L-Recursion and a new Logic for Logarithmic Space

Martin Grohe, Berit Grußien, André Hernich, and Bastian Laubner

Humboldt University Berlin
Germany
`{grohe,grussien,hernich,laubner}@informatik.hu-berlin.de`

Abstract

We extend first-order logic with counting by a new operator that allows it to formalise a limited form of recursion which can be evaluated in logarithmic space. The resulting logic LREC has a data complexity in LOGSPACE, and it defines LOGSPACE-complete problems like deterministic reachability and Boolean formula evaluation. We prove that LREC is strictly more expressive than deterministic transitive closure logic with counting and incomparable in expressive power with symmetric transitive closure logic STC and transitive closure logic (with or without counting). LREC is strictly contained in fixed-point logic with counting FP+C. We also study an extension $\mathsf{LREC}_=$ of LREC that has nicer closure properties and is more expressive than both LREC and STC, but is still contained in FP+C and has a data complexity in LOGSPACE.

Our main results are that LREC captures LOGSPACE on the class of directed trees and that $\mathsf{LREC}_=$ captures LOGSPACE on the class of interval graphs.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes; F.4.1 Mathematical Logic

Keywords and phrases Descriptive complexity, logarithmic space, fixed-point logics

Digital Object Identifier 10.4230/LIPIcs.CSL.2011.277

1 Introduction

Descriptive complexity theory gives logical characterisations for most of the standard complexity classes. For example, Fagin's Theorem [6] states that a property of finite structures is decidable in NP if and only if it is definable in existential second-order logic Σ_1^1. More concisely, we say that Σ_1^1 *captures* NP. Similarly, Immerman [11] and Vardi [23] proved that fixed-point logic FP captures PTIME,[1] and Immerman [13] proved that deterministic transitive closure logic DTC captures LOGSPACE. However, these and all other known logical characterisations of PTIME and LOGSPACE and all other complexity classes below NP have a serious drawback — they only hold on ordered structures. (An *ordered structure* is a structure that has a distinguished binary relation which is a linear order of the elements of the structure.) The question of whether there are logical characterisations of these complexity classes on arbitrary, not necessarily ordered structures, is viewed as the most important open problem in descriptive complexity theory. For the class PTIME this problem goes back to Chandra and Harel's fundamental article [3] on query languages for relational databases.

[1] More precisely, Immerman and Vardi's theorem holds for *least fixed-point logic* and the equally expressive *inflationary fixed-point logic*. Our indeterminate FP refers to either of the two logics. For the counting extension FP+C considered below, it is most convenient to use an inflationary fixed-point operator. See any of the textbooks [4, 9, 14, 20] for details.

For PTIME, at least partial positive results are known. The strongest of these say that fixed-point logic with counting FP+C captures PTIME on all classes of graphs with excluded minors [10] and on the class of interval graphs [17]. It is well-known that fixed-point logic FP (without counting) is too weak to capture PTIME on any natural class of structures that are not ordered. The idea that the extension FP+C by counting operators might remedy the weakness of FP goes back to Immerman [12]. Together with Lander he proved that FP+C captures PTIME on the class of trees [15]. Later, Cai, Fürer, and Immerman [2] proved that FP+C does not capture PTIME on all finite structures.

Much less is known for LOGSPACE. In view of the results described so far, an obvious idea is to try to capture LOGSPACE with the extension DTC+C of deterministic transitive closure logic DTC by counting operators. However, Etessami and Immerman [5] proved that (directed) tree isomorphism is not definable in DTC+C, not even in the stronger transitive closure logic with counting TC+C. Since Lindell [21] proved that tree isomorphism is decidable in LOGSPACE, this shows that DTC+C does not capture LOGSPACE.

We introduce a new logic LREC and prove that it captures LOGSPACE on directed trees. An extension $\mathsf{LREC}_=$ captures LOGSPACE on the class of interval graphs (and on the class of undirected trees). The logic LREC is an extension of first-order logic with counting by a "limited recursion operator". The logic is more complicated than the transitive closure and fixed-point logics commonly studied in descriptive complexity, and it may look rather artificial at first sight. To explain the motivation for this logic, recall that fixed-point logics may be viewed as extensions of first-order logic by fixed-point operators that allow it to formalise recursive definitions in the logics. LREC is based on an analysis of the amount of recursion allowed in logarithmic space computations. The idea of the limited recursion operator is to control the depth of the recursion by a "resource term", thereby making sure that we can evaluate the recursive definition in logarithmic space. Another way to arrive at the logic is based on an analysis of the classes of Boolean circuits that can be evaluated in LOGSPACE. We will take this route when we introduce the logic in Section 3.

LREC is easily seen to be (semantically) contained in FP+C. We show that LREC contains DTC+C, and as LREC captures LOGSPACE on directed trees, this containment is strict and, moreover, LREC is not contained in TC+C. Then we prove that undirected graph reachability is not definable in LREC. Hence LREC does not contain transitive closure logic TC, not even in its symmetric variant STC, and therefore LREC is strictly contained in FP+C.

It can be argued that our proof of the inability of LREC to express graph reachability reveals a weakness in our definition of the logic rather than a weakness of the limited recursion operator underlying the logic: LREC is not closed under (first-order) logical reductions. To remedy this weakness, we introduce an extension $\mathsf{LREC}_=$ of LREC. It turns out that undirected graph reachability is definable in $\mathsf{LREC}_=$ (this is a convenient side effect of the definition and not a deep result). Thus $\mathsf{LREC}_=$ strictly contains symmetric transitive closure logic with counting. We prove that $\mathsf{LREC}_=$ captures LOGSPACE on the class of interval graphs. To complete the picture, we prove that plain LREC, even if extended by a symmetric transitive closure operator, does not capture LOGSPACE on the class of interval graphs.

The paper is organised as follows: After giving the necessary preliminaries in Section 2, in Section 3 we introduce the logic LREC and prove that its data complexity is in LOGSPACE. Then in Section 4, we prove that directed tree isomorphism and canonisation are definable in LREC. As a consequence, LREC captures LOGSPACE on directed trees. In Section 5, we study the expressive power of LREC and prove that undirected graph reachability is not definable in LREC. The extension $\mathsf{LREC}_=$ is introduced in Section 6. Finally, our results on interval graphs are presented in Section 7. We close with concluding remarks and open problems. Due to space limitations, we defer many of the proofs to the full version of this paper.

2 Basic Definitions

$\mathbb{N}$ denotes the set of all non-negative integers. For all $m, n \in \mathbb{N}$, we define $[m, n] := \{p \in \mathbb{N} \mid m \leq p \leq n\}$, and $[n] := [1, n]$. For mappings $f : A \to B$, and tuples $\bar{a} = (a_1, \dots, a_k)$ over A, we let $f(\bar{a}) := (f(a_1), \dots, f(a_k))$. For a tuple $\bar{a} = (a_1, \dots, a_k)$, we let $\tilde{a} := \{a_1, \dots, a_k\}$.

A *vocabulary* is a finite set τ of relation symbols, where each $R \in \tau$ has a fixed arity $\mathrm{ar}(R)$. A *τ-structure* A consists of a non-empty finite set $V(A)$, its *universe*, and for each $R \in \tau$ a relation $R(A) \subseteq V(A)^{\mathrm{ar}(R)}$. For logics L, L' we write $\mathsf{L} \leq \mathsf{L}'$ if L is semantically contained in L', and $\mathsf{L} < \mathsf{L}'$ if this containment is strict.

All logics considered in this paper are extensions of *first-order logic with counting (*FO+C*)*; see, e.g., [4, 9, 14, 20] for a detailed discussion of FO+C and its extensions. FO+C extends first-order logic by a counting operator that allows for counting the cardinality of FO+C-definable relations. It lives in a two-sorted context, where structures A are equipped with a *number sort* $N(A) := [0, |V(A)|]$. FO+C-variables are either *structure variables* that range over the universe of a structure, or *number variables* that range over the number sort. For each variable u, let $A^u := V(A)$ if u is a structure variable, and $A^u := N(A)$ if u is a number variable. Tuples $(u_1, \dots, u_k)$ and $(v_1, \dots, v_\ell)$ of variables are *compatible* if $k = \ell$, and for every $i \in [k]$ the variables u_i and v_i have the same type. Let $A^{(u_1,\dots,u_k)} := A^{u_1} \times \dots \times A^{u_k}$. An *assignment in* A is a mapping α from the set of variables to $V(A) \cup N(A)$, where for each variable u we have $\alpha(u) \in A^u$. For tuples $\bar{u} = (u_1, \dots, u_k)$ of variables and $\bar{a} = (a_1, \dots, a_k) \in A^{\bar{u}}$, the assignment $\alpha[\bar{a}/\bar{u}]$ maps u_i to a_i for each $i \in [k]$, and each variable $v \notin \tilde{u}$ to $\alpha(v)$.

FO+C is obtained by extending first-order logic with the following formula formation rules: $p \leq q$ is a formula for all number variables p, q; and $\#\bar{u}\,\psi = \bar{p}$ is a formula for all tuples $\bar{u}$ of variables, all tuples $\bar{p}$ of number variables, and all formulae ψ. Free variables are defined in the obvious way, with $\mathrm{free}(\#\bar{u}\,\psi = \bar{p}) := (\mathrm{free}(\psi) \setminus \tilde{u}) \cup \tilde{p}$. Formulas $\#\bar{u}\,\psi = \bar{p}$ hold in a structure A under an assignment α in A if $|\{\bar{a} \in A^{\bar{u}} \mid (A, \alpha[\bar{a}/\bar{u}]) \models \psi\}| = \langle\alpha(\bar{p})\rangle_A$, where for tuples $\bar{n} = (n_1, \dots, n_k) \in N(A)^k$ we let $\langle\bar{n}\rangle_A$ be the number

$$\langle\bar{n}\rangle_A := \sum_{i=1}^{k} n_i \cdot (|V(A)| + 1)^{i-1}.$$

If A is understood from the context, we write $\langle\bar{n}\rangle$ instead of $\langle\bar{n}\rangle_A$.

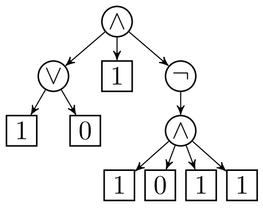
We write $\varphi(u_1, \dots, u_k)$ to denote a formula φ with $\mathrm{free}(\varphi) \subseteq \{u_1, \dots, u_k\}$. Given a formula $\varphi(u_1, \dots, u_k)$, a structure A and $a_1, \dots, a_k \in A^{(u_1,\dots,u_k)}$, we write $A \models \varphi[a_1, \dots, a_k]$ if φ holds in A with u_i assigned to the element a_i, for each $i \in [k]$. We use similar notation for substitution: For a tuple $(v_1, \dots, v_k)$ of variables that is compatible to $(u_1, \dots, u_k)$, we let $\varphi(v_1, \dots, v_k)$ be the result of substituting v_i for u_i for every $i \in [k]$. We write $\varphi[A, \alpha; \bar{u}]$ for the set of all tuples $\bar{a} \in A^{\bar{u}}$ with $(A, \alpha[\bar{a}/\bar{u}]) \models \varphi$.

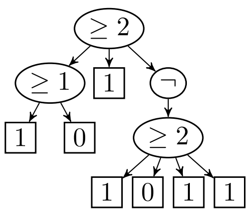
In many places throughout this paper we refer to various transitive closure and fixed-point logics (all mentioned in the introduction). Our results and remarks about the relation between these logics and our new logics LREC and $\mathsf{LREC}_=$ are relevant for a reader familiar with descriptive complexity theory to put our results in context, but they are not essential to follow the technical core of this paper. Therefore, we omit the definitions and refer the reader to the textbooks [4, 9, 14, 20].

3 The Logic LREC

Let us start our development of LREC by looking at how certain kinds of Boolean circuits can be evaluated in LOGSPACE.

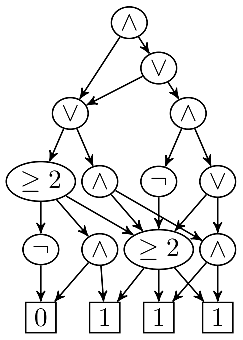
The figure on the right shows a *Boolean formula*, i.e., a Boolean circuit whose underlying graph is a *tree*. It is easy to evaluate such circuits in LOGSPACE: Start at the output node, determine the value of the first child recursively, then determine the value of the second child, and so on. We only have to store the current node and its value (if it has been determined already), since the parent node and the next child of the parent (if any) are uniquely determined by the current node. It is known that Boolean formula evaluation is LOGSPACE-complete under NC^1-reductions [1].[2] In contrast, Boolean *circuit* evaluation is PTIME-complete.





Let us now turn to formulas with *threshold gates*, which may contain gates of the form "$\geq i$" for a number i in addition to the Boolean gates. An example is shown on the left. To evaluate such formulas in LOGSPACE, we again start at the root and evaluate the values of the children recursively. For each node we count how many 1-values we have seen already. To this end, when evaluating the values of the children of a node v, we begin with the child with the largest subtree and proceed to children with smaller subtrees. Note that the ith child of v in this order has a subtree of size at most s/i, where s is the size of the subtree of v. So, we can store a counter of up to $\log_2 i$ bits for the number of 1-values seen so far. It is easy to extend the algorithm to formulas with other *arithmetic gates* such as *modulo-gates*.

As a more complicated example, let us consider the following circuits. A circuit C has the *m-path property* if for all paths P in C the product of the in-degrees of the nodes on P is at most m. For example, formulas have the 1-path property, whereas the circuit on the right has the 16-path property. It is not hard to see that for every $k \geq 1$, circuits C having the $|C|^k$-path property can be evaluated in LOGSPACE. The $|C|^k$-path property here guarantees that in addition to a counter we can also store the path from the current node to the root, so that we can always find the parent of the current node. Another way of evaluating the circuit is to first "unravel" the circuit to a tree (i.e., a formula) which can be done in LOGSPACE due to the $|C|^k$-path property, and then to evaluate the formula as above.



The logic LREC allows it to recursively define sets X of tuples based on graphs G that have the $|G|^k$-path property for some $k \geq 1$.

We turn to the formal definition of the logic LREC. To define the syntax, let τ be a vocabulary. The set of all LREC[τ]-formulae is obtained by extending the formula formation rules of FO+C[τ] by the following rule: If $\bar{u}, \bar{v}, \bar{w}$ are compatible tuples of variables, $\bar{p}, \bar{r}$ are non-empty tuples of number variables, and φ_E and φ_C are LREC[τ]-formulae, then

$$\varphi := [\mathsf{lrec}_{\bar{u},\bar{v},\bar{p}}\ \varphi_E,\ \varphi_C](\bar{w},\bar{r}) \tag{1}$$

is an LREC[τ]-formula, and we let $\mathrm{free}(\varphi) := (\mathrm{free}(\varphi_E) \setminus (\tilde{u} \cup \tilde{v})) \cup (\mathrm{free}(\varphi_C) \setminus (\tilde{u} \cup \tilde{p})) \cup \tilde{w} \cup \tilde{r}$.

To define the semantics of LREC[τ]-formulae, let A be a τ-structure and α an assignment in A. The semantics of LREC[τ]-formulae that are not of the form (1) is defined as usual.

Let φ be an LREC[τ]-formula of the form (1). We define a set $X \subseteq A^{\bar{u}} \times \mathbb{N}$ recursively as follows. We consider $E := \varphi_E[A, \alpha; \bar{u}, \bar{v}]$ as the edge relation of a directed graph with

[2] Here, the Boolean formula is represented by the list of all edges plus gate types in the circuit representing the formula.

vertex set $V := A^{\bar{u}}$. Moreover, for each vertex $\bar{a} \in V$ we think of the set $C(\bar{a}) := \{\langle \bar{n} \rangle \mid \bar{n} \in \varphi_C[A, \alpha[\bar{a}/\bar{u}]; \bar{p}]\}$ of integers as the label of $\bar{a}$. Let $\bar{a}E := \{\bar{b} \in V \mid \bar{a}\bar{b} \in E\}$ and $E\bar{b} := \{\bar{a} \in V \mid \bar{a}\bar{b} \in E\}$. Then, for all $\bar{a} \in V$ and $\ell \in \mathbb{N}$,

$$(\bar{a}, \ell) \in X \;:\Longleftrightarrow\; \ell > 0 \text{ and } \left| \left\{ \bar{b} \in \bar{a}E \;\middle|\; \left(\bar{b}, \left\lfloor \frac{\ell - 1}{|E\bar{b}|} \right\rfloor \right) \in X \right\} \right| \in C(\bar{a}).$$

Notice that X contains only elements $(\bar{a}, \ell)$ with $\ell > 0$. Hence, the recursion eventually stops at $\ell = 0$. We call X the *relation defined by φ in (A, α)*. Finally, we let

$$(A, \alpha) \models \varphi \;:\Longleftrightarrow\; (\alpha(\bar{w}), \langle \alpha(\bar{r}) \rangle) \in X.$$

▶ **Example 3.1** (Boolean circuit evaluation). Let $\sigma := \{E, P_\wedge, P_\vee, P_\neg, P_0, P_1\}$. A Boolean circuit C may be viewed as a σ-structure, where $E(C)$ is the edge relation of C, and $P_\star(C)$ contains all $\star$-gates for $\star \in \{\wedge, \vee, \neg, 0, 1\}$. If C has the $|C|$-path-property, then $\exists r_1, r_2\, [\mathsf{lrec}_{x,y,p}\; E(x,y),\; \varphi_C](z, (r_1, r_2))$ with $\varphi_C(x,p) := (P_\wedge(x) \wedge \#y\, E(x,y) = p) \vee (P_\vee(x) \wedge$ "$p > 0$"$) \vee (P_\neg(x) \wedge$ "$p = 0$"$) \vee P_1(x)$ states that gate z evaluates to 1. ◀

▶ **Example 3.2** (Deterministic transitive closure). Let $\psi(\bar{u}, \bar{v})$ be an $\mathsf{LREC}[\tau]$-formula, and let $\bar{s}, \bar{t}$ be tuples of variables such that $\bar{u}, \bar{v}, \bar{s}, \bar{t}$ are pairwise compatible. We give a formula φ such that for any τ-structure A and assignment α in A, we have $(A, \alpha) \models \varphi(s, t)$ iff in the graph $G = (V, E)$ defined by $V := A^{\bar{u}}$ and $E := \psi[A, \alpha; \bar{u}, \bar{v}]$ there is a *deterministic path* from $\alpha(\bar{s})$ to $\alpha(\bar{t})$, i.e., a path $v_1, \ldots, v_n$ from $\alpha(\bar{s})$ to $\alpha(\bar{t})$ such that for every $i \in [n-1]$, v_{i+1} is the unique out-neighbour of v_i. This is the same as reversing the edges of G and finding a path $v_n, \ldots, v_1$ from $\alpha(\bar{t})$ to $\alpha(\bar{s})$ such that for every $i \in [n-1]$, v_{i+1} is the unique in-neighbour of v_i. Therefore,

$$\varphi \;:=\; \exists \bar{r}\, [\mathsf{lrec}_{\bar{u},\bar{v},\bar{p}}\; \varphi_E(\bar{u}, \bar{v}),\; \varphi_C(\bar{u}, \bar{p})](\bar{t}, \bar{r}), \tag{2}$$

where $\bar{p}$ and $\bar{r}$ are $(|\bar{u}| + 1)$-tuples of number variables, and

$$\varphi_E(\bar{u}, \bar{v}) \;:=\; \psi(\bar{v}, \bar{u}) \wedge \forall \bar{u}' (\psi(\bar{v}, \bar{u}') \to \bar{u}' = \bar{u}), \qquad \varphi_C(\bar{u}, \bar{p}) \;:=\; \bar{u} = \bar{s} \vee (\bar{u} \neq \bar{s} \wedge \bar{p} \neq \bar{0}).$$

In the following, we use $[\mathsf{dtc}_{\bar{u},\bar{v}}\; \psi](\bar{s}, \bar{t})$ as an abbreviation for the LREC-formula in (2). ◀

The following theorem shows that the data complexity of LREC is in $\mathsf{LOGSPACE}$.

▶ **Theorem 3.3.** *For every vocabulary τ, and every $\mathsf{LREC}[\tau]$-formula φ there is a deterministic logspace Turing machine that, given a τ-structure A and an assignment α in A, decides whether $(A, \alpha) \models \varphi$.*

▶ Remark. It follows from Example 3.2 that $\mathsf{DTC{+}C} \leq \mathsf{LREC}$. This containment is strict as directed tree isomorphism is definable in LREC (we will show this in the next section), but not in $\mathsf{DTC{+}C}$. On the other hand, it is easy to see that the relation X defined by an LREC-formula of the form (1) in an interpretation (A, α) can be defined in fixed point logic with counting $\mathsf{FP{+}C}$. Hence, $\mathsf{LREC} \leq \mathsf{FP{+}C}$, and this containment is strict since we show in Section 5 that undirected graph reachability is not LREC-definable.

4 Capturing Logspace on Directed Trees

In this section we show that LREC captures $\mathsf{LOGSPACE}$ on the class of all directed trees. Our construction is based on Lindell's $\mathsf{LOGSPACE}$ tree canonisation algorithm [21]. Note, however, that Lindell's algorithm makes essential use of a linear order on the tree's vertices that is given implicitly by the encoding of the tree. Here we do not have such a linear order, so we cannot directly translate Lindell's algorithm to an LREC-formula. We show that we can circumvent using the linear order if we have a formula for directed tree isomorphism. Hence, our first task is to construct such a formula.

4.1 Directed Tree Isomorphism

Let T be a directed tree. For every $v \in V(T)$ let T_v be the subtree of T rooted at v, let $\text{size}(v) := |V(T_v)|$ be the *size* of v, and let $\#_s(v)$ be the number of children of v of size s. We construct an LREC$[\{E\}]$-formula $\varphi_{\cong}(x, y)$ that is true in a directed tree T with interpretations $v, w \in V(T)$ of x, y if and only if $T_v \cong T_w$. We assume that $|V(T)| \geq 4$, but it is easy to adapt the construction to directed trees with less than 4 vertices.

We implement the following recursive procedure to check whether $T_v \cong T_w$:

1. If $\text{size}(v) \neq \text{size}(w)$ or if $\#_s(v) \neq \#_s(w)$ for some $s \in [0, |V(T_v)| - 1]$, return "$T_v \ncong T_w$".
2. If for all children $\hat{v}$ of v there is a child $\hat{w}$ of w and a number k such that
   a. $T_{\hat{v}} \cong T_{\hat{w}}$,
   b. there are exactly k children $\mathring{w}$ of w with $T_{\hat{v}} \cong T_{\mathring{w}}$, and
   c. there are exactly k children $\mathring{v}$ of v with $T_{\mathring{v}} \cong T_{\hat{w}}$,

   then return "$T_v \cong T_w$".
3. Return "$T_v \ncong T_w$".

Clearly, this procedure outputs "$T_v \cong T_w$" if and only if $T_v \cong T_w$.

To simplify the presentation we fix a directed tree T and an assignment α in T, but the construction will be uniform in T and α.

We construct a directed graph $G = (V, E)$ with labels $C(v) \subseteq \mathbb{N}$ for each $v \in V$ as follows. Let $V := N(T) \times V(T)^4 \times N(T)$. The first component of each vertex is its *type*; the meaning of the other components will become clear soon. Although G will not be a tree, it is helpful to think of it as a *decision tree* for deciding $T_v \cong T_w$. For each pair $(v, w) \in V(T)^2$, we designate the vertex $\bar{a}_{v,w} = (0, v, w, v, w, 0)$ to stand for "$T_v \cong T_w$". Let us call (v, w) *easy* if v, w satisfy the condition in line 1 of the procedure (i.e., $\text{size}(v) \neq \text{size}(w)$, or $\#_s(v) \neq \#_s(w)$ for some $s \in [0, |V(T_v)| - 1]$). Note that the set of all such easy pairs is LREC-definable.[3] If (v, w) is easy, then $\bar{a}_{v,w}$ has no outgoing edges and $C(\bar{a}_{v,w}) = \emptyset$. On the other hand, if (v, w) is not easy, then G contains the following edges and labels (see Figure 1 for an illustration):

- The vertex $\bar{a}_{v,w}$ has an outgoing edge to $\bar{a}_{v,w,\hat{v}} := (1, v, w, \hat{v}, w, 0)$, for each child $\hat{v}$ of v. Furthermore, $C(\bar{a}_{v,w}) = \{\#\text{ of children of } v\}$. This corresponds to "for all children $\hat{v}$ of v…" in the above procedure's step 2.
- The vertex $\bar{a}_{v,w,\hat{v}}$ has an outgoing edge to $\bar{a}_{v,w,\hat{v},\hat{w},k} := (2, v, w, \hat{v}, \hat{w}, k)$, for each child $\hat{w}$ of w with $\text{size}(\hat{w}) = \text{size}(\hat{v})$ and each $k \in [\#_{\text{size}(\hat{v})}(v)]$. Furthermore, $C(\bar{a}_{v,w,\hat{v}}) = N(T) \setminus \{0\}$. This branching corresponds to "…there is a child $\hat{w}$ of w and a number k such that…".
- The vertex $\bar{a}_{v,w,\hat{v},\hat{w},k}$ has an outgoing edge to $\bar{a}_{\hat{v},\hat{w}}$. If $\hat{v}$ is the only child of v of size $\text{size}(\hat{v})$, then this is the only outgoing edge, and we let $C(\bar{a}_{v,w,\hat{v},\hat{w},k}) = \{1\}$. Otherwise, there are additional outgoing edges to $\bar{a}^i_{v,w,\hat{v},\hat{w},k} = (3 + i, v, w, \hat{v}, \hat{w}, k)$ for $i \in \{0, 1\}$, and we let $C(\bar{a}_{v,w,\hat{v},\hat{w},k}) = \{3\}$. This corresponds to conditions 2a–2c.
- The vertex $\bar{a}^0_{v,w,\hat{v},\hat{w},k}$ has outgoing edges to $\bar{a}_{\hat{v},\mathring{w}}$ for each child $\mathring{w}$ of w of size $\text{size}(\hat{v})$, and $\bar{a}^1_{v,w,\hat{v},\hat{w},k}$ has outgoing edges to $\bar{a}_{\mathring{v},\hat{w}}$ for each child $\mathring{v}$ of v of size $\text{size}(\hat{w}) = \text{size}(\hat{v})$.

[3] Using the dtc-operator from Example 3.2 we can construct an LREC$[\{E\}]$-formula defining the descendant relation between vertices in a directed tree, and using this formula it is easy to determine the size and the number of children of size s of a vertex.

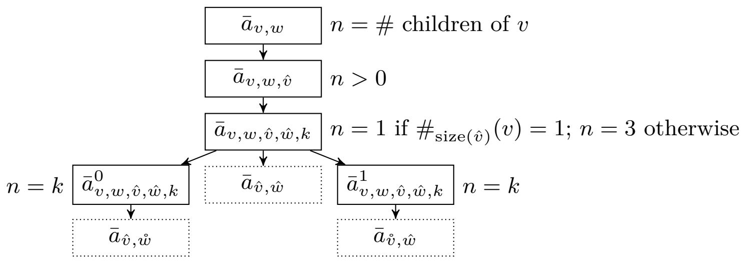

$\bar{a}_{v,w}$
$n = \#$ children of v
$\bar{a}_{v,w,\hat{v}}$
$n > 0$
$\bar{a}_{v,w,\hat{v},\hat{w},k}$
$n = 1$ if $\#_{\mathrm{size}(\hat{v})}(v) = 1$; $n = 3$ otherwise
$n = k$
$\bar{a}^0_{v,w,\hat{v},\hat{w},k}$
$\bar{a}_{\hat{v},\hat{w}}$
$\bar{a}^1_{v,w,\hat{v},\hat{w},k}$
$n = k$
$\bar{a}_{\mathring{v},\hat{w}}$
$\bar{a}_{\hat{v},\mathring{w}}$

Figure 1 Sketch of "decision tree" for deciding $T_v \cong T_w$. Here, $\hat{v}, \mathring{v}$ range over the children of v; $\hat{w}, \mathring{w}$ range over the children of w; and $k \in [\#_{\mathrm{size}(\hat{v})}(v)]$. Moreover, $\hat{v}, \mathring{v}, \hat{w}, \mathring{w}$ all have the same size. Labels indicate which integers n belong to the set $C(\bar{a})$ labelling each vertex $\bar{a}$. If $\hat{v}$ is the only child of v of size size$(\hat{v})$, then $\bar{a}_{\hat{v},\hat{w}}$ is the only child of $\bar{a}_{v,w,\hat{v},\hat{w},k}$.

Furthermore, $C(\bar{a}^i_{v,w,\hat{v},\hat{w},k}) = \{k\}$. The vertex $\bar{a}^i_{v,w,\hat{v},\hat{w},k}$ corresponds to condition 2b for $i = 0$, and to 2c for $i = 1$.

From the above description it should be easy to construct LREC[$\{E\}$]-formulae $\varphi_E(\bar{u}, \bar{u}')$ and $\varphi_C(\bar{u}, p)$, where $\bar{u} = (q_t, x, y, \hat{x}, \hat{y}, q_k)$ and $\bar{u}' = (q_t', x', y', \hat{x}', \hat{y}', q_k')$, such that $\varphi_E[T, \alpha; \bar{u}, \bar{u}'] = E$, and $\{\langle n \rangle \mid n \in \varphi_C[T, \alpha[\bar{a}/\bar{u}]; p]\} = C(\bar{a})$ for each $\bar{a} \in V$.

Let $\varphi_{\cong}(x, y) := \exists \bar{r}\, [\mathsf{lrec}_{\bar{u},\bar{u}',p}\ \varphi_E,\ \varphi_C]((0, x, y, x, y, 0), \bar{r})$, where $\bar{r}$ is a 5-tuple of number variables.[4] Let X be the relation defined by $\varphi_{\cong}$ in (T, α). By induction on size(v) it is easy to see that $(\bar{a}_{v,w}, \ell) \in X$ implies $T_v \cong T_w$. It remains to prove completeness:

▶ **Lemma 4.1.** *If $T_v \cong T_w$, then for all $\ell \geq \mathrm{size}(v)^5$ we have $(\bar{a}_{v,w}, \ell) \in X$.*

Proof. The proof is by induction on size(v). Suppose that size$(v) = 1$ and $T_v \cong T_w$. Then size$(w) = 1$ which implies that (v, w) is not easy. Furthermore, as v has no children in T, we know that $\bar{a}_{v,w}$ has no children in G and $C(\bar{a}_{v,w}) = \{0\}$. Hence, $(\bar{a}_{v,w}, \ell) \in X$ for all $\ell \geq 1 = \mathrm{size}(v)^5$.

Now suppose that size$(v) = s + 1$ for some $s \geq 1$, and $T_v \cong T_w$. First note that (v, w) is not easy. Let $\ell \geq (s + 1)^5$. We show that $(\bar{a}_{v,w,\hat{v}}, \ell - 1) \in X$ for all children $\hat{v}$ of v, which implies $(\bar{a}_{v,w}, \ell) \in X$. Let $\hat{v}$ be a child of v in T. Since $T_v \cong T_w$, there is a child $\hat{w}$ of w of size $s' := \mathrm{size}(\hat{v})$ and a number $k \in [\#_{s'}(v)]$ such that

- $T_{\hat{v}} \cong T_{\hat{w}}$,
- there are exactly k children $\mathring{w}$ of w of size s' such that $T_{\hat{v}} \cong T_{\mathring{w}}$, and
- there are exactly k children $\mathring{v}$ of v of size s' such that $T_{\mathring{v}} \cong T_{\hat{w}}$.

Pick such $\hat{w}$ and k.

Let us deal with the case $\#_{s'}(v) = 1$ first. In this case, $\bar{a}_{\hat{v},\hat{w}}$ is the only child of $\bar{a}_{v,w,\hat{v},\hat{w},k}$; moreover, $\bar{a}_{v,w,\hat{v},\hat{w},k}$ and $\bar{a}_{\hat{v},\hat{w}}$ have exactly one incoming edge each. Since $T_{\hat{v}} \cong T_{\hat{w}}$ and $\ell - 3 \geq (s')^5$, the induction hypothesis implies $(\bar{a}_{\hat{v},\hat{w}}, \ell - 3) \in X$. Consequently $(\bar{a}_{v,w,\hat{v}}, \ell - 1) \in X$.

In the following we assume $\#_{s'}(v) \geq 2$. Let $d := 3 \cdot \#_{s'}(v)^2$. Note that all vertices in Figure 1 except the type 0-vertices have exactly one incoming edge, and that the in-degree d' of a type 0-vertex $\bar{a}_{v',w'}$, where v', w' are children of v and w, respectively, of size s' is at most d, because it has incoming edges from

- vertices $\bar{a}_{v,w,v',w',k}$, where v and w are the (unique) parents of v' and w', respectively, and $k \in [\#_{s'}(v)]$;

[4] We use 0 as a constant, but clearly we can modify $\varphi_{\cong}$ to a formula that does not use the constant 0.

- vertices $\bar{a}^0_{v,w,v',\hat{w},k}$, where v, w, k are as above and $\hat{w}$ is a child of w of size s'; and
- vertices $\bar{a}^1_{v,w,\hat{v},w',k}$, where v, w, k are as above and $\hat{v}$ is a child of v of size s'.

Let $\ell' := \lfloor(\ell - 4)/d\rfloor$. Then

$$\ell' \ \geq \ \frac{\ell - d - 3}{d} \ \geq \ \frac{s^5}{d} + \frac{s^4}{d} - 2 \ \geq \ \frac{\#_{s'}(v)^5 \cdot (s')^5}{3 \cdot \#_{s'}(v)^2} + \frac{\#_{s'}(v)^4}{3 \cdot \#_{s'}(v)^2} - 2 \ \geq \ 2(s')^5 - 1 \ \geq \ (s')^5,$$

where for the second inequality we use $(s+1)^5 \geq s^5 + s^4$, for the third one we use $\#_{s'}(v) \cdot s' \leq s$, and for the fourth one we use $\#_{s'}(v) \geq 2$. Hence, by the induction hypothesis we have:

- $(\bar{a}_{\hat{v},\hat{w}}, \lfloor(\ell - 3)/d'\rfloor) \in X$ (note that $\lfloor(\ell - 3)/d'\rfloor \geq \ell'$).
- There are exactly k children $\mathring{w}$ of w of size s' with $(\bar{a}_{\hat{v},\mathring{w}}, \lfloor(\ell - 4)/d'\rfloor) \in X$ (note that $\lfloor(\ell - 4)/d'\rfloor \geq \ell'$), which implies $(\bar{a}^0_{v,w,\hat{v},\hat{w},k}, \ell - 3) \in X$.
- There are exactly k children $\mathring{v}$ of v of size s' with $(\bar{a}_{\mathring{v},\hat{w}}, \lfloor(\ell - 4)/d'\rfloor) \in X$, which implies that $(\bar{a}^1_{v,w,\hat{v},\hat{w},k}, \ell - 3) \in X$.

It follows immediately that $(\bar{a}_{v,w,\hat{v},\hat{w},k}, \ell - 2) \in X$, and therefore $(\bar{a}_{v,w,\hat{v}}, \ell - 1) \in X$. ◄

Finally, let $(v, w) \in V(T)^2$. Then we have $\text{size}(v)^5 \leq |V(T)|^5 \leq |N(T)|^{|\bar{r}|} - 1$, and therefore $T \models \varphi_{\cong}[v, w]$ iff $(\bar{a}_{v,w}, |N(T)|^{|\bar{r}|} - 1) \in X$ iff $T_v \cong T_w$.

4.2 Defining an Order on Directed Trees

Lindell's tree canonisation algorithm is based on a logspace-computable linear order on isomorphism classes of directed trees. We show that a slightly refined version of this order is LREC-definable.

Let T be a directed tree. For each $v \in V(T)$ let $\pi(v) := \big(\text{size}(v), \#_1(v), \ldots, \#_{\text{size}(v)-1}(v)\big)$ be the *profile* of v.[5] Let $\preceq$ be the total preorder on $V(T)$,[6] where $v \prec w$ whenever

1. $\pi(v) < \pi(w)$ lexicographically, or
2. $\pi(v) = \pi(w)$ and the following is true: Let $v_1, \ldots, v_k$ and $w_1, \ldots, w_k$ be the children of v and w, respectively, ordered such that $v_1 \preceq \cdots \preceq v_k$ and $w_1 \preceq \cdots \preceq w_k$. Then there is an $i \in [k]$ with $v_i \prec w_i$, and for all $j < i$ we have $v_j \preceq w_j$ and $w_j \preceq v_j$.

Note that $v \preceq w$ and $w \preceq v$ imply $T_v \cong T_w$. We show that $\preceq$ is LREC-definable.

To simplify the presentation, we again fix a directed tree T and an assignment α, and we assume that $|V(T)| \geq 4$.

We apply the lrec-operator to the following graph $G = (V, E)$ with labels $C(v) \subseteq \mathbb{N}$ for each $v \in V$. Let $V := N(T) \times V(T)^4 \times N(T)$. For each $(v, w) \in V(T)^2$, the vertex $\bar{a}_{v,w} = (0, v, w, v, w, 0)$ represents "$v \prec w$". If $\pi(v) < \pi(w)$, then $\bar{a}_{v,w}$ has no outgoing edges and $C(\bar{a}_{v,w}) = \{0\}$. If $\pi(v) > \pi(w)$, then $\bar{a}_{v,w}$ has no outgoing edges and $C(\bar{a}_{v,w}) = \emptyset$. Note that the relation "$\pi(v) \leq \pi(w)$" is LREC-definable.

Suppose that $\pi(v) = \pi(w)$. For all $t, u \in V(T)$ let $\theta_u(t)$ be the number of children u' of u with $T_{u'} \cong T_t$. Call a child $\hat{v}$ of v *good* if $\theta_v(\hat{v}) > \theta_w(\hat{v})$ and for all children v' of v with $\text{size}(v') < \text{size}(\hat{v})$ we have $\theta_v(v') = \theta_w(v')$. Then it is not hard to see that $v \prec w$ precisely if there is a good child $\hat{v}$ of v, a child $\hat{w}$ of w of size $s := \text{size}(\hat{v})$ and a $k \in [\#_s(v)]$ such that $\hat{v} \prec \hat{w}$, there are exactly k children $\mathring{w}$ of w of size s with $\mathring{w} \prec \hat{v}$, there are exactly k children $\mathring{v}$ of v of size s with $\mathring{v} \prec \hat{w}$ and $T_{\mathring{v}} \ncong T_{\hat{v}}$, and for all k children w' of w of size s with $w' \prec \hat{v}$ we have $\theta_v(w') = \theta_w(w')$. The "decision tree" in Figure 2 checks precisely these conditions.

Using the formula $\varphi_{\cong}$ from the previous section it is now straightforward to construct LREC$[\{E\}]$-formulae $\varphi_E(\bar{u}, \bar{u}')$ and $\varphi_C(\bar{u}, p)$ that define the edge relation E of G and the

[5] Lindell's order can be obtained by replacing $\pi(v)$ with $\pi'(v) := \big(\text{size}(v), \#\text{children of } v\big)$.

[6] That is, $\preceq$ is a preorder on $V(T)$ such that for all $v, w \in V(T)$ we have $v \preceq w$ or $w \preceq v$.

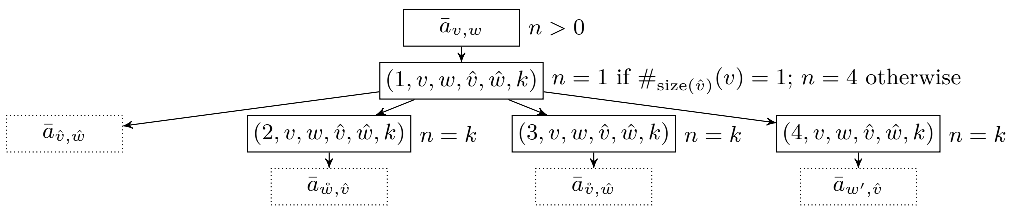

$\bar{a}_{v,w}$
$n > 0$
$(1, v, w, \hat{v}, \hat{w}, k)$
$n = 1$ if $\#_{\text{size}(\hat{v})}(v) = 1$; $n = 4$ otherwise
$\bar{a}_{\hat{v},\hat{w}}$
$(2, v, w, \hat{v}, \hat{w}, k)$
$n = k$
$(3, v, w, \hat{v}, \hat{w}, k)$
$n = k$
$(4, v, w, \hat{v}, \hat{w}, k)$
$n = k$
$\bar{a}_{\mathring{w},\hat{v}}$
$\bar{a}_{\mathring{v},\hat{w}}$
$\bar{a}_{w',\hat{v}}$

Figure 2 Gadget for deciding $v \prec w$ when $\pi(v) = \pi(w)$. Here, $\hat{v}$ ranges over good children of v; $\mathring{v}$ ranges over children of v of size $s := \text{size}(v)$ and $T_{\mathring{v}} \not\cong T_{\hat{v}}$; $\hat{w}, \mathring{w}$ range over children of w of size s; w' ranges over children of w of size s with $\theta_v(w') = \theta_w(w')$; and $k \in [\#_s(v)]$. The edges from $(2, v, w, \hat{v}, \hat{w}, k)$ to $(t, \dots)$ for $t \in \{2, 3, 4\}$ exist only if $\#_s(v) > 1$. Labels indicate which integers n belong to the set $C(\bar{a})$ labelling each vertex $\bar{a}$.

sets $C(\bar{a})$ for each $\bar{a} \in V$, where $\bar{u}$ and $\bar{u}'$ are as in the definition of $\varphi_{\cong}$. Let $\varphi_{\prec}(x, y) := \exists \bar{r}\,[\mathsf{lrec}_{\bar{u},\bar{u}',p}\ \varphi_E,\ \varphi_C]((0, x, y, x, y, 0), \bar{r})$, where $\bar{r}$ is a 5-tuple of number variables. Let X be the relation defined by $\varphi_{\prec}$ in (T, α). It is then possible to show by induction on size(v) that $(\bar{a}_{v,w}, \ell) \in X$ implies $v \prec w$ and that $v \prec w$ implies $(\bar{a}_{v,w}, \ell) \in X$ for all $\ell \geq \text{size}(v)^5$. Hence, $T \models \varphi_{\prec}[v, w]$ iff $(\bar{a}_{v,w}, |N(T)|^{|\bar{r}|} - 1) \in X$ iff $v \prec w$.

4.3 Canonising Directed Trees

We now construct an LREC-formula $\gamma(p, q)$ such that for every directed tree T we have $T \cong ([|V(T)|], \gamma[T; p, q])$. Since DTC captures LOGSPACE on ordered structures [13] and a linear order is available on the number sort, we immediately obtain:

▶ **Theorem 4.2.** LREC *captures* LOGSPACE *on the class of directed trees.*

Since directed tree isomorphism is in LOGSPACE by Lindell's tree canonisation algorithm, but not TC+C-definable [5], we obtain:

▶ **Corollary 4.3.** LREC $\not\leq$ TC+C *on the class of all directed trees.*

We use l-recursion to define a set $X \subseteq V(T) \times N(T)^2$ (for simplicity, we omit the "resources" in the description) such that for every $v \in V(T)$ the set $X_v := \{(m, n) \in N(T)^2 \mid (v, m, n) \in X\}$ is the edge relation of an isomorphic copy $([|V(T_v)|], X_v)$ of T_v. Each vertex of T is numbered by its position in the preorder traversal sequence, e.g., the root is numbered 1, its first child v_1 is numbered 2, its second child v_2 is numbered $2 + \text{size}(v_1)$, and so on.

To apply the lrec-operator, we define a graph $G = (V, E)$ with labels $C(v) \subseteq \mathbb{N}$ for each $v \in V$ as follows. Let $V := V(T) \times N(T)^2$, where $(v, m, n) \in V$ stands for "$(m, n) \in X_v$?". If v is a leaf, then X_v should be empty, so for all $m, n \in N(T)$ we let (v, m, n) have no outgoing edges and define $C((v, m, n)) := \emptyset$. Suppose that v is not a leaf and w is a child of v. Let S_w be the set of all children w' of v with $w' \prec w$, and let e_w be the number of children w' of v with $T_w \cong T_{w'}$. For each $i \in [0, e_w - 1]$, the set X_v will contain an edge from 1 to $p_{w,i} := 2 + \sum_{w' \in S_w} \text{size}(w') + i \cdot \text{size}(w)$, and the edges in $\{(p_{w,i} - 1 + m, p_{w,i} - 1 + n) \mid (m, n) \in X_w\}$. Hence we let $(v, 1, p_{w,i})$ have no outgoing edges and define $C((v, 1, p_{w,i})) := \{0\}$. Furthermore, for all $m, n \in N(T)$ and all $i < e_w$, we let $\bar{a} := (v, p_{w,i} - 1 + m, p_{w,i} - 1 + n)$ have an edge to (w, m, n) and define $C(\bar{a}) := \{1\}$.

It is now easy to construct LREC-formulae $\varphi_E(x_1, p_1, p_1', x_2, p_2, p_2')$ and $\varphi_C(x_1, p_1, p_1', q)$ that define the graph G and the labels $C(\cdot)$. Let

$$\gamma(p_1, p_2) := \exists x \exists r \big(\text{"}x \text{ is the root"} \wedge [\mathsf{lrec}_{(x_1,p_1,p_1'),(x_2,p_2,p_2'),q}\ \varphi_E,\ \varphi_C]((x, p_1, p_2), r)\big).$$

Noting that the in-degree of each vertex (v, m, n) is at most e_v, it is straightforward to show that γ defines an isomorphic copy of a directed tree.

5 Inexpressibility of Reachability in Undirected Graphs

In LREC it is not possible to define reachability in undirected graphs:

▶ **Theorem 5.1.** *There is no* $\mathsf{LREC}[\{E\}]$*-formula* $\varphi(x, y)$ *such that for all undirected graphs* G *and all* $v, w \in V(G)$, $G \models \varphi[v, w]$ *iff there is a path from* v *to* w *in* G*.*

As an immediate corollary we obtain:

▶ **Corollary 5.2.** $\mathsf{STC} \not\leq \mathsf{LREC}$

For the proof of Theorem 5.1, we consider the following undirected graphs G_n, for $n \geq 1$. Each graph G_n consists of $2 \cdot n^2$ vertices partitioned into *layers* $V_1^1, \ldots, V_n^1, V_1^2, \ldots, V_n^2$ with $|V_i^j| = n$, where every two vertices in consecutive layers V_i^j and V_{i+1}^j are connected by an edge, i.e., $E(G_n) = \{(v, w) \in V_i^j \times V_{i+1}^j \mid i \in [n-1],\, j \in [2]\}$. Vertices of G_n that belong to the same layer are called *siblings*. We show that reachability is not LREC-definable on the class $\mathcal{C}$ of all graphs that are isomorphic to G_n for some $n \geq 1$.

More precisely, we show that on $\mathcal{C}$ every $\mathsf{LREC}[\{E\}]$-formula φ is equivalent to a formula in the *infinitary counting logic* $\mathcal{L}^*_{\infty\omega}(\mathbf{C})$ (see [19] or [20, Section 8.2]). Theorem 5.1 then immediately follows from the fact that every $\mathcal{L}^*_{\infty\omega}(\mathbf{C})$-formula without free number variables is Gaifman-local [19].

To construct an equivalent $\mathcal{L}^*_{\infty\omega}(\mathbf{C})$-formula we proceed by induction on the structure of the formula φ. The only interesting case is that of an $\mathsf{LREC}[\{E\}]$-formula of the form $\varphi = [\mathsf{lrec}_{\bar{u}_1, \bar{u}_2, \bar{p}}\ \varphi_E,\ \varphi_C](\bar{w}, \bar{r})$. Let $\tilde{v}_E$ be an enumeration of all variables in free(φ_E) that are not listed in $\bar{u}_1\bar{u}_2$ and let $\tilde{v}_C$ be an enumeration of all variables in free(φ_C) that are not listed in $\bar{u}_1\bar{p}$. Let $n > |\bar{u}_1| + |\tilde{v}_E| + 2$, and consider an assignment α in G_n. Further, let $V := G_n^{\bar{u}_1}$ and $E := \varphi_E[G_n, \alpha; \bar{u}_1, \bar{u}_2]$. For every $\bar{a} \in V$ and $\ell \in \mathbb{N}$, let $\mathcal{P}_{n,\ell}(\bar{a})$ be the set of all sequences $((\bar{a}_0, \ell_0), \ldots, (\bar{a}_m, \ell_m)) \in (V \times [0, \ell])^{m+1}$, where $m \in \mathbb{N}$, $(\bar{a}_0, \ell_0) = (\bar{a}, \ell)$, $(\bar{a}_0, \ldots, \bar{a}_m)$ is a path in (V, E), and $\ell_i = \lfloor (\ell_{i-1} - 1)/|E\bar{a}_i| \rfloor$ for each $i \in [m]$. The key property which enables us to construct a $\mathcal{L}^*_{\infty\omega}(\mathbf{C})$-formula equivalent to φ on $\mathcal{C}$ is:

▶ **Lemma 5.3.** *Let* $\bar{a} \in V$, $\ell \in \mathbb{N}$, *and* $((\bar{a}_0, \ell_0), \ldots, (\bar{a}_m, \ell_m)) \in \mathcal{P}_{n,\ell}(\bar{a})$*. Let* $\mathcal{I}$ *be the set of all* $i \in [m]$ *such that* $(\tilde{a}_{i-1} \cup \alpha(\tilde{v}_E)) \cap V(G_n) \neq (\tilde{a}_i \cup \alpha(\tilde{v}_E)) \cap V(G_n)$*. Then* $|\mathcal{I}|$ *is bounded by a constant that depends only on* φ*.*

The main insight for proving the lemma is that for every two siblings $b, b' \in V(G_n)$ there is an automorphism of G_n swapping b and b' and fixing all other vertices pointwise. Therefore, if $((\bar{a}_0, \ell_0), \ldots, (\bar{a}_m, \ell_m)) \in \mathcal{P}_{n,\ell}(\bar{a})$ and $i \in [m]$ is such that $\tilde{a}_{i-1} \cap V(G_n) \not\subseteq (\tilde{a}_i \cup \alpha(\tilde{v}_E)) \cap V(G_n)$, then for any $b \in \tilde{a}_{i-1} \cap V(G_n)$ with $b \notin \tilde{a}_i \cup \alpha(\tilde{v}_E)$, there is a linear number of siblings of b that do not occur in $\tilde{a}_i \cup \alpha(\tilde{v}_E) \cup \{b\}$, each leading to an incoming edge at $\bar{a}_i$. It is not hard to bound the number of all other $i \in \mathcal{I}$ by a constant.

6 An Extension of LREC

The previous section's Theorem 5.1 reveals that LREC is not closed under (first-order) logical reductions.[7] To remedy this weakness, we introduce the following extension $\mathsf{LREC}_=$ of LREC.

[7] There is an FO-reduction that takes the graphs G_n, $n \geq 3$, considered in Section 5 to disjoint unions $\hat{G}_n$ of two undirected paths on n vertices each by identifying siblings. It is not hard to see that reachability

The idea is to admit a third formula $\varphi_=$ in the lrec-operator that generates an equivalence relation on the vertices of the graph defined by φ_E.

Let τ be a vocabulary. The set of all $\mathsf{LREC}_=[\tau]$-formulae is obtained from $\mathsf{LREC}[\tau]$ by replacing the rule for the lrec-operator from Section 3 as follows: If $\bar{u}, \bar{v}, \bar{w}$ are compatible tuples of variables, $\bar{p}, \bar{r}$ are non-empty tuples of number variables, and $\varphi_=$, φ_E and φ_C are $\mathsf{LREC}_=$-formulae, then the following is an $\mathsf{LREC}_=[\tau]$-formula:

$$\varphi := [\mathsf{lrec}_{\bar{u},\bar{v},\bar{p}}\ \varphi_=,\ \varphi_E,\ \varphi_C](\bar{w}, \bar{r}). \tag{3}$$

We let $\text{free}(\varphi) := \big(\text{free}(\varphi_=) \setminus (\tilde{u} \cup \tilde{v})\big) \cup \big(\text{free}(\varphi_E) \setminus (\tilde{u} \cup \tilde{v})\big) \cup \big(\text{free}(\varphi_C) \setminus (\tilde{u} \cup \tilde{p})\big) \cup \tilde{w} \cup \tilde{r}$.

To define the semantics of $\mathsf{LREC}_=[\tau]$-formulae φ of the form (3) let A be a τ-structure and α an assignment in A. Let $V_0 := A^{\bar{u}}$ and $E_0 := \varphi_E[A, \alpha; \bar{u}, \bar{v}]$. We define $\sim$ to be the reflexive, symmetric, transitive closure of the binary relation $\varphi_=[A, \alpha; \bar{u}, \bar{v}]$ over V_0. For every $\bar{a} \in V_0$ let $[\bar{a}]$ be the equivalence class of $\bar{a}$ with respect to $\sim$. Now consider the graph with vertex set $V := \{[\bar{a}] \mid \bar{a} \in V_0\}$ and edge set $E := \{[\bar{a}][\bar{b}] \in V^2 \mid \text{there are } \bar{a}' \in [\bar{a}],\ \bar{b}' \in [\bar{b}] \text{ with } \bar{a}'\bar{b}' \in E_0\}$. To every $[\bar{a}] \in V$ we assign the set $C([\bar{a}]) := \{\langle \bar{n} \rangle \mid \bar{n} \in \varphi_C[A, \alpha[\bar{a}'/\bar{u}]; \bar{p}], \bar{a}' \in [\bar{a}]\}$ of labels. Then the definition of X can be taken verbatim from Section 3. We let $(A, \alpha) \models \varphi$ if and only if $\big([\alpha(\bar{w})], \langle \alpha(\bar{r}) \rangle\big) \in X$.

As for LREC, the data complexity of $\mathsf{LREC}_=$ is in $\mathsf{LOGSPACE}$ and $\mathsf{LREC}_= \leq \mathsf{FP{+}C}$. Furthermore, $\mathsf{LREC}_=$ is closed under logical reductions.

The following example shows that undirected graph reachability is definable in $\mathsf{LREC}_=$. This does not involve an implementation of Reingold's algorithm in our logic, but just uses the observation that the computation of the equivalence relation $\sim$ boils down to the computation of undirected reachability.

▶ **Example 6.1** (Undirected reachability)**.** The following LREC-formula defines undirected graph reachability:

$$\varphi(s, t) \ := \ [\mathsf{lrec}_{x,y,p}\ \varphi_=(x, y),\ \varphi_E(x, y),\ \varphi_C(x, p)](s, 1),$$

where $\varphi_=(x, y) := E(x, y)$, $\varphi_E(x, y) := x \neq x$ and $\varphi_C(x, p) := x = t$. Let G be an undirected graph and α an assignment in G. Define V, E, C and the set X as above. Clearly, the set V consists of the connected components of G. Furthermore, the set E is empty since φ_E is unsatisfiable. Therefore, for all $v \in V(G)$ we have $([v], 1) \in X$ iff $0 \in C([v])$. The latter is true precisely if $\alpha(t) \in [v]$, i.e., if v and $\alpha(t)$ are in the same connected component of G. It follows that for all $v, w \in V(G)$ we have $G \models \varphi[v, w]$ if and only if v and w are in the same connected component of G, that is, if there is a path from v to w in G. ◀

It follows immediately from the previous example that $\mathsf{STC{+}C} \leq \mathsf{LREC}_=$. Actually, the containment is strict, because $\mathsf{LREC} \not\leq \mathsf{STC{+}C}$. Since trees can be made directed in $\mathsf{STC{+}C}$, the results from Section 4 imply that $\mathsf{LREC}_=$ captures $\mathsf{LOGSPACE}$ on the class of all trees.

7 Capturing Logspace on Interval Graphs

We now prove that $\mathsf{LREC}_=$ captures $\mathsf{LOGSPACE}$ on the class of all interval graphs. The result is shown by describing an $\mathsf{LREC}_=$-definable canonisation procedure for interval graphs, which relies on a specific decomposition of graphs known as modular decomposition (first introduced

on the class of all graphs $\hat{G}_n$ is LREC-definable. Hence, if LREC was closed under FO-reductions, then reachability on the class of all graphs G_n would be LREC-definable, contradicting Theorem 5.1.

by Gallai [7]). It combines algorithmic techniques from [16] with the logical definability framework in [17]. The results in [17] are stated for fixed-point logic with counting only, but many of the results that are of interest for our construction hold in fact for STC+C. Parts of Sections 7.1 and 7.2 can be found in more detail in [18].

7.1 Definition of Interval Graphs and Basic Properties

▶ **Definition 7.1** (Interval graph). Let $\mathcal{I}$ be a finite collection of closed intervals $I_i = [a_i, b_i] \subset \mathbb{N}$. The graph $G_\mathcal{I} = (V, E)$ has vertex set $V = \mathcal{I}$ and edge relation $I_iI_j \in E :\Leftrightarrow I_i \cap I_j \neq \emptyset$. $\mathcal{I}$ is called an *interval representation* of a graph G if $G \cong G_\mathcal{I}$, and a graph G is an *interval graph* if there is a collection of closed intervals $\mathcal{I}$ which is an interval representation of G.

A *clique* of a graph $G = (V, E)$ is a subset $C \subseteq V$ of the vertex set, such that the subgraph induced by C is complete. A maximal clique, or *max clique*, is a clique that is not properly contained in another clique. It is known [8, 22] that a graph G is an interval graph if and only if its max cliques can be brought into a linear order, so that each vertex of G is contained in consecutive max cliques. Let us denote the set of a graph's max cliques by $\mathcal{M}$. For canonisation it is essential to linearly order the max cliques of G.

Let $\mathrm{N}^\mathrm{c}(v)$ denote the closed neighbourhood of a vertex v, i.e. the set containing v and all vertices adjacent to v. The first lemma shows that the maximal cliques are FO-definable, as is the equivalence relation on V^2 of vertex pairs defining the same max clique.

▶ **Lemma 7.2** ([17], Lemma 4.1). *Let G be an interval graph and let M be a max clique of G. Then there are vertices $u, v \in M$, not necessarily distinct, such that $M = \mathrm{N}^\mathrm{c}(u) \cap \mathrm{N}^\mathrm{c}(v)$.* ◀

The *span* of a vertex $v \in V$, denoted $\mathrm{span}(v)$, is the number of max cliques of G that v is contained in. Since equivalence classes can be counted in STC+C (Lemma 2.7. in [17]), the span of a vertex is STC+C-definable on the class of interval graphs.

7.2 Modular Decomposition Tree

A set W of vertices in a graph $G = (V, E)$ is a *module* if for all vertices $v \in V \setminus W$ either $\{v\} \times W \subseteq E$ or $(\{v\} \times W) \cap E = \emptyset$. The vertex set V and all vertex sets of size 1 are *trivial modules* by this definition. A module W is *proper* if $W \subset V$.

Let us call a vertex of G which is adjacent to all other vertices an *apex* of G. If G is a connected interval graph without an apex, then the complement graph of G is connected as well, and by [7] the set of maximal proper modules of G is a partition of G's vertex set. Thus, the set of proper modules $\mathcal{W}_G = \{W_1, \dots, W_k\}$ of G which replaces every maximal proper module that is a subset of just one maximal clique by modules of size 1 for all contained vertices is also partition of G's vertex set. Each pair of modules W_i, W_j, $i \neq j$, is either completely connected or completely disconnected. Let $\sim_G$ be the equivalence relation corresponding to the partition $\mathcal{W}_G$, and let $L_G = G \big/ {\sim_G} := (V \big/ {\sim_G}, E_L)$, where $[u][v] \in E_L \Leftrightarrow \exists x \in [u], y \in [v]$ such that $xy \in E$. If G contains an apex, we let $\sim_G$ be the equivalence relation for which $x \sim_G y$ if and only if $x = y$ or $x, y \in V \setminus A$, where $A \subseteq V$ is the set of apices, and define L_G equivalently.

It is easy to see that L_G is an interval graph, that if A is a max clique of G, then $A_{L_G} := \{[v] \mid v \in A\}$ is a max clique of L_G, and that all max cliques of L_G are of this form. The following lemma gives further information about L_G.

▶ **Lemma 7.3.** *Let m be the number of max cliques of L_G.*

1. *There are* STC+C*-formulae $\varphi_{\sim_G}$, φ_{L_G} such that for all connected interval graphs, $\varphi_{\sim_G}$ defines the equivalence relation $\sim_G$, and φ_{L_G} the graph L_G.*
2. *If $m > 1$, then there exist exactly two linear orderings of L_G's max cliques, each the reverse of the other. There is an* STC+C*-formula that given a max clique A of G defines the one with A_{L_G} appearing within the first $\lfloor \frac{m}{2} \rfloor$ max cliques of L_G.*
3. *There is an* STC+C*-formula that for all connected interval graphs G canonises L_G.*

According to the preceding lemma we can define an isomorphic copy $\mathcal{K}(L_G)$ of L_G on the number sort. What is left is to deal with the contents of the non-singular modules $W_{i_1}, \dots, W_{i_l}$ of $\mathcal{W}_G$. If we continue decomposing the graphs $G[W_{i_1}], \dots, G[W_{i_l}]$ inductively until we arrive at singular sets everywhere, we obtain a *modular decomposition* of G.

Let $P' = \{(M,n) \mid M \in \mathcal{M}, n \in [|V|]\}$. For each $(M,n) \in P'$ define $V_{M,n}$ as the set of vertices of the connected component of $G[\{v \in V \mid \mathrm{span}(v) \leq n\}]$ which intersects M (if non-empty).

Let W be a non-singular maximal proper module. We define $\mathcal{C}$ to be the set of max cliques C such that $C \cap W \neq \emptyset$. It is immediate from the definition of a module W that $W = \bigcup_{C \in \mathcal{C}} V_{C,|\mathcal{C}|}$. Thus, for any $C \in \mathcal{C}$, the set $V_{C,|\mathcal{C}|}$ defines a component of W.

Let P be the set of those $(M,n) \in P'$ for which n is maximal among all n' with the property that $V_{M,n'} = V_{M,n}$ and which satisfy that $V_{M,n}$ is the connected component of a module occurring in the modular decomposition of G. Then P contains exactly all the components of modules occurring in the modular decomposition. There exists an STC+C-formula deciding whether (M,n) is in P.

Now we want to construct a coloured modular decomposition tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$. An illustration of the tree can be found in the full version. We will later need STC+C-definability of this coloured tree. Thus, notice that the tree's vertices are equivalence classes, which are STC+C definable. Also the edge relation and the colours are STC+C-definable (Lemma 7.3).

Let $V_\mathcal{T}$ be the union of the following sets:

- the set $\mathcal{V}$ of *component vertices* $v_{V_{M,n}}$, one for each set $V_{M,n}$ with $(M,n) \in P$,
- the set $\mathcal{A}$ of *arrangement vertices* $a_{\prec_Q, V_{M,n}}$ where $\prec_Q$ is the distinguished order on $L_{V_{M,n}}$'s max cliques if $\mathcal{K}(L_{V_{M,n}})$ is not order isomorphic under its two linear orderings, and if $\mathcal{K}(L_{V_{M,n}})$ is order isomorphic under its two linear orderings, then max clique Q identifies an order $\prec_Q$, namely, the order where $Q_{L_{V_{M,n}}}$ occurs first. (Q defines both orders if $Q_{L_{V_{M,n}}}$ is in the middle.)
- the set $\mathcal{S}$ of *module vertices* $s_{W_A, V_{M,n}}$ for which W_A is the module of $V_{M,n}$ that contains vertices of max clique A, and
- $\{s_V\}$, where s_V is a special vertex acting as the root of $\mathcal{T}$.

We colour the vertices in $\mathcal{V}$ by assigning to each $v_{V_{M,n}} \in \mathcal{V}$ the ordered graph $\mathcal{K}(L_{V_{M,n}})$. The vertices in $\mathcal{A}$ remain uncoloured and may therefore be exchanged by an automorphism of $\mathcal{T}$ whenever their subtrees are isomorphic. Each $s_{W_A, V_{M,n}} \in \mathcal{S}$ is coloured with the multiset of integers corresponding to the positions that the max clique $A_{L_{V_{M,n}}}$ takes in the orders of $L_{V_{M,n}}$.

The edge relation $E_\mathcal{T}$ of $\mathcal{T}$ is now defined in a straight-forward manner, with all edges directed away from the root s_V.

- s_V is connected to all $v_{V_{M,n}} \in \mathcal{V}$ with $n = |V|$.
- Each $v_{V_{M,n}} \in \mathcal{V}$ is connected to all vertices in $\mathcal{A}$ of the form $a_{\prec_Q, V_{M,n}}$ with $Q \cap V_{M,n} \neq \emptyset$.
- Each $a_{\prec_Q, V_{M,n}} \in \mathcal{A}$ is connected to all those $s_{W_A, V_{M,n}} \in \mathcal{S}$ so that $\prec_Q$ belongs to the set of orders of $L_{V_{M,n}}$ under which module $W_A \in L_{V_{M,n}}$ attains its minimal position,

that is, for every max clique Q that intersects with a non-singular module of $V_{M,n}$ vertex $a_{\prec_Q, V_{M,n}} \in \mathcal{A}$ is connected to $s_{W_Q, V_{M,n}} \in \mathcal{S}$.

- Every $s_{W_A, V_{M,n}} \in \mathcal{S}$ is connected to those $v_{V_{M',n'}} \in \mathcal{V}$ for which $V_{M',n'}$ is a connected component of the module W_A, that is, $s_{W_A, V_{M,n}} \in \mathcal{S}$ is connected to $v_{V_{A,n'}} \in \mathcal{V}$ with $n' = \max\{m < n \mid (V_{A,m}) \in P\}$.

The point of the arrangement vertices $\mathcal{A}$ is to ensure that the order of submodules is properly accounted for. If our modular tree did not have such a safeguard, exchanging modules in symmetric positions might give rise to a non-isomorphic graph, but it would not change the tree, so $\mathcal{T}$ would be useless for the task of distinguishing between these two graphs.

Lemma 7.4 below shows that our modular trees are a complete invariant of interval graphs, so modular trees can be used to tell whether two interval graphs are isomorphic.

▶ **Lemma 7.4** ([16], see full version for further remarks). *Let G and H be interval graphs. If their modular trees are isomorphic, then so are G and H.* ◀

7.3 Canonisation

We can now make use of the STC+C-definable modular decomposition tree:

▶ **Lemma 7.5.** *Let $\theta_V(\bar{u})$, $\theta_{\approx}(\bar{u}, \bar{v})$, $\theta_E(\bar{u}, \bar{v})$ and $\theta_L(\bar{u}, \bar{q})$ be STC+C-formulae with $\bar{u}, \bar{v}$ compatible tuples and $\bar{q}$ a tuple of number variables, such that for all interval graphs G and assignments α, $\theta_{\approx}[G, \alpha; \bar{u}, \bar{v}]$ generates an equivalence relation $\approx$, $\theta_V[A, \alpha; \bar{u}] \,/\!\approx$ the vertices, $\theta_E[A, \alpha; \bar{u}, \bar{v}] \,/\!\approx$ the edge relation and $\theta_L[A, \alpha; \bar{u}, \bar{q}] \,/\!\approx$ the colours of the modular decomposition tree $\mathcal{T}_G$. Then there is an LREC$_=$-formula $\psi_{\preceq'}(\bar{u}, \bar{v})$ such that $\psi_{\preceq'}[A, \alpha; \bar{u}, \bar{v}] \,/\!\approx$ defines for all G a total preorder on the vertices of $\mathcal{T}_G$, which is more precisely, a linear order on the isomorphism classes of the (coloured) subtrees of $\mathcal{T}_G$ identified by its root vertices.*

Finally, we can use the modular decomposition tree and the total preorder on its vertices for canonisation. We use l-recursion on the modular decomposition tree, and as we have done for canonising trees we build the canon from the leaves to the root of the tree. Recursively, we construct the canon by first building the disjoint union of the canons of the components of submodules, then use the arrangement vertices to insert all submodules at the correct side and build the canon of the correspondent component of a module.

▶ Remark. It is possible to show that there is no LREC+TC[$\{E\}$]-sentence φ such that for all connected interval graphs G_1, G_2 we have $G_1 \uplus G_2 \models \varphi$ if and only if $G_1 \cong G_2$. The proof is based on similar ideas as the proof of Theorem 5.1.

8 Conclusion

We introduce the new logics LREC and LREC$_=$, extending first-order logic with counting by a recursion operator that can be evaluated in logarithmic space. By capturing LOGSPACE on trees and interval graphs, we obtain the first nontrivial descriptive characterisations of LOGSPACE on natural classes of unordered structures. It would be interesting to extend our results to further classes of structures such as the class of planar graphs or classes of graphs of bounded tree width.

The expressive power of LREC$_=$ is not yet well-understood. For example, it is an open question whether directed graph reachability is expressible in LREC$_=$, and even whether LREC$_=$ has the same expressive power as FP+C. (Of course assumptions from complexity theory indicate that the answer to both questions is negative.) It is also an open question whether reachability on undirected trees is expressible in plain LREC.

It is obvious that our capturing results can be transferred to nondeterministic logarithmic space NL by adding a transitive closure operator to the logic. However, it would be much nicer to have a natural “nondeterministic” variant of our limited recursion operator that allows it to express directed graph reachability and thus yields a logic that contains TC. We leave it as an open problem to find such an operator.

References

1 Martin Beaudry and Pierre McKenzie. Circuits, matrices, and nonassociative computation. *J. Comput. Syst. Sci.*, 50(3):441–455, 1995.

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

3 A. Chandra and D. Harel. Structure and complexity of relational queries. *J. Comput. Syst. Sci.*, 25:99–128, 1982.

4 H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1995.

5 K. Etessami and N. Immerman. Tree canonization and transitive closure. *Inf. Comput.*, 157:2–24, 2000.

6 R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings 7*, pages 43–73, 1974.

7 T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Hungar.*, 18(1-2):25–66, 1967.

8 P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

9 E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications.* Springer-Verlag, 2007.

10 M. Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. In *LICS*, 2010.

11 N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

12 N. Immerman. Expressibility as a complexity measure: results and directions. In *Structure in Complexity Theory Conference*, pages 194–202, 1987.

13 N. Immerman. Languages that capture complexity classes. *SIAM JoC*, 16:760–778, 1987.

14 N. Immerman. *Descriptive Complexity.* Springer-Verlag, 1999.

15 N. Immerman and E. Lander. Describing graphs: a first-order approach to graph canonization. In A. Selman, editor, *Complexity theory retrospective.* Springer-Verlag, 1990.

16 J. Köbler, S. Kuhnert, B. Laubner, and O. Verbitsky. Interval graphs: Canonical representation in logspace. In *ICALP (1)*, pages 384–395, 2010.

17 B. Laubner. Capturing polynomial time on interval graphs. In *LICS*, pages 199–208, 2010.

18 B. Laubner. *The Structure of Graphs and New Logics for the Characterization of Polynomial Time.* PhD thesis, Humboldt-Universität zu Berlin, 2011.

19 L. Libkin. Logics with counting and local properties. *ACM ToCL*, 1(1):33–59, 2000.

20 L. Libkin. *Elements of Finite Model Theory.* Springer-Verlag, 2004.

21 S. Lindell. A logspace algorithm for tree canonization. In *STOC*, pages 400–404, 1992.

22 R. H. Möhring. *Graphs and Order*, volume 147 of *NATO ASI Series C, Mathematical and Physical Sciences*, pages 41–102. D. Reidel, 1984.

23 M.Y. Vardi. The complexity of relational query languages. In *STOC*, pages 137–146, 1982.