition 4, $\mathsf{ed}(r) \geq \lambda(u'+o')$ and $\mathsf{ed}(s) \geq \lambda(u'+o')$. We have argued above that s was shortened by at most $\lambda(o+1)(u+1)$ nodes, thus $s^{j'}$ satisfies Condition 4. By the premise of this subcase either $i < \lambda(u+1)(o+1)$ or r^i is not a suffix of s. In both cases $\mathsf{ed}(r^i) \geq \lambda(u'+o')$ and hence $\mathsf{ed}(r^{i'}) \geq \lambda(u'+o')$.

- **Subcase 3.** Spoiler selects a suffix r^i of r such that $i > \lambda(o+1)(u+1)$ and r^i is a suffix of s. In this case Duplicator chooses s^j such that $r^i =_{\mathsf{lab}} s^j$. See Figure 8.

  In the second half move, assume that Spoiler selects a suffix $s^{j'}$ of s for some $1 < j' \leq j$. Duplicator first tries to select a suffix $r^{i'}$ of r such that $r^{i'} =_{\mathsf{lab}} s^{j'}$. If r does not have such a suffix then Duplicator selects the largest suffix of r such that $r^{i'}$ and $s^{j'}$ are equal under $=_{\mathsf{pd}}$, $=_{\mathsf{td}}$, and $\doteq$.

  We argue that there is a suffix as above exists. Note that $i > \lambda(o+1)(u+1)$ and we have argued in Subcase 2 that $\lambda(o+1)(u+1)$ nodes are sufficient to find a $r^{i'}$ with the desired properties. Hence we can find i' such $1 < i' \leq i$.

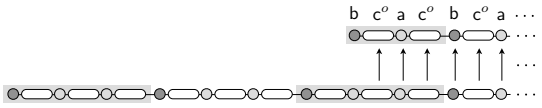


Figure 8: Duplicator's strategy for Subcase 3.

  It is obvious that the conditions are maintained, if Duplicator chooses $r^{i'}$ such that $r^{i'} =_{\mathsf{lab}} s^{j'}$. Otherwise Conditions 1, 2, 3, and 5 are obvious. We now argue that Condition 4 is maintained. First assume that $r =_{\mathsf{ed}} s$. Then there is a suffix $r^{i'}$ of r such that $r^{i'} =_{\mathsf{lab}} s^{j'}$ contradicting the hypothesis. Therefore assume $r \neq_{\mathsf{ed}} s$. Then by Condition 4 $\mathsf{ed}(r) \geq \lambda(u'+o')$ and $\mathsf{ed}(s) \geq \lambda(u'+o')$. By Duplicator's strategy in the first half move, r only got shortened by $\lambda(u+1)(o+1)$ nodes and hence $\mathsf{ed}(r^{i'}) \geq \mathsf{ed}(r) - \lambda$. In addition r must be a suffix of $s^{j'}$ and hence $\mathsf{ed}(s^{j'}) \geq \mathsf{ed}(r) \geq \lambda(u'+o')$. Thus Condition 4 is maintained.

This completes the proof of Claim 5.

Case 2. Spoiler picks a path q in s^k that is rooted at q. In this case Duplicator's strategy is largely symmetric to her strategy in Case 1. The only difference is that if Duplicator

plays a fooling move, then she chooses a path that moves off the stem on the lower fooling path.

Case 2.1. $n \leq (u+o)\lambda$. This case is exactly like Case 1.1.

Case 2.2. $n > (u+o)\lambda$. The Duplicator picks $p = p_1 \cdot \ldots \cdot p_{(u+o)\lambda}$ such that each p_i for $i \leq (u+o)\lambda - 1$ is the prefix of a stem and

- $p_1 =_{\mathsf{lab}} q_1$
- p_2 is labelled $\mathsf{b}(\mathsf{c}^o\mathsf{a})^{u+1}\mathsf{c}^o$. That is, q_2 is a prefix of a stem that departs from its stem on the *lower* fooling node.
- $p_i =_{\mathsf{lab}} q_i$ for $3 \leq i \leq \lambda(u+o)+1$.

The p_i exist by Condition 4. Duplicator picks $p_{\lambda(u+o)+2}$ as in Case 1.2.

Duplicator's strategy for the path-game is similar to Case 1.2. In fact, Duplicator can maintain the same invariant as in Case 1.2.

We fix a suffix p' of $p_2 \cdot \ldots \cdot p_{\lambda(u+o)+1}$ and a suffix q' of $q_2 \cdot \ldots \cdot q_{\lambda(u+o)+1}$ satisfying the invariant.

X-moves are as in Case 1.2.

Duplicator's strategy on F-moves is as in Case 1.2, apart from the following case. Assume that Spoiler plays an F-move and he selects p^i for some $i \leq o+2$. Observe that

$$p_2 \cdot \ldots \cdot p_{\lambda(o+u)+1} =_{\mathsf{lab}} \mathsf{b}(\mathsf{c}^o\mathsf{a})^{u+1}\mathsf{c}^o(\mathsf{b}(\mathsf{c}^o\mathsf{a})^u\mathsf{c}^o)^{\lambda(o+u)}$$
$$q_2 \cdot \ldots \cdot q_{\lambda(o+u)+1} =_{\mathsf{lab}} (\mathsf{b}(\mathsf{c}^o\mathsf{a})^u\mathsf{c}^o)^{\lambda(o+u)}$$

Thus there is no q^j such that $p^i =_{\mathsf{td}} q^j$. In this case Duplicator will select q^i. It is easy to check that this maintains the invariant.

Observe that if $i > o+2$ or Duplicator selects a suffix of q' then Duplicator can use the strategy described in Case 1.2.

On U-moves Duplicator uses the same strategy as in Case 1.2. □

4.3 The Finite Case

Part (ii) of Lemma 16 can also be shown for finite trees. The idea is to define trees t^k_{fin} and s^k_{fin} that are obtained from t^k and s^k by pruning the stems at some point. In particular, each stem in t^k_{fin} and s^k_{fin} spells out the finite word $\mathsf{b}(\mathsf{c}^*\mathsf{a})^{o(u+o)+u}$. The witness node and the fooling nodes are as in t^k and s^k.

Let the *Stem-Depth* $\mathsf{sd}(\alpha)$ of a node α be the number of a labelled nodes on a full downward path rooted at α that contains no right siblings. The Stem-Depth of a path p is the Stem-Depth of its root. One can show the following for some fixed $o \geq 0$ and $u \geq 1$.

Claim 6. *Let $d, l, r \leq o$ and let $k > \lambda(u+o)(l+d+r)+\mu$. Assume that p, q are paths in t^k_{fin}, s^k_{fin} that satisfy the Conditions 1 to 4 below. Then Duplicator can win the $CTL^{*\leftrightarrow}(\mathsf{dud} \leq u-1; \mathsf{other} \leq o)$-game with (l, d, r) path moves left to play and p, q selected.*

1. $\mathsf{lab}(\mathsf{root}(p)) = \mathsf{lab}(\mathsf{root}((q))$.
2. $p \doteq q$.
3. $p =_{pd} q$.
4. *If $\mathsf{pol}(\mathsf{root}(p)) \neq \mathsf{pol}(\mathsf{root}(q))$ then $\mathsf{level}(\mathsf{root}(p)) > m$ and $\mathsf{level}(\mathsf{root}(q)) > m$ where $m = (\lambda(u+o)+1)(l+d+r)+\mu$.*
5. $\mathsf{sd}(p) \geq d(u+o)$ *and* $\mathsf{sd}(q) \geq d(u+o)$

Duplicator can use a similar strategy as in the infinite case. But if the stems are finite, Spoiler might try to detect that the roots of the selected nodes are off by one, when Duplicator plays a "fooling" move. Therefore the finite stems must be chosen long enough for Duplicator to disguise the non-isomorphic position on the stem. We omit the details.

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 $\mathrm{CTL}^{*\leftrightarrow}$ 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 $\mathrm{CTL}^{*\leftrightarrow}_{\mathsf{ud}=k}$, in terms of fragments of first-order logic. We also need to investigate more thoroughly the relationship of the languages $\mathrm{CTL}^{*\leftrightarrow}_{\mathsf{ud}=k}$ with the queries of "bounded operator depth" mentioned by Bojańczyk [3]. For the moment we note the following distinction: [3] states that the queries of bounded operator depth cannot capture all languages of the form:

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

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

Thérien and Wilke [20] 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 $\mathrm{CTL}^{*\leftrightarrow}_{\mathsf{ud}=k}$ (or in $\mathrm{CTL}^{*}_{\mathsf{ud}=k}$).

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 5.

6. REFERENCES

[1] Pablo Barcelo and Leonid Libkin. Temporal Logics on Unranked Trees. In *LICS*, 2005.
[2] Michael Benedikt and Alan Jeffrey. Efficient and Expressive Tree Filters. In *FSTTCS*, 2007.
[3] Mikołaj Bojańczyk. Effective Characterizations of Tree Logics. In *PODS*, 2008.

[4] E. Allen Emerson. Temporal and Modal Logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, chapter 16, pages 995–1072. Elsevier Science Publishers B.V., 1990.

[5] Kousha Etessami and Thomas Wilke. An Until Hierarchy and Other Applications of an Ehrenfeucht-Fraïsse Game for Temporal Logic. *Information and Computation*, 160:88–108, 2000.

[6] Dov Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *POPL*, 1980.

[7] Thilo Hafer and Wolfgang Thomas. Computation Tree Logic CTL* and Path Quantifiers in the Monadic Theory of the Binary Tree. In *ICALP*, 1987.

[8] Hans Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.

[9] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[10] Leonid Libkin and Cristina Sirangelo. Reasoning about xml with temporal logics and automata. In *LPAR*, pages 97–112, 2008.

[11] Maarten Marx. Conditional XPath, the First Order Complete XPath Dialect. In *PODS*, 2004.

[12] Maarten Marx. Conditional XPath. *ACM Trans. Database Syst.*, 30(4):929–959, 2005.

[13] Maarten Marx. First Order Paths in Ordered Trees. In *ICDT*, 2005.

[14] Maarten Marx and Maarten de Rijke. Semantic Characterizations of XPath. In *TDM'04 Workshop on XML Databases and Information Retrieval*, 2004.

[15] Faron Moller and Alexander Rabinovich. On the Expressive Power of CTL. In *LICS*, 1999.

[16] Andreas Potthoff. First-order logic on finite trees. In *Theory and Practice of Software Development (TAPSOFT)*, 1995.

[17] Alexander Rabinovich. Expressive Power of Temporal Logics. In *Concur*, 2002.

[18] Alexander Rabinovich. Personal communciation, 2008.

[19] Alexander Rabinovich and Shahar Maoz. Why so Many Temporal Logics Climb up the Trees? In *MFCS*, 2000.

[20] Denis Thérien and Thomas Wilke. Temporal Logic and Semidirect Products: An Effective Characterization of the Until Hierarchy. *SIAM J. Comput*, 31(3):777–798, 2001.

[21] World Wide Web Consortium. XML Path Language (XPath) Recommendation. http://www.w3c.org/TR/xpath/, November 1999.