Tree Canonization and Transitive Closure [*]

Kousha Etessami
etessami@dimacs.rutgers.edu

DIMACS
P.O. Box 1179, Rutgers University
Piscataway, NJ 08855-1179

Neil Immerman
immerman@cs.umass.edu

Computer Science Department
University of Massachusetts
Amherst, MA 01003

Abstract

We prove that tree isomorphism is not expressible in the language (FO + TC + COUNT). This is surprising since in the presence of ordering the language captures NL, whereas tree isomorphism and canonization are in L ([Lin92]). Our proof uses an Ehrenfeucht-Fraïssé game for transitive closure logic with counting [Grä91, IL90].

As a corresponding upper bound, we show that tree canonization is expressible in (FO + COUNT)[$\log n$]. The best previous upper bound had been (FO + COUNT)[$n^{O(1)}$] ([DM90]). The lower bound remains true for bounded-degree trees, and we show that for bounded-degree trees counting is not needed in the upper bound. These results are the first separations of the unordered versions of the logical languages for NL, AC^1, and ThC^1.

Our results were motivated by our conjecture in [EI95] that (FO + TC + COUNT + 1LO) = NL, i.e., that a one-way local ordering sufficed to capture NL. We disprove this conjecture, but we prove that a *two-way* local ordering does suffice, i.e., (FO + TC + COUNT + 2LO) = NL.

1 Introduction

It has been known for some time that for first-order logics with ordering, a transitive closure operator (TC) gives the power of nondeterministic log-space (NL). Similarly, a deterministic restriction of transitive closure (DTC) captures deterministic log-space (L).

Fact 1 ([Imm87, Imm88])
(FO + TC+ $\leq$) = NL ; (FO + DTC+ $\leq$) = L

The ordering is necessary. Indeed without the ordering the parity of the number of vertices in a graph is not expressible in (FO + TC).

In [EI95] we introduced local orderings in graphs as an intermediate step between ordered and unordered graphs. We showed that the language (FO + DTC + 1LO) extends the Jumping Automata on Graphs (JAG) model [CR80] to a more robust complexity class that

[*]This research was supported by NSF grant CCR-9207797.

still permits interesting lower bounds on graph reachability. On the other hand, we showed that the language (FO + TC + 1LO) is strong enough to express a total ordering on the set of vertices reachable from a given vertex. This work had led us to conjecture in [EI95] that, when we add counting, all NL properties are expressible in this logic:

Conjecture 1.1 ([EI95]) (FO + TC + COUNT + 1LO) = NL

In the present paper we prove that this conjecture is false. We do so by showing

Theorem 1.2 *(Bounded Degree) Tree Isomorphism is not expressible in* (FO + TC + COUNT).

Theorem 1.2 is quite surprising because tree isomorphism is so simple, and order independent, and it seemed to require little more than counting plus a limited use of TC. Tree Isomorphism and even Tree Canonization[1] are known to be in L [Lin92]. A Corollary of Theorem 1.2 is that Conjecture 1.1 is false. This is because a 1LO on a tree gives no new information when edges are directed from the leaves to the root. On the other hand we prove:

Theorem 1.3 (FO + TC + $COUNT$ + 2LO) = NL

That is, a two-way local ordering plus the ability to count is enough for (FO + TC) to recognize any NL graph property. The proof of Theorem 1.3 involves first computing a canonical ordering on each weakly connected component of the input graph. Next, using these ordered components, we define an isomorphic graph on the (ordered) number domain.

Coming back to Theorem 1.2, we asked, how much descriptive power is needed to express tree isomorphism in the absence of ordering? We prove the following:

Theorem 1.4 *Tree Isomorphism and Tree Canonization are expressible in*

$$(\mathrm{FO} + \mathrm{COUNT})[\log n]$$

Thus, first-order formulas with counting quantifiers iterated $\log n$ times can express canonical forms for any unordered input tree. This improves the previous best upper bound of (FO + LFP + COUNT) = (FO + COUNT)$[n^{O(1)}]$ ([DM90]). The proof of Theorem 1.4 produces an inductive definition of the canonical form using counting and a 2/3 reduction argument on trees. Combining Theorems 1.2 and 1.4, we separate the languages (FO + TC + COUNT) and (FO + COUNT)$[\log n]$:

Corollary 1.5 *Tree Isomorphism* $\in$ (FO + COUNT)$[\log n]$ − (FO + TC + COUNT)

[1]Here, by "tree canonization" we mean a function f that, given a tree T, outputs an isomorphic tree T' on an ordered domain such that for all trees T'' isomorphic to T the same ordered (canonical) tree T' is the output of f.

Recall that $(\mathrm{FO} + \mathrm{COUNT}+ \leq)[\log n]$ is equal to ThC^1, the set of problems recognized by uniform sequences of polynomial-size, log-depth threshold circuits, [BIS90]. Furthermore, for the transitive closure logic, in the presence of ordering, counting quantifiers give no extra power, i.e.,

$$(\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT}+ \leq) = (\mathrm{FO} + \mathrm{TC}+ \leq) = \mathrm{NL}$$

Thus, Corollary 1.5 separates the unordered versions of the languages for NL and ThC^1. Interestingly, the lower bound of Theorem 1.2 is proved for bounded degree trees. For bounded degree trees we don't need counting:

Theorem 1.6 *Bounded Degree Tree Isomorphism and Canonization are expressible in* $(\mathrm{FO})[\log n]$.

Corollary 1.7 *Bounded Degree Tree Isomorphism* $\in (\mathrm{FO})[\log n] - (\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$.

Corollary 1.7 tells us that bounded degree tree isomorphism separates the unordered versions of the languages for NL and AC^1. An easy Ehrenfeucht-Fraïssé game argument shows that general tree isomorphism is not expressible in $(\mathrm{FO})[\log n]$, thus general tree isomorphism separates the unordered versions of the logics for AC^1 and ThC^1.

Our results thus give us much new information about the power of ordering and local ordering in logics for classes low in the NC hierarchy: L, NL, AC^1, ThC^1. To prove Theorem 1.2, we use an Ehrenfeucht-Fraïssé game for the language $(\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$ ([Grä91, IL90]). The lower bound constructs a new kind of winning strategy in which whole paths are played.

In section 2, we provide some background. In section 3, we describe the E-F Game and its correspondence to TC logic with counting. In section 4, we prove Theorem 1.2, our main lower bound. In section 5 we show that two-way locally ordered transitive closure logic corresponds precisely to NL. In section 6, we prove Theorems 1.4 and 1.6.

2 Background

In this paper our notation follows the conventions of Descriptive Complexity [Imm96]. See [Imm87] for a survey, [CFI92] for discussion of numbers, counting quantifiers and their associated Ehrenfeucht-Fraïssé games (hereafter called E-F games), and [EI95] for needed background and results concerning local orderings.

As usual, an ordered logical structure of type $\tau = \langle R_1, \ldots, R_k, c_1, \ldots, c_t\rangle$ is a tuple $\mathcal{A} = \langle\{1,\ldots,n\}, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_t^{\mathcal{A}}\rangle$. The first-order language with ordering, denoted explicitly as $(\mathrm{FO}+ \leq)$, has a numeric predicate $\leq$, denoting the usual total ordering on the universe $|\mathcal{A}| = \{1,\ldots,n\}$. When ordering is not present, we will assume in this paper that we have a second domain of numbers:

$$\mathcal{A} = \langle\{1,\ldots,n\}, \{v_1,\ldots,v_n\}, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_t^{\mathcal{A}}\rangle$$

The symbols of τ are restricted to the domain $\{v_1,\ldots,v_n\}$, and $\leq$ is defined on numbers $\{1,\ldots,n\}$. For such structures with numbers we can add counting quantifiers. Let the meaning of the formula:

$$(\exists i\, x)\varphi(x)$$

be that there exist at least i distinct points x such that $\varphi(x)$, where i here is a free variable that ranges over the number domain, and x is bound. The first-order language with counting quantifiers is denoted (FO + COUNT).

We also consider transitive closure operators TC and DTC. $(\mathrm{TC}_{x_1 \dots x_k x'_1 \dots x'_k}\, \varphi)$ denotes the reflexive, transitive closure of the binary relation $\varphi(\bar{x}, \bar{x}')$. Let (FO + TC) be the closure of first-order logic with arbitrary occurrences of TC. DTC is the deterministic transitive closure in which all multiple outgoing edges are deleted:

$$(\mathrm{DTC}_{\bar{x},\bar{y}}\varphi) \overset{def}{=} (TC_{\bar{x},\bar{y}}\varphi_d)$$
$$\varphi_d(\bar{x},\bar{y}) \overset{def}{=} \varphi(\bar{x},\bar{y}) \wedge (\forall \bar{z})(\varphi(\bar{x},\bar{z}) \rightarrow \bar{y} = \bar{z})$$

Recall that (FO + DTC+ $\leq$) and (FO + TC+ $\leq$) capture classes L and NL, respectively (Fact 1).

In our study of graph structures there are *intermediate* forms of ordering that one could augment a structure with besides a total ordering. Two of these, One-Way Local Ordering (denoted 1LO) and Two-way Local Ordering (denoted 2LO) were studied in [EI95]:

Definition 2.1 `One-Way Local Ordering`. Consider a graph

$$G = \langle \{1, 2, \dots, n\}, \{v_1, v_2, \dots, v_n\}, \leq, \mathbf{1}, \mathbf{n}, E, F, s, t \rangle$$

in which F is a ternary relation on vertices. Suppose that for each vertex, v, $F(v, \cdot, \cdot)$ is a total ordering on the vertices w for which there is an edge from v to w. Then F is called a *one-way local ordering* on (the outgoing edges of) G, and G is called a *one-way locally ordered* graph. We denote logics over graph structures augmented with one-way local ordering with the abbreviation 1LO.

A *two-way local ordering* (denoted 2LO) is just a one-way local ordering, H, on the incoming edges to each vertex, in addition to the one-way local ordering, F, on the outgoing edges. There is no assumption about consistency between F and H. ∎

In [EI95] it was shown that (FO + DTC + 1LO) is related to, but strictly more powerful than a well known structured model for space bounded computation called *JAG*s [CR80]. It was there also shown that with (FO + TC + 1LO) one can express a *total ordering* on all vertices *reachable from a particular vertex*. It was then conjectured that (FO + TC + 1LO + COUNT) = NL. It will follow from our lower bound in section 4 that this conjecture is false. However, we shall prove in section 5 that (FO + TC + 2LO + COUNT) = NL, affirming a slightly modified version of our original conjecture.

We will be considering expressibility via first-order formulas of non-constant size. Recall that FO$[t(n)]$ denotes the set of properties expressible by first-order formulas iterated $t(n)$ times [Imm89]: by a **quantifier block**, denoted $[\boldsymbol{QB}]$, we mean a sequence

$$(Q_1 z_1.M_1)(Q_2 z_2.M_2)\dots(Q_r z_r.M_r)$$

where each Q_i is a quantifier and each M_i is a quantifier free formula.

Definition 2.2 **FO$[t(n)]$** *denotes the set of properties K for which there exists a* `quantifier block` $[QB]$ *and a quantifier-free formula* M_0 *such that*

$$\mathcal{A} \in K \iff \mathcal{A} \models [QB]^{t(|\mathcal{A}|)} M_0$$

∎

3 Ehrenfeucht-Fraïssé Game for TC logic with Counting

In this section we describe an E-F game for the logic (FO + TC + COUNT) ([Grä91, IL90], see also [CM92] for a different E-F game for TC). We will use it in section 4 to prove a lower bound on tree-isomorphism. We first recall some notation for similar games such as the C_k-game from [IL90, CFI92].

For a formula $\varphi \in$ (FO + TC + COUNT), let $\boldsymbol{nd(\varphi)}$ denote the **nesting depth**, the combination of quantifier depth and TC depth for φ, defined inductively by:

$$nd(\varphi) \leftarrow \begin{cases} 0 & \text{if } \varphi \text{ is atomic} \\ nd(\psi) & \text{if } \varphi = \neg\psi \\ \max(nd(\psi), nd(\gamma)) & \text{if } \varphi = \psi \vee \gamma \\ nd(\psi) + 1 & \text{if } \varphi = (\mathrm{TC}\psi) \text{ or } \varphi = (\exists i\ x)\psi \end{cases}$$

For t a *term* (we only deal with relational vocabularies with constants, thus t is either a constant c_k or a variable x_j), let $\boldsymbol{var(t)}$ denote the set of variables that occur in t. The set of **free variables** of a formula φ, denoted $\boldsymbol{free(\varphi)}$, is defined inductively by:

$$free(\varphi) \leftarrow \begin{cases} var(t_1) \cup var(t_2) & \text{if } \varphi = (t_1 = t_2) \\ var(t_1) \cup \ldots \cup var(t_r) & \text{if } \varphi = R(t_1, \ldots, t_r) \\ free(\psi) & \text{if } \varphi = \neg\psi \\ free(\psi) \cup free(\gamma) & \text{if } \varphi = \psi \vee \gamma \\ (free(\psi) \setminus x_i) \cup j & \text{if } \varphi = (\exists j\ x_i)\psi \\ (free(\psi) \setminus \{x_1, \ldots, x_{2k}\}) \cup \bigcup_{i=1}^{2k} var(t_i) & \\ \quad \text{if } \varphi = (\mathrm{TC}_{x_1,\ldots,x_k;x_{k+1},\ldots,x_{2k}}\ \psi)(t_1, \ldots, t_{2k}) & \end{cases}$$

We use $\boldsymbol{var(\varphi)}$ to denote the set of all variables that occur in φ. We use A or $|\mathcal{A}|$ to denote the universe of a structure $\mathcal{A}$. For a structure $\mathcal{A}$, we want to define tuples of elements from A and associate them with variables in our logical language. Our logic has variables $X = \{x_1, x_2, \ldots\}$. We define an *assignment* for $\mathcal{A}$, $\tilde{a} : X \to A$ to be a partial function with domain $\boldsymbol{Dom(\tilde{a})}$ and range $\boldsymbol{Rng(\tilde{a})}$. We will say that $(\mathcal{A}, \tilde{a})$ **interprets** φ if $Dom(\tilde{a}) \supseteq free(\varphi)$. For convenience later on, we want the assignment $\tilde{a}$ to also evaluate the constants of $\mathcal{A}$, so we extend our definition to $\boldsymbol{\tilde{a} : (X \cup \{c_1, \ldots, c_t\}) \to A}$, with $\tilde{a}(c_i) = c_i^{\mathcal{A}}$. Thus, we alway have $Dom(\tilde{a}) \supseteq \{c_1, \ldots, c_t\}$. We will say that $(\tilde{a}, \tilde{b})$ is a $\boldsymbol{p}$**-configuration** if $Dom(\tilde{a}) = Dom(\tilde{b})$ and $|Dom(\tilde{a})| \leq p + t$.[2]

Definition 3.1 *Define* $\boldsymbol{(\mathcal{A}, \tilde{a}) \equiv_{m,p} (\mathcal{B}, \tilde{b})}$ *to mean: for every formula* φ *interpreted by* $\tilde{a}$ *and* $\tilde{b}$*, with* $nd(\varphi) \leq m$ *and* $|var(\varphi)| \leq p$*,* $(\mathcal{A}, \tilde{a})$ *and* $(\mathcal{B}, \tilde{b})$ agree *on* φ*, i.e.:*

$$(\mathcal{A}, \tilde{a}) \models \varphi \iff (\mathcal{B}, \tilde{b}) \models \varphi$$

∎

When $\tilde{a}$ is **empty**, i.e., it evaluates nothing but the constants, we abbreviate $(\mathcal{A}, \tilde{a})$ by $\mathcal{A}$. Thus $\mathcal{A} \equiv_{m,p} \mathcal{B}$, means $\mathcal{A}$ and $\mathcal{B}$ agree on all *sentences* φ with $nd(\varphi) \leq m$ and $|var(\varphi)| \leq p$.

[2]The additive t is to account for the constants $\{c_1, \ldots, c_t\}$ in the domain.

Definition 3.2 `Partial Isomorphism.` *Given structures $\mathcal{A} = \langle A, R_1^{\mathcal{A}}, \ldots, R_r^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots c_t^{\mathcal{A}}\rangle$ and $\mathcal{B} = \langle B, R_1^{\mathcal{B}}, \ldots, R_r^{\mathcal{B}}, c_1^{\mathcal{B}}, \ldots c_t^{\mathcal{B}}\rangle$, over the same vocabulary τ, and given assignments $\tilde{a}$ and $\tilde{b}$, define: $\tilde{\boldsymbol{a}} \cong_{\mathcal{P}_{\mathcal{A},\mathcal{B}}} \tilde{\boldsymbol{b}}$ to mean that the induced substructures of $\mathcal{A}$ and $\mathcal{B}$ generated by $Rng(\tilde{a})$ and $Rng(\tilde{b})$, respectively, are isomorphic under the mapping induced by $\tilde{a}(x_i) \mapsto \tilde{b}(x_i)$, for $x_i \in Dom(\tilde{a})$. In particular, it is necessary that $\tilde{a}(x_i) = \tilde{a}(x_j)$ iff $\tilde{b}(x_i) = \tilde{b}(x_j)$, for $x_i, x_j \in Dom(\tilde{a})$.* ∎

For $\tilde{a}$, an assignment for a structure $\mathcal{A}$, and for $a' \in A$, let $\tilde{\boldsymbol{a}}\frac{a'}{x_i}$ be the assignment for $\mathcal{A}$ that agrees with $\tilde{a}$ everywhere except it maps x_i to a'.

Definition 3.3 E-F Game for TC+COUNT. *The m round, p pebble E-F game for* TC *logic with Counting, denoted: $\boldsymbol{G_{m,p}((\mathcal{A},\tilde{a}),(\mathcal{B},\tilde{b}))}$, is played between two players called I and II, and consists of m consecutive rounds, such that at the start and after each round $(\tilde{a},\tilde{b})$ is a p-configuration. At the start and after each round PLAYER I WINS if $\tilde{a} \not\cong_{\mathcal{P}_{\mathcal{A},\mathcal{B}}} \tilde{b}$. Each round consists of the following:*

PLAYER I chooses either:
THE COUNTING MOVE:
Player I chooses a variable x_i and
a subset A' of A (or B' of B)
Player II chooses a subset B' of B (A' of A)
such that $|A'| = |B'|$
Player I picks $b' \in B'$ ($a' \in A'$)
and sets $\tilde{b} \leftarrow \tilde{b}\frac{b'}{x_i}$ ($\tilde{a} \leftarrow \tilde{a}\frac{a'}{x_i}$)
Player II responds with $a' \in A'$ ($b' \in B'$)
and sets $\tilde{a} \leftarrow \tilde{a}\frac{a'}{x_i}$ ($\tilde{b} \leftarrow \tilde{b}\frac{b'}{x_i}$)
OR
THE TRANSITIVE CLOSURE MOVE:
Player I chooses a constant $c \leq p/2$ and
$2c$ variables $x_{l_1}, \ldots, x_{l_{2c}}$ and then chooses
c-tuples $\overline{a_1}, \ldots, \overline{a_d}$ in A
(or $\overline{b_1}, \ldots, \overline{b_{d'}}$ in B), such that
$\overline{a_1}$, and $\overline{a_d}$ ($\overline{b_1}$ and $\overline{b_d}$) are already in
$Rng(\tilde{a})$ ($Rng(\tilde{b})$).
d (d') may be arbitrarily large.
Player II chooses c-tuples $\overline{b_1}, \ldots, \overline{b_{d'}}$ in B
($\overline{a_1}, \ldots, \overline{a_d}$ in A) so that $\tilde{b}^{-1}(b_{1_j}) = \tilde{a}^{-1}(a_{1_j})$
and $\tilde{b}^{-1}(b_{d'_j}) = \tilde{a}^{-1}(a_{d_j})$ for $j \in \{1, \ldots, c\}$
Player I chooses tuples $\overline{b_j}$, $\overline{b_{j+1}}$ ($\overline{a_i}$, $\overline{a_{i+1}}$)
$j \in \{1, \ldots, d'-1\}$ ($i \in \{1, \ldots d-1\}$)
and sets $\tilde{b} \leftarrow \tilde{b}\frac{\overline{b_j}\,,\,\overline{b_{j+1}}}{x_{l_1},\ldots,x_{l_{2c}}}$
(or similarly for $\overline{a_i}$ and $\overline{a_{i+1}}$)
Player II responds by choosing $\overline{a_i}$ and $\overline{a_{i+1}}$
($\overline{b_j}$ and $\overline{b_{j+1}}$), for some i (j)
and setting $\tilde{a} \leftarrow \tilde{a}\frac{\overline{a_i}\,,\,\overline{a_{i+1}}}{x_{l_1},\ldots,x_{l_{2c}}}$
(similarly for $\overline{b_j}, \overline{b_{j+1}}$)

PLAYER II WINS if PLAYER I does not win on any of the m rounds.

We write $(\boldsymbol{\mathcal{A}}, \tilde{\boldsymbol{a}}) \sim_{m,p} (\boldsymbol{\mathcal{B}}, \tilde{\boldsymbol{b}})$*, to denote the fact that PLAYER II has a* **winning strategy** *in* $G_{m,p}((\mathcal{A}, \tilde{a}),(\mathcal{B}, \tilde{b}))$*, meaning it can win regardless of the moves made by PLAYER I.* ∎

The Counting Move in the above game comes straight from [IL90, CFI92], and essentially the same TC Move was presented in [Grä91]. The idea behind the TC move is as follows: Player I, in order to reveal that the two structures disagree on the transitive closure of some $2k$-ary formula φ, will choose a φ-path of k-tuples in one structure. In response, Player II will answer with a φ-path of k-tuples in the other structure. Player I then challenges Player II by choosing a pair of consecutive tuples on the path chosen by Player II. Player II then responds to the challenge by choosing a consecutive pair of tuples in the path chosen by Player I, claiming in essence that any property that Player I could have in mind for the tuple pair it chose is also satisfied by the pair chosen by Player II. The game then proceeds with one less round left to play. The key fact about $G_{m,p}$ is:

Lemma 3.4 *For any pair of structures $\mathcal{A}$, $\mathcal{B}$ of the same finite relational vocabulary τ, and for any p-configuration $(\tilde{a}, \tilde{b})$:*

$$(\mathcal{A}, \tilde{a}) \sim_{m,p} (\mathcal{B}, \tilde{b}) \quad \Longrightarrow \quad (\mathcal{A}, \tilde{a}) \equiv_{m,p} (\mathcal{B}, \tilde{b})$$

Proof [3] We proceed by induction on m:
For the base case, when $m = 0$, clearly, if the substructures induced by $\tilde{a}$ and $\tilde{b}$ are isomorphic, then no quantifier-free formula will distinguish them.

For the inductive case, suppose the theorem is true for m, i.e., that $(\mathcal{A}, \tilde{a}) \sim_{m,p} (\mathcal{B}, \tilde{b}) \Longrightarrow (\mathcal{A}, \tilde{a}) \equiv_{m,p} (\mathcal{B}, \tilde{b})$. We want to prove it is true for $m + 1$.

Suppose $(\mathcal{A}, \tilde{a}) \sim_{m+1,p} (\mathcal{B}, \tilde{b})$, and let φ be a depth $m + 1$ formula:

Case (i): $\varphi = \exists i x_j \psi$:

Suppose, without loss of generality, that $(\mathcal{A}, \tilde{a}) \models \varphi$. Then let PLAYER I pick x_j and a subset A' of A of size i such that, for each $a' \in A'$, $(\mathcal{A}, \tilde{a}\frac{a'}{x_j}) \models \psi$. PLAYER II answers according to its winning strategy with a subset B' of B such that $i = |A'| = |B'|$. Now, for any arbitrary $b' \in B'$ that PLAYER I chooses, there is an $a' \in A'$ such that $(\mathcal{A}, \tilde{a}\frac{a'}{x_j}) \sim_{m,p} (\mathcal{B}, \tilde{b}\frac{b'}{x_j})$. Thus, by induction, $(\mathcal{A}, \tilde{a}\frac{a'}{x_j}) \models \psi \Leftrightarrow (\mathcal{B}, \tilde{b}\frac{b'}{x_j}) \models \psi$. Thus, since for each $a' \in A'$, $(\mathcal{A}, \tilde{a}\frac{a'}{x_j}) \models \psi$, we have, for each $b' \in B'$, $(\mathcal{B}, \tilde{b}\frac{b'}{x_j}) \models \psi$. Hence $(\mathcal{B}, \tilde{b}) \models \varphi$. Thus $(\mathcal{A}, \tilde{a}) \models \varphi \iff (\mathcal{B}, \tilde{b}) \models \varphi$.

Case (ii): $\varphi = (\mathrm{TC}_{\bar{x},\bar{x}'}\ \psi)(\bar{1}, \bar{n})$:

Suppose, w.l.o.g., that $(\mathcal{A}, \tilde{a}) \models \varphi$. Then let PLAYER I choose $\bar{x}, \bar{x}'$ and a set of tuples $\bar{1}, \overline{a_2}, \ldots, \overline{a_{d-1}}, \bar{n}$, such that $(\mathcal{A}, \tilde{a}\frac{\overline{a_i}, \overline{a_{i+1}}}{\bar{x}, \bar{x}'}) \models \psi$, for $i \in \{1, \ldots, d-1\}$. PLAYER II answers

[3] The reason the implication only goes one way in Lemma 3.4 is that we made the counting move slightly stronger than the counting quantifiers. Note that this just makes our lower bound slightly stronger. An equivalence can be proved if we either (1) Increase the power of the language by adding constants for all the numbers $1, \ldots, n$; or (2) Decrease the power of the counting move as follows: Let Player I choose a set A' of cardinality $\tilde{a}(x_i)$, a previously chosen number, and force Player II to reply with a set B' of cardinality $\tilde{b}(x_i)$.

according to its winning strategy with tuples $\bar{1}, \overline{b_2}, \ldots, \bar{n}$. Now for any arbitrary $\overline{b_j}, \overline{b_{j+1}}$ that PLAYER I chooses there is a pair $\overline{a_i}, \overline{a_{i+1}}$, such that $(\mathcal{A}, \tilde{a}\frac{\overline{a_i}, \overline{a_{i+1}}}{\bar{x}, \bar{x}'}) \sim_{m,p} (\mathcal{B}, \tilde{b}\frac{\overline{b_j}, \overline{b_{j+1}}}{\bar{x}, \bar{x}'})$. Thus, by the induction hypothesis, $(\mathcal{A}, \tilde{a}\frac{\overline{a_i}, \overline{a_{i+1}}}{\bar{x}, \bar{x}'}) \equiv_{m,p} (\mathcal{B}, \tilde{b}\frac{\overline{b_j}, \overline{b_{j+1}}}{\bar{x}, \bar{x}'})$. Hence, for all j, there is an i such that $(\mathcal{A}, \tilde{a}\frac{\overline{a_i}, \overline{a_{i+1}}}{\bar{x}, \bar{x}'}) \models \psi \Leftrightarrow (\mathcal{B}, \tilde{b}\frac{\overline{b_j}, \overline{b_{j+1}}}{\bar{x}, \bar{x}'}) \models \psi$. Hence, since for all i, $(\mathcal{A}, \tilde{a}\frac{\overline{a_i}, \overline{a_{i+1}}}{\bar{x}, \bar{x}'}) \models \psi$, for all j, $(\mathcal{B}, \tilde{b}\frac{\overline{b_j}, \overline{b_{j+1}}}{\bar{x}, \bar{x}'}) \models \psi$. Thus, $(\mathcal{A}, \tilde{a}) \models \varphi \iff (\mathcal{B}, \tilde{b}) \models \varphi$.

Case (iii): φ is a boolean combination of forms (i) and (ii): we only need note that the " $\iff$ " at the end of the proofs for (i) and (ii) is preserved under boolean combination.

Thus $(\mathcal{A}, \tilde{a}) \equiv_{m+1,p} (\mathcal{B}, \tilde{b})$. ∎

Observe that for $\varphi \in (\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$, $nd(\varphi)$ and $|var(\varphi)|$ are fixed with respect to n. Hence for every such formula φ, there is an equivalent formula $\varphi' \in (\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$ such that no variable is ever re-quantified and, moreover, variables are quantified in successive order, i.e., the first variable quantified in the scope of $x_1, \ldots, x_i$ is x_{i+1}. Thus, in our E-F game $G_{m,p}$ for fixed m and p, we may restrict PLAYER I's strategy so that it always chooses to map, in sequence, the lowest variables that are not already in $Dom(\tilde{a})$. We can now alternatively think of $\tilde{a} : (X \cup \{c_1, \ldots c_t\}) \rightarrow A$ as simply a k-tuple $\bar{a}$, where $a_1 = c_1^{\mathcal{A}}, \ldots, a_t = c_t^{\mathcal{A}}$ and $a_{t+1} = \tilde{a}(x_1), a_{t+2} = \tilde{a}(x_2), \ldots$. These observations allow us to simplify the presentation of the lower bound: since we are only concerned with $\varphi \in (\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$, **from now on, we use tuples $\bar{a}$, with the format described above, to denote $\tilde{a}$, and we consider only the formulas and game strategies that are restricted accordingly.**

4 Lower Bound for $(\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$

Let Tree-isomorphism be the set of structures $\langle V, V', E, E' \rangle$ such that the relations E and E' describe isomorphic directed trees on domains V and V', respectively. Let 8-bounded-Tree-isomorphism be the subset of Tree-isomorphism, where the out-degree on each vertex is bounded by 8. In this section we prove Tree-isomorphism is not expressible in $(\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$. More precisely, we prove the stronger result that 8-bounded-Tree-isomorphism is not expressible in $(\mathrm{FO} + \mathrm{TC} + \mathrm{COUNT})$.

For each i and p, we shall construct trees A_l and B_l with the following property: A_l and B_l are not isomorphic but $A_l \sim_{i,p} B_l$. It will follow from our proof of Theorem 4.2 that PLAYER II wins the game between the tree pair $\langle A_l, B_l \rangle$ and the tree pair $\langle A_l, A_l \rangle$ (where $\langle A, B \rangle$ denotes a structure with A and B defined over disjoint domains). Her strategy will be to answer each move involving the left component of either pair with the identical element of the other left component. Moves in the right component are answered according to the winning strategy for the game on A_l and B_l. As we will see, in her winning strategy for A_l and B_l, Player II always matches sets by sets of exactly the same cardinality, and paths by paths of the same length. Furthermore, in the second half of the transitive closure move, when Player I chooses tuples $\overline{b_j}, \overline{b_{j+1}}$, Player II always answers with the same numbered tuples: $\overline{a_j}, \overline{a_{j+1}}$. It thus follows that Player II's combined strategy wins the game on $\langle A_l, B_l \rangle$ and $\langle A_l, A_l \rangle$. Since the first pair are not isomorphic and the second pair are isomorphic, the result follows by Lemma 3.4.

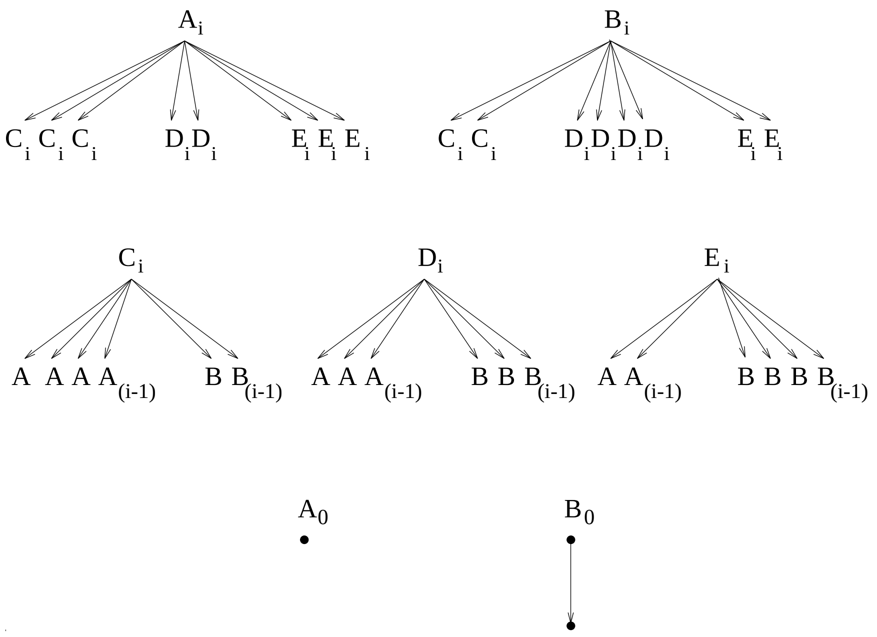

Figure 1: The Trees A_i and B_i.

We recursively define the directed trees A_i and B_i (Please see Figure 1). A_0 and B_0 are a vertex and a pair of vertices connected by an edge, respectively. For $i > 0$, A_i is a tree with a root vertex having as children 3 C_i, 2 D_i trees, and 3 E_i trees. B_i is a tree with a root vertex having as children 2 C_i, 4 D_i, and 2 E_i trees. A C_i tree has a root with children: 4 A_{i-1} and 2 B_{i-1} trees. A D_i tree has a root with: 3 A_{i-1} and 3 B_{i-1} children. An E_i tree has a root with 2 A_{i-1} and 4 B_{i-1} children. It is clear, by induction, that for all i, A_i and B_i are non-isomorphic.

Note 4.1 *There are exactly the same number of A_{i-1}'s that are descendants, two levels below, of A_i as there are descending from B_i. Likewise for the number of B_{i-1}'s.*

Theorem 4.2 $A_{ip+1} \sim_{i,p} B_{ip+1}$.

Proof The idea of the game is that at the bottom, Player II must answer A_0 with A_0 but at the top Player II must answer the root A_{ip+1} with B_{ip+1}. Player I will try to push the distinction down the tree toward the leaves, but we will show that Player II can keep distinctions from moving down more than p *levels*[4] per round.

Let $\boldsymbol{dist(v,w)}$ denote the distance between a pair of vertices. Let $\boldsymbol{depth(v)}$ denote the depth of a vertex v in a tree, the root being at depth 0. Let $\boldsymbol{st(v)}$ denote the subtree rooted at a vertex v. For a given vertex v in A_{ip+1} (or in B_{ip+1}), let the **path name** of v, denoted $\boldsymbol{pn(v)}$, be the string of symbols that denotes the subtrees leading up from v to the root of the tree. For example, the path name for a vertex at depth 4 that is at an A subtree in the tree B_{ip+1} might be $ACAEB$. For a given path name, pn, let $\boldsymbol{pn_i}$ denote the path name with the $2(i+1)$ *right most* symbols truncated. Thus, for example, for a path name

[4]By *level* we mean those depths in the trees at which A and B subtrees occur.

$pn = ACBDBCAEB$, $pn_2 = ACB$. For vertices v and w in a tree, let $\boldsymbol{lca(v,w)}$ denote the lowest common ancestor of v and w.

Definition 4.3 (e.t.s.j.) *Given a pair of vertices v_1, v_2 in A_{ip+1} and w_1, w_2 in B_{ip+1}, we will say that v_1 and v_2 are in* **exactly the same juxtaposition** *(e.t.s.j) to each other as w_1 and w_2, and we denote this by* $\boldsymbol{e.t.s.j((v_1,v_2),(w_1,w_2))}$, *if*

$$dist(lca(v_1,v_2),v_1) = dist(lca(w_1,w_2),w_1)$$

$$\&$$

$$dist(lca(v_1,v_2),v_2) = dist(lca(w_1,w_2),w_2)$$

∎

Definition 4.4 (j-Similar) *Given a k-tuple of vertices and numbers $\bar{a}$ in A_{ip+1} and a k-tuple $\bar{b}$ in B_{ip+1}, we say $\bar{a}$ and $\bar{b}$ are* **j-similar** *if they satisfy the following equivalence relation:*

$\boldsymbol{\bar{a} \approx_j \bar{b}} \Leftrightarrow$ *For each $l \in \{1,\ldots,k\}$:*

```
        If $a_l$ is a number then $b_l$ is the same number.
        Else (i.e., for vertices):
          $depth(a_l) = depth(b_l)$
          $pn(a_l)_j = pn(b_l)_j$
          For each $r \in \{1,\ldots,k\}$, such that $a_r$ and $b_r$ are vertices:
            $e.t.s.j.((a_l,a_r),(b_l,b_r))$
```

∎

Note that the equivalence relation is more restrictive for small j, i.e.:

$$\tilde{a} \approx_j \tilde{b} \;\Rightarrow\; \tilde{a} \approx_{j+1} \tilde{b}$$

Lemma 4.5 *In an i round game, given that $\bar{a}$ and $\bar{b}$ are k-tuples in A_{ip+1} and B_{ip+1}, respectively, with $i \geq j$, and $p \geq k$:*

$$\bar{a} \approx_{ip-jp} \bar{b} \implies (A_{ip+1},\bar{a}) \sim_{j,p} (B_{ip+1},\bar{b})$$

Proof The proof proceeds by induction on j. Player II's winning strategy will be such that, for both the Counting Move and the TC move, if $\bar{a} \approx_{ip-jp-p} \bar{b}$ before a round, then $\bar{a} \approx_{ip-jp} \bar{b}$ after the round, where $\bar{a}$ and $\bar{b}$ have been modified according to the round.

For $j = 0$, we have that $\bar{a} \approx_{ip} \bar{b}$. Thus, since the vertices in $\bar{a}$ have e.t.s.j. to each other as the vertices in $\bar{b}$, $(A_{ip+1},\bar{a}) \sim_{0,p} (B_{ip+1},\bar{b})$. Note that this holds regardless of what k is.

Suppose true for j, we will prove it is true for $j+1$. There are two cases to consider: either **1)** PLAYER I's first move is a counting move, or **2)** PLAYER I's first move is a transitive closure move.

1) Counting Move: In the counting move case, we only need the weaker hypothesis $\bar{a} \approx_{pi-pj-1} \bar{b}$. A one-to-one correspondence, f, is constructed between the vertices of A_{ip+1} and B_{ip+1}, such that $\bar{a}$ gets mapped to $\bar{b}$, and such that vertices with a given juxtaposition to $\bar{a}$ get mapped to vertices with the same juxtaposition to $\bar{b}$, and moreover, such that the mapping preserves path names up to level $ip - jp + 1$. The key feature of A_{ip+1} and B_{ip+1} used to achieve this is mentioned in Note 4.1.

The mapping f then determines Player II's response in the counting move. Player II responds to a chosen set A' by choosing $f(A')$, and thereafter when Player I chooses $b \in f(A')$, Player II responds with $f^{-1}(b)$. The following claim establishes f:

Claim 4.6 *Given k-tuples $\bar{a}$ and $\bar{b}$, such that $\bar{a} \sim_{ip-jp-1} \bar{b}$, there exists a one-to-one correspondence $f : A_{ip+1} \mapsto B_{ip+1}$, with the following properties:*

1. $\forall l \in \{1, \ldots, k\}\ f(a_l) = b_l$
2. $\forall x \in A_{ip+1} \forall l \in \{1, \ldots, k\}$
   $e.t.s.j.((x, a_l), (f(x), b_l))$
3. $\forall x \in A_{ip+1}\ pn(x)_{ip-jp} = pn(f(x))_{ip-jp}$

Proof Here is how f is defined. For each vertex x in A_{ip+1}, look at the path from x to the root of the tree, and find the first vertex, x', going up from x to the root, which has a descendant chosen already, i.e., one of its descendants is in $\bar{a}$, say a_l. Thus $x' = lca(x, a_l)$ (if $k = 0$, i.e., there are no chosen points yet, let x' be the root).

If this vertex x' is at or below level $ip - jp$, then we know that the path name of a_l and b_l agree up to x' (we'll say, y' for b_l), thus x' is the root of a subtree isomorphic, even in labels, to the subtree rooted at y' in B_{ip+1}, and we will construct the mapping, f, so that it maps the subtree at x' to that at y' using the isomorphism (including the labels).

If, the vertex x' is above level $ip - jp$, then we can no longer guarantee that the labels at x' and y' are identical, however, we know that x' and y' have exactly the same number of children whose subtree contains no chosen point (i.e., points in $\bar{a}$ and $\bar{b}$, respectively), and among one of these subtrees of children of x' is x. The mapping f, will map the empty subtree that x is in to one of the empty subtrees rooted at a child of y', such that every vertex at level $ip - jp + 1$ or below is mapped to a vertex with the same path name up to level $ip - jp + 1$. We know this can be achieved since x' and y' are above level $ip - jp$, and thus their children each have the same number of A descendants and B descendants at level $ip - jp + 1$.

We have thus defined a one-to-one correspondence between A_{ip+1} and B_{ip+1}, such that path names at or below level $ip - jp + 1$ are preserved, i.e., $pn(x)_{ip-jp} = pn(f(x))_{ip-jp}$. ∎

Thus, w.l.o.g., regardless of the set A' that Player I chooses, and of the point $b' \in f(A')$ that it chooses, $f^{-1}(b')$ and b' have identical juxtapositions to $\bar{a}$ and $\bar{b}$ respectively, and $f^{-1}(b')$ and b' have path names that agree up to level $ip - jp + 1$, i.e., $(A_{ip+1}, \bar{a}, f^{-1}(b')) \approx_{ip-jp} (B_{ip+1}, \bar{b}, b')$ and hence $(A_{ip+1}, \bar{a}, f^{-1}(b')) \sim_{j,p} (B_{ip+1}, \bar{b}, b')$. Thus, $(A_{ip+1}, \bar{a}) \sim_{j+1,p} (B_{ip+1}, \bar{b})$.

2) TC move: For the TC move, we need the full hypothesis $\bar{a} \approx_{ip-jp-p} \bar{b}$.

Recall the situation: Player I plays c-tuples $\overline{a_1}, \ldots, \overline{a_d}$, and we must reply with $\overline{b_1}, \ldots, \overline{b_{d'}}$. We will think of the moves proceeding in time, so Player I plays $\overline{a_1}$ and Player II answers with $\overline{b_1}$, Player I plays $\overline{a_2}$ and Player II answers with $\overline{b_2}$, etc, except that Player II can look ahead at Player I's future moves. Furthermore, we only have to remember points in the last c-tuple and the current c-tuple.

Just when Player I is pebbling the vertex $a_{l,m}$, we will call **currently pebbled** those vertices pebbled prior to this TC move, call them $\bar{a}$, plus $a_{l-1,1}, \ldots, a_{l-1,c}$ and $a_{l,1}, \ldots, a_{l,m-1}$.

Definition 4.7 Sound Response. *When PLAYER I pebbles a point $a_{l,m}$, we will say that a response pebbling of a vertex $b_{l,m}$ by PLAYER II is* **sound** *if for all currently pebbled pairs a' and b':*

$$e.t.s.j.((a', a_{l,m}), (b', b_{l,m}))$$

∎

Thus, with a sound response the juxtapositions to currently pebbled points are the same. PLAYER II will always make sound responses.

Player II's Strategy: We are going to describe Player II's response when Player I pebbles $a_{l,m}$ for some l and m. If $a_{l,m}$ is a number, then Player II's response $b_{l,m}$ will be the same number. Otherwise, a vertex $a_{l,m}$ at some depth r in A_{ip+1} defines a path $p_0, p_1, \ldots, p_r = a_{l,m}$ from the root, p_0, to $a_{l,m}$. Player II's strategy will inductively determine a corresponding path $q_0, q_1, \ldots, q_r$, in B_{ip+1}. Initially, q_0 is the root of B_{ip+1}. q_r will determine $b_{l,m}$, the actual response to $a_{l,m}$.

Inductively, suppose that thus far $q_0, \ldots, q_j$ have been determined. There are several possible cases:

1. If some point a_s in $st(p_{j+1})$ is currently pebbled, i.e., p_{j+1} is a_s's ancestor: Player II should respond by picking q_{j+1} to be the child of q_j which is b_s's ancestor[5].

2. If $st(p_{j+1})$ is currently an empty subtree:

a If no sibling subtree $st(w)$ of $st(p_{j+1})$ is currently pebbled, then Player II responds with a child q_{j+1} of q_j such that q_{j+1} and p_{j+1} have the same label (this is always possible because there are at least two children of q_j with the same label as p_{j+1}).

b If just *one* sibling subtree $st(w_1)$ of $st(p_{j+1})$ is currently pebbled:

Look Ahead in Player I's moves for the *first time* $l' > l$ that $a_{l',1}, \ldots, a_{l',c}$, $a_{l'+1,1}, \ldots, a_{l'+1,c}$, and $\bar{a}$ pebble at most *one* of the subtrees rooted at p_j's children.

Either exactly one subtree $st(w_2)$ is pebbled, or none of the siblings are pebbled, or $l' = d + 1$, i.e., the entire rest of the c-tuple sequence has more than one subtree pebbled. In either of the latter cases we arbitrarily pick one of the pebbled subtrees $st(w_2)$ at time $l' - 1$.

Construct a BIJECTION, f, between the children of p_j and the children of q_j with the following properties:

[5]Inductively, this response is always consistent because responses are always sound.

(i) Map w_1 to the child w_1' of q_j which contains the responses to the pebbles currently present in $st(w_1)$.

(ii) If p_j and q_j have identical labels, then construct f so that it preserves labels.

(iii) Construct f so that w_2 gets mapped to a w_2' with the same label. (This is possible because whatever w_2's label is, there are at least two vertices among q_j's children with the same label, and at most one of these is already occupied by w_1'.)

Now, let $q_{j+1} = f(p_{j+1})$.

c If more than one sibling subtree of $st(w_1)$ is currently pebbled, then, if there is already a constructed bijection, f, play according to that bijection, i.e., let $q_{j+1} = f(p_{j+1})$.

If there is no constructed bijection, then if there is an empty child of q_j with the same label as p_{j+1}, let q_{j+1} be that child. Otherwise[6], let q_{j+1} be any empty child of q_j.

The following claim is the key to why Player II will win with this strategy:

Claim 4.8 *For $r \geq ip - jp - p + 1$, if for some l and some m, $a_s \in \{a_{l,m}, a_{l-1,m}\}$ pebbles a subtree $st(w)$*[7] *at level r such that no sibling subtree of $st(w)$ is pebbled by $a_{l-1,1}, \ldots, a_{l-1,c}, a_{l,1}, \ldots, a_{l,c}$ and $\bar{a}$, then the path name of the response b_s will be identical to that of a_s up to level r, i.e.:*

$$pn(a_s)_{r-1} = pn(b_s)_{r-1}$$

Proof This follows immediately from Player II's strategy. The lookahead in the strategy is designed specifically to force this condition. ∎

To finish the lemma, we need to prove:

Claim 4.9 *If $\bar{a} \approx_{ip-jp-p} \bar{b}$, then for all l, $\bar{a}, \overline{a_l}, \overline{a_{l+1}} \approx_{ip-jp} \bar{b}, \overline{b_l}, \overline{b_{l+1}}$.*

Proof We say that a level d is **established** when a pebble pair, $a_{l,m}$ and $b_{l,m}$, disagree in their path names at that level or below. In other words, $pn(a_{l,m})_{d-1} \neq pn(b_{l,m})_{d-1}$.

First, note that in order for a new level (i.e. lower than any level established before it) to be established, it must be the case that a pebble $a_{l,m}$ has moved into an empty subtree $st(w)$ at that level, for when moving into a non-empty subtree, say one that contains a_s, PLAYER II's response guarantees that if $b_{l,m}$ and $a_{l,m}$ disagree in their path name at the level of w, then so did a_s and b_s. But, note that since we assume that $\bar{a} \approx_{ip-jp-p} \bar{b}$ to begin with, p new levels need to be established with this transitive closure move in order for the conclusion not to hold. We will show that at least $p+1$ pebbles are needed to do this.

Suppose the conclusion doesn't hold. Look at the time when level $ip - jp + 1$ is established. It must be the case that PLAYER II just moved a pebble $b_{l,m}$ into an empty subtree $st(v_1')$ at level $ip - jp + 1$. It must also be the case that the path name of v_1' disagrees with its correspondent v_1 in A_{ip+1} on *every level* from $ip - jp + 1$ up to $ip - jp - p + 1$, for if they agree on some level t in between, the subtrees at level t containing $a_{l,m}$ and $b_{l,m}$ would be

[6]Note that this case can only arise if p_{j+1} is above level $ip - jp - p + 1$ in the tree.

[7]a_s might not be the only pebble in $st(w)$ among $a_{l-1,1}, \ldots, a_{l-1,c}, a_{l,1}, \ldots, a_{l,c}$ and $\bar{a}$.

isomorphic even in labels, and thus Player II's strategy would guarantee that no level below t is established by $a_{l,m}$ and $b_{l,m}$.

But then, by Claim 4.8, for each r such that $ip - jp + 1 \geq r \geq ip - jp - p + 1$, the subtree containing $a_{l,m}$ at level r must have a sibling subtree also pebbled. But this is impossible since there are only p pebbles in the game and this would require $p + 1$. ∎

Thus, by the induction hypothesis, $(A_{ip+1}, \bar{a}) \sim_{j+1,p} (B_{ip+1}, \bar{b})$. This concludes case 2, and that concludes the lemma. ∎

Observe that in the case where $\bar{a}$ and $\bar{b}$ are empty tuples, the lemma yields the theorem. ∎

Corollary 4.10 $\langle A_{ip+1}, B_{ip+1} \rangle \sim_{i,p} \langle A_{ip+1}, A_{ip+1} \rangle$

Proof The winning strategy for Player II is to copy the above winning strategy for the game on A_{ip+1}, B_{ip+1}, whenever Player I plays in the right-hand component, and to copy the identical element of A_{ip+1} whenever Player I plays in the left-hand component. Notice that this is still a winning strategy because (1) in counting moves, sets are still matched by sets of exactly the the same size; and (2) in the TC move, Player II's original strategy matches a path $\overline{a_1}, \ldots, \overline{a_d}$ with a path of the same length, $\overline{b_1}, \ldots, \overline{b_d}$, such that any consecutive pair, $(\overline{b_i}, \overline{b_{i+1}})$ is indistinguishable in the remaining moves from the pair, $(\overline{a_i}, \overline{a_{i+1}})$. It follows that it is no advantage to Player I to choose vertices from the left-hand component in such a path. Any such vertex will be answered by the identical vertex. ∎

5 Local Orderings

As noted in the Introduction, Theorem 1.2 refutes Conjecture 1.1[8]. We now prove Theorem 1.3, namely, that the graph properties expressible with the logic (FO+TC+COUNT+2LO) are exactly those in NL.

Proof of Theorem 1.3: The key lemma is the following:

Lemma 5.1 *Let $G = \langle N, V, E, c_1^G, \ldots, c_k^G \rangle$, where $V = \{v_1, \ldots, v_n\}$ and $N = \{1, 2, \ldots, n\}$. Then, there is a formula $E'(i,j)$, expressible in* (FO + TC + COUNT + 2LO)*, that defines a graph G' (over numbers) isomorphic to the input graph (over vertices). For each constant c of the vocabulary, there is a formula $V_c(i)$ which holds true for a unique i_c, and such that the isomorphism between G and G' maps i_c to c^G.*

Proof The idea for expressing E' is to

1. Totally order each (weakly) connected component of E, using the method of [EI95] (specifically, see Theorem 3.11 of that paper), which is based on distinguishing vertices by the lexicographically least shortest path that leads to them from a given vertex. To do so view all incoming edges as greater in the ordering than the outgoing edges.

[8]A refutation of Conjecture 1.1 can also be obtained as a corollary of the results of [CFI92]. A slight modification of the gadgets used there, from undirected gadgets to directed ones, yields this result.

2. Note that there is one ordering for each vertex in a component. For each connected component, choose the minimal one with respect to the lexicographic order of the adjacency sub-matrix.

3. Now order these components based on the following criteria (in decreasing order of significance):

   (a) Size of the component (non-decreasing order).

   (b) Lexicographic value of the minimal adjacency sub-matrix (non-decreasing order).

   (c) Containment of the constants $c_1, c_2, \ldots, c_k$ in the component (in increasing order, from c_1 to c_k and lastly none of them).

We have thus partitioned the graph into an ordered sequence of sets of components: $S_1, S_2, \ldots, S_r$, where each $S_j = \{C_1, \ldots, C_{i_j}\}$ is a set of one or more ordered, isomorphic components. We now define E' which puts an edge relation on the set of numbers N, depending on where that number sits in the ordering of all n vertices in $S_1, \ldots, S_r$. Note that even though the components in each S_j are not ordered, it doesn't matter. The point is that we have i_j identical copies of the relevant component. The unique place Counting is essential is in determining the numbers i_j. Now we provide the expression of E'.

$$
\begin{aligned}
E'(i,j) \equiv \exists x, y (E(x,y) \wedge \\
&\exists l, m((\exists ! l\ x' \ \preceq (x', x) \wedge \exists ! m\ y' \ \preceq (y', y)) \wedge \\
&\quad \exists b\ ((componentsize(x,b) \wedge componentsize(y,b)) \wedge \\
&\qquad \exists r\ ((incomparable(x,r) \wedge incomparable(y,r)) \wedge \\
&\qquad\quad \exists q\ (r = q * b \wedge \\
&\qquad\qquad (\exists d\ (0 \leq d \leq q))(i = l + d * b \ \wedge \ j = m + d * b)))
\end{aligned}
$$

The formula $\preceq (x,y)$ will be the partial order we obtain from the criteria outlined above. It is described below. $incomparable(x,r)$ means "There are exactly r vertices incomparable to x according to the partial ordering defined by $\preceq$", and it is expressed by:

$$incomparable(x,r) \equiv (\exists ! r\, y) \neg \preceq (x,y) \wedge \neg \preceq (y,x)$$

The only vertices which remain incomparable to x and y in this ordering are vertices in disjoint components isomorphic to the component x and y are in (clearly, if x and y have an edge to each other they are in the same component). Thus, counting the number of such incomparable vertices, the formula assures that i and j are in one component and are positioned where there is an edge, according to the possible orderings of the isomorphic components.

$$
\begin{aligned}
\preceq (x,y) \equiv \exists b_1, b_2\ ((componentsize(x,b_1) \wedge componentsize(y,b_2)) \wedge \\
&(b_1 < b_2 \vee (b_1 = b_2 \wedge mincomp \leq (x,y))))
\end{aligned}
$$

$$componentsize(x,b) \equiv \exists ! b\ y\ undirectedpath(x,y)$$

Here $mincomp \leq (x,y)$ means "the minimal adjacency matrix (minimal among the choices of ordering we have for the component) of the component of x is less than or equal to that of the component of y (lexicographically and in terms of containment of constants)." We leave it to the reader to verify that this formula is expressible using TC, counting quantifiers, and the component orderings constructible, via the [EI95] method from the available two-way local ordering.

Since each c^G is in a singleton component, its position is determined uniquely, and the formula $V_c(i)$ is easy to derive from the above construction.

∎

Thus, given an input graph with a two-way local ordering, we can define an isomorphic graph on an ordered set of numbers. Theorem 1.3 follows. ∎

6 Tree Canonization $\in (\mathrm{FO} + \mathrm{COUNT})[\log n]$

We obtain a $(\mathrm{FO} + \mathrm{COUNT})[\log n]$ formula for tree canonization. This improves on the previous best upper bound of $(\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT})$ [DM90] for tree canonization *without* ordering. Note that $(\mathrm{FO}+\mathrm{LFP}+\mathrm{COUNT}) = (\mathrm{FO}+\mathrm{COUNT})[n^{O(1)}]$. We also show that for bounded-degree tree isomorphism a $(\mathrm{FO})[\log n]$ formula is sufficient, i.e., without counting.

Let $|v|$ denote the number of nodes in $st(v)$. We use the following simple fact about trees:

Lemma 6.1 *For any tree T with root s, there is a unique node s', such that $|s'| \geq \lceil\frac{2}{3}|s|\rceil$ and such that for all children v of s', $|v| < \lceil\frac{2}{3}|s|\rceil$.*

Proof of Theorem 1.4: To find out whether trees rooted at a and b are isomorphic, we find the unique nodes a' and b' in $st(a)$ and $st(b)$, respectively, with the property of the lemma (this is done by existentially quantifying a' and b' and checking that they and their children satisfy the required subtree size properties).

We existentially quantify d and assert that d is equal to the distance between a and a' and and b and b' (d could be 0, i.e. a could equal a'). Otherwise, we will immediately know that the graphs are non-isomorphic.

Now, we *recursively* check that for each $0 \leq r \leq d$ for each vertex v which is a child off the path $a \rightsquigarrow a'$ (i.e., it is not itself on the path) at level r, $st(v)$ has exactly the same number of siblings subtrees (including itself) that are isomorphic to it as it has isomorphic copies of child subtrees off the path from $b \rightsquigarrow b'$ at level r. The only subtlety is that we need to do exact counting, whereas we only have the regular counting quantifiers available. In order to augment exact counting we will simultaneously build inductive definitions for tree-isomorphism *and* tree-non-isomorphism. For tree-non-isomorphism we use exactly the same 2/3-decomposition and look for a discrepancy. Exact counting can then be done by counting how many siblings are isomorphic and how many are non-isomorphic and making sure the sum adds up to the degree coming off the $a \rightsquigarrow a'$ path at the given level.

Recursively, this assures that the two original trees rooted at a and b are isomorphic. Furthermore, the recursive checks are all done on subtrees with less than 2/3 the size of the original tree, thus the $\log n$ bound on the depth of the formula.

We now give the formal expression of the $(\mathrm{FO} + \mathrm{COUNT})[\log n]$ formula.

We will give an inductive definition for a formula $iso(a, b)$, which determines whether the trees rooted at a and b are isomorphic. $iso()$ will use a relation $SpecialNode(a, a')$ which determines that a' is the unique node in the subtree of a with the property of Lemma 6.1. We will use Lemma 6.1 to conclude that the inductive definition will close after $O(\log n)$ iterations, establishing the theorem.

$$
\begin{aligned}
iso(a,b) \equiv \exists i\ (& (TreeSize(a,i) \wedge TreeSize(b,i)) \wedge \\
& (i = 1 \vee \\
& \exists a', b'\ (SpecialNode(a,a') \wedge SpecialNode(b,b') \wedge \\
& \exists d'\ (Dist(a,a',d') \wedge Dist(b,b',d')) \wedge \\
& \qquad \forall d, a'', b''\ ((Dist(a,a'',d) \wedge Dist(b,b'',d) \wedge \\
& \qquad\qquad Ances(a'',a') \wedge\ Ances(b'',b')\ ) \rightarrow \\
& \qquad\qquad \forall z\ ((Child(a'',z) \wedge \neg Ances(z,a')) \rightarrow \\
& \exists j(\exists! j x(Child(a'',x) \wedge \neg Ances(x,a') \wedge iso(x,z)) \wedge \\
& \quad \exists! j y(Child(b'',y) \wedge \neg Ances(y,b') \wedge iso(y,z))))))))
\end{aligned}
$$

$$
\begin{aligned}
SpecialNode(a,a') \quad \equiv \quad & path(a,a') \wedge |a'| \geq \lceil 2/3 \rceil |a| \wedge \\
& \forall c(Child(a',c) \rightarrow |c| < \lceil 2/3 \rceil |a|)
\end{aligned}
$$

$$
TreeSize(x,i) \quad \equiv \quad \exists! i\ x\ path(a,x)
$$

The $Dist(x, y, d)$ predicate means: “The distance from x to y is d”, and the $Ances(x, y)$ predicate means: “x is an ancestor of y”, and $Child(x, y)$ means “y is a child of x”. These simple predicates can very easily be expressed in FO$[\log n]$.

Note that in the above expressions we use *exact counting* in several places, i.e., expressions of the form $\exists! x\ \varphi(x)$. As was mentioned, we can augment these inductive definitions using only the usual counting quantifiers. To do so, we will simultaneously build inductive definitions for tree-isomorphism *and* tree-non-isomorphism. We then determine the exact number j of edges leaving a vertex and check that there are counts i and i' such that

$$
(\exists i\, x)\varphi(x) \ \wedge\ (\exists i'\, x)\neg\varphi(x) \ \wedge\ i + i' = j
$$

As mentioned, for tree-non-isomorphism we use exactly the same 2/3-decomposition and look for a discrepancy. The simultaneous inductive definition for ”non-iso” is the negation of the inductive definition for iso, where of course $\neg iso(x, y)$ is replaced by ”non-iso(x,y)” and $\neg non - iso(x, y)$ is replaced by $iso(x, y)$. The same sort of simultaneous inductive definition allows us to define path(x,y) and no-path(x,y).

Note that all our uses of the relation iso in the definition of iso are on vertices whose subtrees are at most 2/3 the size of the original subtrees rooted at a and b. Thus, since Lemma 6.1 guarantees the existence of the a' and b' claimed in the formula, we know that the inductive definition will close after at most $O(\log n)$ iterations, and we are done.

To express a canonical label for a tree we modify the above idea to express a canonical ordering relation $<'(a, b)$ on the vertices of a tree. Intuitively, the “canonical ordering” will consist of the following criteria (in decreasing order of significance):

1. $Size(st(a)) < Size(st(b))$

2. $depth(a') < depth(b')$, for a' and b' the vertices in the subtree of a and b, respectively, determined by Lemma 6.1

3. Walking down the path from a and b to a' and b', respectively, let depth d be the first place where there is any "difference". Then some smallest child z off the path from a to a' or b to b' has c isomorphic siblings as children of a'', but it has k isomorphic siblings as children of b'', and $c < k$.

Once we have the formal expression for $<'$, we use it to express a relation $E'(i,j)$ on numbers, which, like Lemma 5.1, will define a canonical tree isomorphic to the input tree E.

More formally, here is the expression[2] for $<'$, note the similarity to $iso(a,b)$:

$$
\begin{aligned}
<'(a,b) \equiv \exists i,j(&(TreeSize(a,i) \wedge TreeSize(b,j) \wedge i < j) \vee \\
&\exists a', b' \; ((SpecialNode(a,a') \wedge SpecialNode(b,b') \wedge \\
&\qquad Dist^*(a,a') < Dist^*(b,b')) \vee \\
&\qquad (Dist^*(a,a') = Dist^*(b,b') \wedge \exists d, a'', b'' \; (Dist(a,a'',d) \wedge \\
&\qquad\; Dist(b,b'',d) \;\wedge\; Ances(a'',a') \wedge\; Ances(b'',b') \;\wedge \\
&\qquad\; \exists z \; (((Child(a'',z) \wedge \neg Ances(z,a')) \vee (Child(b'',z) \wedge Ances(z,b'))) \wedge \\
&\exists c(\exists! c\, u(Child(a'',u) \wedge \neg Ances(u,a') \wedge iso(u,z)) \;\wedge \\
&\quad \exists! k\, w(Child(b'',w) \wedge \neg Ances(w,b') \wedge iso(w,z) \wedge c < k)))) \\
&\text{(and for smaller children at the level of } z \text{, or for higher children} \\
&\;\text{at depth less than } d \text{, there are the same number} \\
&\;\text{of isomorphic subtrees on both sides.)))}
\end{aligned}
$$

$<'(x,y)$ linearly orders trees. We can use $<'(x,y)$ to canonize a tree as follows: as in Lemma 5.1, we will express a relation $E'(i,j)$ on number variables i and j such that E' defines a canonical tree isomorphic to the input tree E. E' will be defined so that there is an edge from i to j iff the following conditions hold:

1. The i'th vertex in the canonical ordering has an edge to a vertex of the same isomorphism type as the j'th vertex.

2. There are m vertices of a type less than the i'th in the canonical ordering. There are m' numbers d less than i but such that the d'th vertex is of the same type as the i'th.

3. There are b vertices of the same type as the j'th which have an edge coming into them from a vertex of type less than the i'th vertex.

4. There are l vertices of the same type as the j'th to which a vertex of the same type as the i'th has an edge to.

5. The following bounds on j hold: $m + b + l(m') < j \leq m + b + l(m' + 1)$. The bounds garantee that the j'th vertex is among those vertices of the same type as the j'th which has an incoming edge from the i'th vertex.

[2]The $Dist^*(x,y)$ function in the expression is an abbreviation for use of the $Dist(x,y,d)$ relation.

We now give the formal definition of E'. The definition will contain the subformula: $Type(i,v) \equiv$ "The i'th vertex in the canonical ordering has a subtree isomorphic to that of vertex v", which we subsequently define.

$$\begin{aligned} E'(i,j) \equiv\ & \exists u,v(Type(i,u) \wedge Type(j,v) \wedge E(u,v)) \wedge \\ & \quad \exists m \exists! m\ x\ (x <' v) \wedge \\ & \qquad \exists b \exists! b y \exists z (z <' u \wedge E(z,y) \wedge iso(y,v)) \wedge \\ & \qquad\quad \exists m' \exists! m' k (k < i \wedge Type(k,u)) \wedge \\ & \qquad\qquad \exists l \exists! l w (E(u,w) \wedge iso(w,v)) \wedge \\ & \qquad\qquad\quad m + b + l(m') < j \wedge j \leq m + b + l(m'+1) \end{aligned}$$

$$\begin{aligned} Type(i,v) \equiv\ & \exists m \exists! m\ u\ (u <' v \wedge \\ & \qquad \exists r (\exists! r\ w\ iso(r,w) \wedge m < i \wedge i \leq m + r)) \end{aligned}$$

∎

Theorem 1.4 together with Theorem 1.2, yields Corollary 1.5. We finally show that for bounded degree trees, counting is not needed (from which Corollary 1.7 follows immediately):

Proof of Theorem 1.6 In order to convert the proof of Theorem 1.4 to this case it suffices to show that we can count the size of bounded degree subtrees in FO$[\log n]$. For once this is done, the "level-by-level" counting of the number of isomorphic subtrees just off the critical path from a to a' and b to b' can be done "by hand", i.e., quantifying the bounded number of possible vertices and checking that one of the various counting scenarios holds.

To count the size of a bounded degree subtree in FO$[\log n]$ we use the same 2/3-1/3 technique used by Ruzzo [Ruz80] to prove CFL's are recognizable by tree-size bounded alternating Turing machines, and, in the logical setting, used in [Imm82] (specifically, Theorem B.1) to prove a related result. Of course, when the degree bound is k, instead of a 2/3-1/3 lemma we have a $\frac{k}{k+1}$-$\frac{1}{k+1}$ lemma, but this is all that is required to get a $\log n$ bound on the depth of the required formula.

What is needed is to define $kBoundedTreeSize(x, y_1, y_2, \ldots, y_k, i)$, meaning "the size of the k-bounded degree tree rooted at x, excluding the subtrees rooted at $y_1, \ldots, y_k$ in the tree, is i." To define this we quantify a vertex y_{new} and check, recursively, that its subtree size is $1/3|x| \leq l \leq 2/3|x|$, and, again recursively, we check that $kBoundedTreeSize(x, y_1, y_2, \ldots, y_k, y_{new}, l')$ holds, and we also check that $l + l' = i$. Of course, we can not just keep increasing the arity of our relation like that, so whenever our list of removed nodes exceeds k, we look for a vertex y' that is an ancestor of exactly m of the $k+1$ vertices $2 \leq m \leq k$. Such a vertex must exist by induction (walk down from the root of the tree and use the pigeon-hole principle). We then recursively find the size of the tree rooted at y' and add it to the size of the tree that also excludes y' (but we no longer need to exclude the other y's that descend from y' thus we always only keep track of k excluded vertices).

Bounded-degree tree canonization can also be expressed in FO$[\log n]$ using a variant of the canonization technique in Theorem 1.4. The following claim provides what is needed.

Claim 6.2 *Given a formula $\varphi(x)$ expressible in* FO$[\log n]$ *over k-bounded degree (directed) trees, there is a formula $\psi(i)$ expressible in* FO$[\log n]$ *such that*

$$\psi(i) \iff \exists! ix\varphi(x)$$

Proof The proof is another application of Ruzzo's technique [Ruz80]. We quantify the root of the tree and count the number of nodes in its subtree (all vertices) satisfy the property φ. The only difference between this and the previous case is that we are counting only the vertices that satisfy property φ which is itself an FO$[\log n]$ formula, so our inductive depth can increase by (at most) a factor of 2. ∎

Once we have the claim, note that the expression for $E'(i,j)$ in Theorem 1.4 can be converted to an FO$[\log n]$ expression on k-bounded degree trees by inductively eliminating all the counting quantifiers. Note however that, whereas Theorem 1.4 works for both trees and *forests*, the proofs here work only for trees because counting is not available when all vertices do not belong to the same subtree. ∎

7 Conclusion

We have separated the unordered versions of the logics for ThC^1, AC^1, and NL using very natural problems: tree-isomorphism and bounded-degree tree-isomorphism. We have observed that these separations also hold in the presence of *one-way* local ordering. Recently, L. Hella and H. Imhof [Hel96] have extended our results to show that tree isomorphism is not expressible in some even stronger logics, in particular in the logic augmented with the alternating transitive closure (ATC) operator ([Imm87]).

The key remaining open problems deal with *two-way* local ordering and they take on increased significance due to Theorem 1.3:

1. *Separate* (FO + DTC + COUNT + 2LO) *from* (FO + TC + COUNT + 2LO).
2. *Are there any Logspace graph properties not in* (FO + DTC + COUNT + 2LO)*?*[3]
3. Tree isomorphism is a natural and order-independent property. What natural properties can we add to (FO + TC + COUNT) in order to express properties like tree isomorphism, and what does the addition buy us in expressive power? In particular, what do we gain by adding an operator for tree canonization? Do we get much closer to the order-independent complexity class? We would conjecture that properties derived from the results of [CFI92] would still not be expressible in this extended logic.

Note that a polynomial length universal traversal sequence, which exists by the results of [AKL$^+$79], gives the logic (FO + DTC + COUNT + 2LO) a systematic way to reach each node, and thus a way to construct a total ordering of all weakly connected components (based on when a vertex is first hit), and thus a canonization algorithm. It follows that a non-uniform version of (FO + DTC + COUNT + 2LO) includes all of L. Thus a lower bound

[3]Tree-isomorphism-like properties are not a candidate for this, because Lindell's result [Lin92] goes through with 2LO regardless of the edge directions.

on the uniform language (FO + DTC + COUNT + 2LO) will be very interesting and will likely be quite difficult.

As yet, there are no non-trivial lower bounds for transitive closure logics with two-way local ordering and numbers, even without counting. The best that is known is a separation of (FO + COUNT + 2LO) from (FO + DTC + 2LO), implicit in the lower bound of [Ete95].

Acknowledgements

Thanks to Allan Borodin and Steven Lindell for various helpful discussions. Thanks to Janos Makowsky and to the anonymous referees for pointing out an error in our earlier proof of Corollary 4.10.

References

[AKL$^+$79] R. Aleliunas, R. Karp, R. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of the maze problem. In *20th IEEE Found. of Comp. Sci. Symp.*, pages 218–223, 1979.

[BIS90] D. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *JCSS*, 41:274–306, 1990.

[CFI92] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, **12** (4):389–410, 1992.

[CM92] A. Calò and J. Makowsky. The Ehrenfeucht-Fraïssé games for transitive closure. In *2nd Inter. Symp. on Logical Foundations of Computer Science*, 1992.

[CR80] S. A. Cook and C. W. Rackoff. Space lower bounds for maze threadability of restricted machines. *SIAM J. Comput.*, 9 (3):636–652, 1980.

[DM90] P. Dublish and S. Maheshwari. Query languages which express all ptime queries for trees and unicyclic graphs. In *Proc. of MFCS 90 LNCS vol. 452*, 1990.

[EI95] K. Etessami and N. Immerman. Reachability and the power of local ordering. *Theoretical Computer Science*, 148:227–260, 1995.

[Ete95] K. Etessami. Counting quantifiers, successor relations, and logarithmic space. In *10th Structure in Complexity Theory Conf.*, pages 1–10, 1995. to appear in JCSS.

[Grä91] E. Grädel. On transitive closure logic. In *Proceed. 5th Workshop on Comp. Sci. Logic*, volume 626 of *LNCS*, pages 149–163, 1991.

[Hel96] L. Hella. personal communication. 1996.

[IL90] N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In Alan Selman, editor, *Complexity Theory Retrospective*, pages 59–81. Springer-Verlag, 1990.

[Imm82] N. Immerman. Upper and lower bounds for first order expressibility. *JCSS*, 25 (1):76–98, 1982.

[Imm87] N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16:4:760–778, 1987.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17:5:935–938, 1988.

[Imm89] N. Immerman. Expressibility and parallel complexity. *SIAM J. of Comput.*, 18:625–638, 1989.

[Imm96] N. Immerman. *Descriptive Complexity.* Springer-Verlag, New York, to appear 1996.

[Lin92] S. Lindell. A logspace algorithm for tree canonization. In *24th Symp. on Theory of Comput.*, pages 400–404, 1992.

[Ott96] M. Otto. The expressive power of fixed-point logic with counting. *Journal of Symbolic Logic*, 61(1):147–176, 1996.

[Ruz80] W.-L. Ruzzo. Tree-size bounded alternation. *Journal of Computer and System Sciences*, 21, 1980.