1

A Landscape of Logics for Finite Unordered Unranked Trees

STEPHAN KEPSER

Abstract

In this paper, we draw a landscape of the expressive power of diverse logics over finite unordered unranked trees. A tree is unordered iff for each node there is no order on its children. A tree is unranked iff for each node the number of its children is independent of its label. We compare here the expressive power of logics from three non-disjoint areas: logics related to automata theory, logics from descriptive complexity theory, and second-order logics. Several of these logics form natural hierarchies of expressive power. We will show several separation results in these hierarchies thus showing that the hierarchies are mostly proper. We also present that the automata logics are incomparable to the logics from descriptive complexity theory.

Keywords TREE LANGUAGES, TREE AUTOMATA, TREE LOGICS

1.1 Introduction

In this paper, we consider finite labelled unordered unranked trees. A tree is called *ordered* iff for each node there is a linear order on the children of this node. A tree is called *unordered* iff for each node there is no order on its children. The two notions are not complementary. But partially ordered trees have so far not attracted any research interest.

A tree is *ranked* iff for each node the number of its children is a function of its label. More generally, a ranking assigns to each label a *finite* set of natural numbers. Each member of the set is a potential number of child nodes. We consider in this paper the unranked case. That means each node may have an arbitrary, but finite, number of children, independent of the label it bears.

Finite unordered unranked trees have many applications in computer science. The one that is probably best known comes from semi-structured

FG-2008.

database theory. Unordered unranked trees provide the so-called *database*-model of XML (Abiteboul et al., 2000). Unordered unranked trees also have applications in computational linguistics. They can be seen as the underlying data structures of dependency treebanks. The dependency structure usually forms a tree. There exists an ordering on the word level. But this order is not relevant for the dependency structure. Thus the trees are unordered. This forms the main motivation for our work. We intend to investigate the expressive power of diverse logics as query languages for dependency treebanks.

In this paper we study a large number of logics to define languages of unordered unranked trees and compare their expressive power. Generally speaking, the logics we consider stem from three non-disjoint areas: logics related to automata theory, logics discussed in descriptive complexity theory, and second-order logics. The basic logic from automata theory is monadic second-order logic. Two extensions of this logic will also be discussed. From the area of descriptive complexity theory we consider

- deterministic transitive closure logic,
- transitive closure logic,
- least or initial fixed-point logic,
- partial fixed-point logic, and
- infinitary logic with finitely many variables.

We also discuss full second-order logic, its restriction to pure existential quantification of second-order variables and its extension by second-order transitive closure.

Several of these logics form natural hierarchies of expressive power. This is true for the automata logics, the logics from descriptive complexity theory, and second-order logics. We will show numerous separation results in these hierarchies thus showing that the hierarchies are mostly proper. We also present that the automata logics are incomparable to the logics from descriptive complexity theory.

This paper is organised as follows. After the definition of finite unordered unranked trees in the preliminaries we briefly recall the definitions of all logics of this paper in Section 1.3. Section 1.4 provides two simple results to start with. Section 1.5 contains the separation of automata logics from fixed-point logics. How to separate the fixed-point logics from second-order logics is shown in Section 1.6. We close the paper with an overview of the results obtained in Section 1.7. For comparison, we added an appendix containing a description of the situation for finite ordered ranked trees.

Due to restrictions of space, most formal definitions and some proofs had to be omitted from this paper. A technical report containing all definitions and proofs is obtainable from the authors.

1.2 Preliminaries

We consider node-labelled finite unordered unranked trees. A tree is a finite digraph with a distinguished node, the root and the property that for every node there is a unique path from the root to this node. We also assume a finite set Λ of node labels.

Formally, a tree is given by a triple (V,E,λ) where V is a finite, non-empty set of vertices or nodes, $E \subseteq V \times V$ is a finite set of edges, and λ is a mapping from V to Λ. Moreover, there is an $r \in V$, the root, such that for each node $v \in V$ there is $n \in \mathbb{N}$ and nodes $v_0, v_1, \ldots, v_n \in V$ with $r = v_0, v_n = v$ and $(v_i, v_{i+1}) \in E$ for all $0 \leq i < n$ (existence of a path from the root to every node). Finally for all $v, v' \in V$, if there are $n, m \in \mathbb{N}$, nodes $v_0, v_1, \ldots, v_n \in V$, $u_0, u_1, \ldots, u_m \in V$ with $v = v_0 = u_0, v_n = u_m = v'$ and $(v_i, v_{i+1}) \in E$ for $0 \leq i < n$ and $(u_j, u_{j+1}) \in E$ for $0 \leq j < m$ then $n = m$ and $v_i = u_i$ for all $0 \leq i \leq n$ (uniqueness of paths).

A tree language is a set of trees.

Similar to ordered trees, unordered trees can also be defined as terms. This way of formalising them is useful in the discussions concering automata related logics. We provide it here as an equivalent alternative to the definition above. Let M be a set. A multi-set is a function $f : M \to \mathbb{N}$ stating for each element of M its multiplicity. For a sequence $m_1, \ldots, m_k \in M$ of not necessarily different elements from M we denote $\{m_1, \ldots, m_k\}$ its multi-set. A multi-set can also be seen as an unordered sequence.

Based on multi-sets, unordered unranked trees for a given signature Λ are defined as follows. Each $L \in \Lambda$ is an unordered unranked tree. If $t_1, \ldots, t_k$ are unordered unranked trees and $L \in \Lambda$ then $L\{t_1, \ldots, t_k\}$ is an unordered unranked tree. The multi-set union is denoted by $\uplus$. $M \subseteq_{mfin} N$ means that M is a finite sub-multiset of N (where N may also be a set).

1.3 The Logics

The basic logic we consider is first-order logic (denoted FO). From the point of view of logic, trees are particular finite first-order structures. With every tree (V,E,λ) we associate a first-order structure $(V,E,(L)_{L\in\Lambda})$ such that $L(v)$ iff $\lambda(v) = L$ for every $v \in V$. Hence we use the following atomic formulae: $E(x,y)$ denotes the directed edge from x (parent) to y (child). And $L(x)$ expresses that node x is labelled with $L \in \Lambda$.

1.3.1 Automata Related Logics

The logics in this section are logics defined to be equivalent to certain types of tree automata. In opposite to the case of ordered ranked trees, differences in the definition of tree automata lead to differences in expressive power. The automata and logics definitions that follow are taken from (Boneva and Talbot, 2005) and (Seidl et al., 2003).

Monadic second-order logic (MSO) is the extension of first-order logic by set variables and quantification over sets.

The logic Counting MSO, defined by Courcelle (1990), denoted CMSO, is an extension of MSO by predicates that allow modulo counting of sets. The syntax of MSO is extended by atomic formulae $Mod^i_j(X)$ where X is a set variable, $i,j \in \mathbb{N}, j < i$. The formula $Mod^i_j(X)$ is true iff X has j elements modulo i.

Seidl, Schwentick, and Muscholl (2003) propose another, yet more powerful, extension of MSO, namely Presburger MSO (denoted PMSO). The name Presburger refers to the fact that for an arbitrary node subsets of child nodes can be restricted by constraints expressed in Presburger arithmetic. An example would be to state that there are twice as many child nodes labelled L_1 than children labelled L_2.

The syntax of PMSO is given by the following grammar (quoted from (Seidl et al., 2003)):

$$\begin{array}{rcl} f & ::= & E(x,x) \mid x \in S \mid x/p \mid f \wedge f \mid \neg f \mid \exists x.f \mid \exists X.f \\ S & ::= & X \mid L \\ p & ::= & t = t \mid t + t = t \mid p \wedge p \mid \neg p \mid \exists y.p \\ t & ::= & [S] \mid y \mid n \end{array}$$

f is a PMSO formula, S is a set, p is a Presburger constraint, and t is a term. $x \in \mathcal{X}_0$ is a first-order variable, $X \in \mathcal{X}_1$ is a set variable. $y \in \mathcal{Y}$ is a first-order Presburger variable, $\mathcal{Y} \cap \mathcal{X}_0 = \emptyset$. $L \in \Lambda$ is a node label. The formulae p of x/p are Presburger-closed, i.e., do not contain free variables from $\mathcal{Y}$. Intuitively, the assertion x/p means that the children of x satisfy constraint p where a term $[S]$ inside p is interpreted as the number of those children of x which are contained in S. Arithmetic expressions have their natural semantics.

Seidl et al. (2003) also provide an automaton model for PMSO, namely Presburger tree automata (PTA). We explain this automaton model here, because we will use it in subsequent proofs.

Given a finite set Q of states, we consider the canonical set $\mathcal{Y}_Q$ of variables which are indexed by elements in Q, i.e., $\mathcal{Y}_Q = \{y_q \mid q \in Q\}$. A Presburger tree automaton is a quadruple $\mathcal{A} = (Q, \Lambda, \delta, F)$ where

- Q is a finite set of states,
- $F \subseteq Q$ is the set of accepting states,
- Λ is the set of node labels, and
- δ maps pairs (q, L) of states and labels to Presburger constraints with free variables from the set $\mathcal{Y}_Q$.

The formula $\varphi = \delta(q, L)$ represents the *pre-condition* on the children of a node labelled by L for the transition into state q where the possible values of the variables y_p represent the admissible multiplicatives of the state p on the children. We introduce a satisfaction relation $t \models_{\mathcal{A}} q$ between a tree t and a state q as follows. Assume that $t = L(\{t_1, \dots, t_k\})$ and $\delta(q, L) = \varphi$. Then $t \models_{\mathcal{A}} \varphi$ iff there are k cardinalities n_j, and k states $p_j \in Q$ such that

- $t_j \models_{\mathcal{A}} p_j$ for $i \leq j \leq k$, and
- $\{y_{p_j} \mapsto n_j \mid 1 \leq j \leq n\} \models \varphi$.

The language $L(\mathcal{A})$ of unordered unranked trees which is accepted by the automaton $\mathcal{A}$ is given by

$$L(\mathcal{A}) = \{t \mid \exists q \in F : t \models_A q\}.$$

A tree language L is PMSO-definable iff it is accepted by some Presburger tree automaton (Seidl et al., 2003).

We also consider a subclass of Presburger constraints, namely *unary ordering* constraints. An ordering constraint is defined as

$$\begin{array}{rcl} p & ::= & t \leq t \mid p \wedge p \mid \neg p \\ t & ::= & y \mid n \mid t + t \end{array}$$

There is no existential quantification. An atomic constraint $t \leq t'$ is called unary iff it contains only one variable (but potentially several occurrences of this one variable). A Presburger ordering constraint is called unary iff all its atomic constraints are unary. Note that a unary constraint may contain several different variables as long as all of its atomic constraints contain only one variable.

A Presburger tree automaton over unary ordering constraints is called a *unary ordering* PTA. Boneva and Talbot (2005) showed that a tree language L is MSO-definable iff there exists a unary ordering PTA that accepts L.

On the basis of results by Boneva and Talbot (2005), Courcelle (1990), Seidl et al. (2003), the following is known about the expressive power of the different automata logics over unordered unranked trees. Here and in the following, an inclusion $A \subseteq B$ means that every tree language definable in logic A is also definable in logic B. A proper inclusion $A \subsetneq B$ indicates that there exist tree languages definable in B which are *undefinable* in A.

$$\text{FO} \subsetneq \text{MSO} \subsetneq \text{CMSO} \subsetneq \text{PMSO}.$$

1.3.2 Transitive-Closure Logics

Transitive closure logic is the extension of FO by transitive closure operators. This extension is sensible because FO is known to be incapable of expressing transitive closures. Formally, let $k \in \mathbb{N}$ and R a binary relation over k-tuples

($R \subseteq M^k \times M^k$). Then

$$TC(R) := \bigcap \left\{ W \mid R \subseteq W \subseteq M^k \times M^k, \forall \overline{x}, \overline{y}, \overline{z} \in M^k : \begin{array}{c} (\overline{x},\overline{y}), (\overline{y},\overline{z}) \in W \\ \Longrightarrow (\overline{x},\overline{z}) \in W \end{array} \right\}.$$

Deterministic transitive closure is the transitive closure of a deterministic, i.e., functional relation. For an arbitrary binary relation R over k-tuples we define its *deterministic reduct* by $R_D := \{(\overline{x},\overline{y}) \in R \mid \forall \overline{z} : (\overline{x},\overline{z}) \in R \Longrightarrow \overline{y} = \overline{z}\}$. Now $DTC(R) := TC(R_D)$.

The formulae of TC are defined by adding to first-order logic the transitive closure operator (TC):

If φ is a TC formula, $\overline{x} = x_1, \ldots, x_n$, $\overline{y} = y_1, \ldots, y_n$ are a subset of the free variables of φ such that $\forall i, j, x_i \neq y_j$, and $\overline{s} = s_1, \ldots, s_n, \overline{t} = t_1, \ldots, t_n$ are terms, then $[\mathrm{TC}_{\overline{x},\overline{y}}\, \varphi](\overline{s},\overline{t})$ is a TC formula.

For DTC we add the deterministic transitive closure operator. If φ is a DTC formula, then $[\mathrm{DTC}_{\overline{x},\overline{y}}\, \varphi](\overline{s},\overline{t})$ is a DTC formula.

A predicate of the form $[\mathrm{TC}_{\overline{x},\overline{y}}\, \varphi]$ ($[\mathrm{DTC}_{\overline{x},\overline{y}}\, \varphi]$) is supposed to denote the (deterministic) transitive closure of the relation defined by φ.

We also consider the special case where the transitive closure is restricted to binary relations, i.e., the tuple size is 1. These logics are denoted as MTC (where M stands for *monadic*) and MDTC.

We just mention in passing that for every formula in DTC there exists an equivalent formula in TC (see, e.g., (Immerman, 1999)).

1.3.3 Fixed-Point Logics and Infinitary Logics

The concept of adding transitive closure operators to FO can be generalised to adding fixed-point operators. Indeed, the transitive closure is a particularly simple type of a fixed-point operator. In this paper, we will consider least fixed-points, inflationary fixed-points and partial fixed-points. More explanation on these logics can be found in (Ebbinghaus and Flum, 1995, Immerman, 1999, Libkin, 2004).

Let M be a set. An operator on M is a mapping $F : \wp(M) \to \wp(M)$. An operator F is called *monotone*, if $X \subseteq Y$ implies $F(X) \subseteq F(Y)$, and *inflationary*, if $X \subseteq F(X)$ for all $X, Y \in \wp(M)$. Monotone operators are known to have *least fixed-points* (Tarski-Knaster-Theorem). For $F : \wp(M) \to \wp(M)$ monotone we define $\mathrm{LFP}(F) = \bigcap\{X \mid X = F(X)\}$.

Inflationary operators also have fixed-points. This fact is used to transform an arbitrary operator G into a fixed-point operator by making it inflationary. Simply set $G_{\mathrm{infl}}(X) = X \cup G(X)$. Now for $X^0 = \emptyset$ and $X^{i+1} = X^i \cup G(X^i)$ set $\mathrm{IFP}(G) = \bigcup_{i=0}^{\infty} X^i$.

Finally consider an arbitrary operator $F : \wp(M) \to \wp(M)$ and the sequence $X^0 = \emptyset$ and $X^{i+1} = F(X^i)$. This sequence need not be inflationary. It hence need not have a fixed-point. Hence we define the partial fixed-point of F as

$\mathrm{PFP}(F) = X^n$ if $X^n = X^{n+1}$ and $\mathrm{PFP}(F) = \emptyset$ if $X^n \neq X^{n+1}$ for all $n \leq 2^{|M|}$.

These operators will now be added to FO in the following way. Let R be a relational variable of arity k. For each tree $t = (V, E, \lambda)$ the formula $\varphi(R, \overline{x})$ where $|\overline{x}| = k$ gives rise to an operator $F_\varphi : \wp(V^k) \to \wp(V^k)$ defined as

$$F_\varphi(X) = \{\overline{v} \mid t \models \varphi(X/R, \overline{v})\}.$$

Now let $\varphi(R, \overline{x})$ be a formula where $|\overline{x}| = |\overline{t}| = k$. Then $[\mathrm{IFP}_{R,\overline{x}}\, \varphi(R, \overline{x})](\overline{t})$ is a formula of IFP, $[\mathrm{LFP}_{R,\overline{x}}\, \varphi(R, \overline{x})](\overline{t})$ is a formula of LFP (assuming R to be positive in φ), and $[\mathrm{PFP}_{R,\overline{x}}\, \varphi(R, \overline{x})](\overline{t})$ is a formula of PFP. Note that Gurevich and Shelah (1986) showed IFP = LFP.

The infinitary logic $\mathcal{L}_{\infty\omega}$ is the extension of FO by arbitrary infinite disjunctions and conjunctions. If Ψ is a set of formulae then $\bigvee \Psi$ and $\bigwedge \Psi$ are formulae. Because $\mathcal{L}_{\infty\omega}$ is known to be much too powerful, we are interested here in a particular sublogic of $\mathcal{L}_{\infty\omega}$, namely one in which each formula contains only *finitely* many different variables.

The class of $\mathcal{L}_{\infty\omega}$ formulae that use at most k distinct variables will be denoted $\mathcal{L}^k_{\infty\omega}$. And the finite variable infinitary logics $\mathcal{L}^\omega_{\infty\omega}$ is defined by

$$\mathcal{L}^\omega_{\infty\omega} = \bigcup_{k \in \mathbb{N}} \mathcal{L}^k_{\infty\omega}.$$

This logic is interesting because it comprises the fixed point logics LFP, IFP, and PFP, i.e., every class of finite structures definable in one of these logics is definable in $\mathcal{L}^\omega_{\infty\omega}$. The following diagram shows the expressive power of the logics defined in the last two subsections on finite unordered unranked trees. The inclusions are a consequence of the definitions of the logics.

$$\begin{array}{ccccccccc}
\mathrm{DTC} & \subseteq & \mathrm{TC} & \subseteq & \mathrm{LFP} & \subseteq & \mathrm{PFP} & \subseteq & \mathcal{L}^\omega_{\infty\omega} \\
\rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & & & \\
\mathrm{MDTC} & \subseteq & \mathrm{MTC} & \subseteq & \mathrm{MLFP} & & & &
\end{array}$$

Furthermore, Dawar et al. (1995) showed that LFP $\subsetneq \mathcal{L}^\omega_{\infty\omega}$.

1.3.4 Second-Order Logics

In this section we introduce three variants of second-order logics. Full second-order logic (denoted SO) is the extension of FO by arbitrary relation variables and arbitrary (second-order) quantification over these variables.

Existential second-order logic (ESO) is a restriction of SO. In ESO all second-order variables are globally existentially quantified. They are not involved in any quantifier alternation. Hence an ESO-formula consists of a prefix of existential second-order quantifications only and a FO-formula with SO-variables, but without SO-quantification.

The third logic of this section is SO with second-order transitive closure, denoted SO(TC). It was introduced by Immerman (1999) as a logic that strongly captures PSPACE, i.e., the logic and the complexity class have the same expressive power on arbitrary (finite) structures, not just ordered structures. We will only make use of this logic as a logic that strongly captures PSPACE. Hence we will not give a full definition, rather refer the interested reader to the given reference.

Second-order logics are certainly full logics in their own right. But they also have a strong connection to complexity theory. Actually, descriptive complexity theory was initiated by Fagin's result showing that ESO strongly captures NPTIME (Fagin, 1975). The logic SO strongly captures PH, the polynomial hierarchy, and SO(TC) strongly captures PSPACE (see, e.g., (Immerman, 1999)).

It follows immediately from the definitions that

$$\mathrm{FO} \subsetneq \mathrm{ESO} \subseteq \mathrm{SO} \subseteq \mathrm{SO(TC)}.$$

Whether any of these inclusions are strict are famous open problems in complexity theory.

1.3.5 Overview

We close this section with an overview over what is known about the expressive power of the different logics defined above on finite unordered unranked trees (see Figure 1).

Let us explain those parts of Figure 1 that have not yet been justified proceeding from bottom to top.

MLFP $\subseteq$ MSO Every monadic least fixed point is expressible in MSO. See (Ebbinghaus and Flum, 1995).

TC $\subsetneq$ SO(TC) On ordered structures, TC captures NLOGSPACE. The proof of this theorem also shows that TC $\subseteq$ NLOGSPACE on arbitrary structures. Since SO(TC) strongly captures PSPACE = NPSPACE, the proper inclusion follows from the space hierarchy theorem.

PMSO $\subseteq$ ESO Seidl et al. (2003) show that any PMSO definable tree language is recognised in (deterministic) linear time. Since ESO strongly captures NPTIME, the inclusion follows.

LFP $\subseteq$ ESO On ordered structures, LFP captures PTIME. The proof of this theorem also shows that LFP $\subseteq$ PTIME on arbitrary structures. Since ESO strongly captures NPTIME, the inclusion follows.

1.4 Two Initial Results

We start with two smaller results. The first one states that even the weakest logic extending FO, namely MDTC, is truly more powerful than FO.

Theorem 1 *The logic* MDTC *is strictly more powerful than* FO *over unordered unranked trees.*

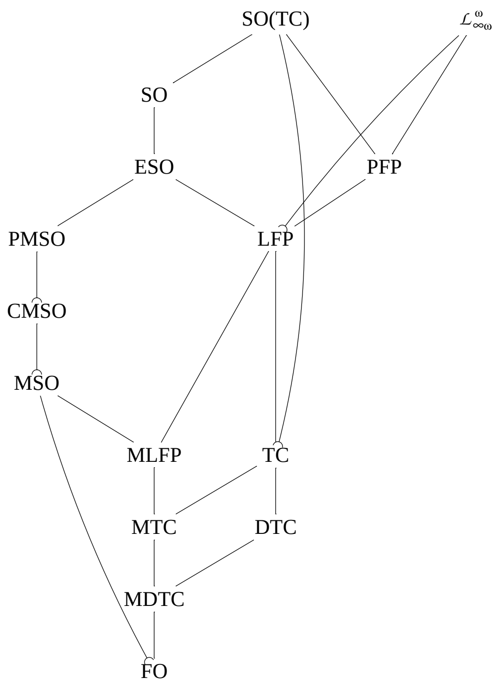


FIGURE 1 Logics for finite unordered unranked trees: the base.
⊃— indicates a proper inclusion.

A tree language undefinable in FO, but definable in MDTC is one where each leaf node is at an even depth level. The second result concerns the expressive power of MSO and MLFP.

Theorem 2 *The logics* MSO *and* MLFP *have the same expressive power over unordered unranked trees.*

It can be shown that an accepting run of a unary ordering PTA can be logically rendered in MLFP. This way the more important direction of the theorem is proven. That MSO can express monadic least fixed points is mentioned just above.

1.5 Separating Automata Logics and Fixed-Point Logics

The aim of this section is to separate automata logics from transitive closure logics and fixed-point logics. This is done in two subparts. In the first one we present a tree language that is DTC-definable, but not PMSO-definable. In the second part we present a tree language that is MSO-definable, but not TC-definable.

1.5.1 A DTC-Definable Tree Language

In this section we present a tree language which is DTC-definable but not PMSO-definable (and therefore neither CMSO-definable nor MSO-definable). It is a variation of a tree language defined by Tiede and Kepser (2006). We have the following node labels f, g where f labels the root, g is the label for all other nodes. The language is defined as $L_1 = \{f\{g^n, g^n\} \mid n \in \mathbb{N}^+\}$. It is the language of two g-chains of equal length below the root.

The language L_1 is definable in DTC as follows. Let

$$Root(x) := \neg\exists y E(y,x)$$

define the root of a tree and

$$Leaf(x) := \neg\exists y E(x,y)$$

define a leaf in the tree. The formula

$$OneCh(x) := \exists y E(x,y) \wedge \forall z (E(x,z) \rightarrow z = y)$$

expresses that node x has exactly one child. Consider the following predicate P:

$$[\mathrm{DTC}_{(y_1,y_3),(y_2,y_4)}\, E(y_1,y_2) \wedge E(y_3,y_4)]$$

which states that y_2 is at the same distance from y_1 as y_4 from y_3. Let $\varphi(x_1,x_2)$ be the formula

$$\begin{aligned}\forall y_1, y_2\, P(x_1,x_2,y_1,y_2) \rightarrow\ & (g(y_1) \wedge g(y_2) \wedge \\ & (Leaf(y_1) \wedge Leaf(y_2)) \vee \\ & (OneCh(y_1) \wedge OneCh(y_2))\end{aligned}$$

expressing that if y_1 is at the same distance from x_1 as y_2 from x_2 then both are labelled with g and either both are leaves or both have exactly one child. Now the tree language is given by

$$\begin{aligned}\exists r, x_1, x_2\ & Root(r) \wedge f(r) \wedge E(r,x_1) \wedge E(r,x_2) \wedge g(x_1) \wedge g(x_2) \wedge \\ & x_1 \neq x_2 \wedge \forall z\, E(r,z) \rightarrow (z = x_1 \vee z = x_2) \wedge \\ & \varphi(x_1,x_2)\end{aligned}$$

The formula says that r is the root, labelled f and that r has exactly two children x_1 and x_2 both labelled g and φ holds for x_1 and x_2.

It is known that this tree language is *not* MSO-definable. It can be shown that it is not even PMSO-definable. The proof method is a variant of the proof for the pumping lemma for recognisable tree languages adopted to unordered unranked trees and PTA.

Proposition 3 *The tree language L_1 is DTC-definable, but is* not *PMSO-definable.*

PROOF. Suppose $\mathcal{A} = (Q, \Lambda, \delta, F)$ is a tree automaton accepting L_1 and $k = |Q|$ is the number of states. Let $m > k$. Consider the tree $t = f\{g^m, g^m\} \in L_1$ and in particular its subtree g^m. Since $m > k$ there must be a tree $t' = g^{l_1}$ a non-empty context $C = g^{l_2}\{\bullet\}$ and a context $C' = g^{l_3}\{\bullet\}$ and a state $q \in Q$

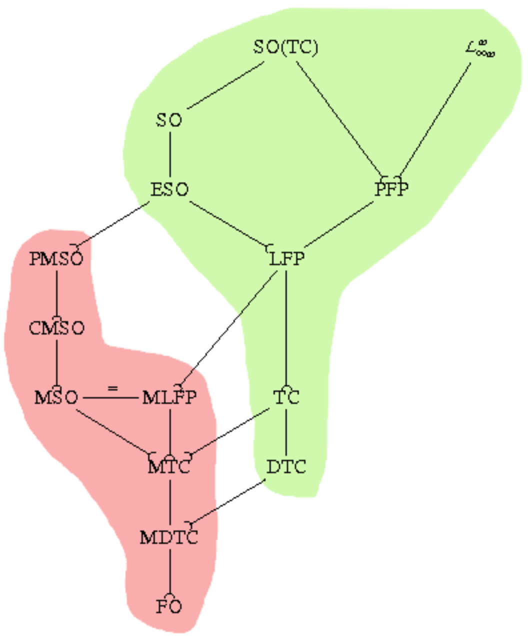


FIGURE 2 Separating automata logics from fixed-point logics.

such that $l_1 + l_2 + l_3 = m$ and $g^m = C'\{C\{t'\}\}$ and both the root of t' and $C\{t'\}$ receive state q in an accepting run for t.

Therefore $u = f\{g^m, C'\{C\{C\{t'\}\}\}\}$ is accepted by $\mathcal{A}$ because both $C\{t'\}$ and $C\{C\{t'\}\}$ receive state q in an accepting run.
But $u \notin L_1$. □

The results of this subsection are depicted in Figure 2. Logics in the green area are capable of defining L_1, whereas logics in the red area are not.

Theorem 4 *The following inclusions are* strict.

- MDTC *is* strictly *less powerful than* DTC.
- MTC *is* strictly *less powerful than* TC.
- MLFP *is* strictly *less powerful than* LFP.
- PMSO *is* strictly *less powerful than ESO.*

1.5.2 An MSO-Definable Tree Language

Consider the following tree language. It is originally defined in (Ebbinghaus and Flum, 1995) as a class of finite graphs. All leaves are labelled either with 0 or 1. All internal nodes are labelled with B for blank, some void node label that is there only because we demand all nodes to be labelled. The leaf labels 0 and 1 are interpreted as false and true (resp.). Internal nodes function as gates. They are set to true iff exactly one child node is set to false. We consider the class of trees whose root node is evaluated to true.

Formally we define two tree languages inductively as follows. Let $\Lambda = \{0,1,B\}$ be a set of labels. The tree languages L_2 and L_3 are the smallest sets such that

$$\begin{array}{rcl} 0 & \in & L_3 \\ 1 & \in & L_2 \\ B(L') & \in & L_3 \text{ where } L' \subseteq_{mfin} L_2 \\ B(\{t\} \uplus L') & \in & L_2 \text{ where } t \in L_3 \text{ and } L' \subseteq_{mfin} L_2 \\ B(\{t,t'\} \uplus L' \uplus L'') & \in & L_3 \text{ where } t,t' \in L_3, L' \subseteq_{mfin} L_3, \text{ and } L'' \subseteq_{mfin} L_2. \end{array}$$

The tree language L_2 is recognised by the following Presburger tree automaton $\mathcal{A} = (\{q_t,q_f\},\Lambda,\delta,\{q_t\})$ where

$$\begin{array}{rclcrcl} \delta(0,q_f) & = & true & & \delta(0,q_t) & = & false \\ \delta(1,q_f) & = & false & & \delta(1,q_t) & = & true \\ \delta(B,q_f) & = & y_{q_f} = 0 \vee y_{q_f} \geq 2 & & \delta(B,q_t) & = & y_{q_f} = 1 \end{array}$$

Hence L_2 is PMSO-definable. A close inspection of δ reveals that all constraints in the transitions are unary ordering constraints. Hence L_2 is even MSO-definable.

Proposition 5 *There exists a tree language which is* MSO-*definable, but is* not TC-*definable.*

The proof that the tree language L_2 is not TC-definable is an application of results by Grohe (1994), reported in (Ebbinghaus and Flum, 1995, Chap. 7.6.3).

The results of this subsection are depicted in Figure 3. Logics in the green area are capable of defining L_2, whereas logics in the red area are not.

Theorem 6 *The following inclusions are* strict*.*

- MTC *is* strictly *less powerful than* MLFP *and* MSO*.*
- TC *is* strictly *less powerful than* LFP*.*

Theorem 7 *The logics* (P)MSO *and* TC *are incomparable over the class of finite unordered unranked trees.*

1.6 Separating Fixed-Point Logics and Second-Order Logics

The main result of this section is that there is a tree languages definable in CMSO that is not $\mathcal{L}^{\omega}_{\infty\omega}$-definable. We use the well known fact that $\mathcal{L}^{\omega}_{\infty\omega}$ is not

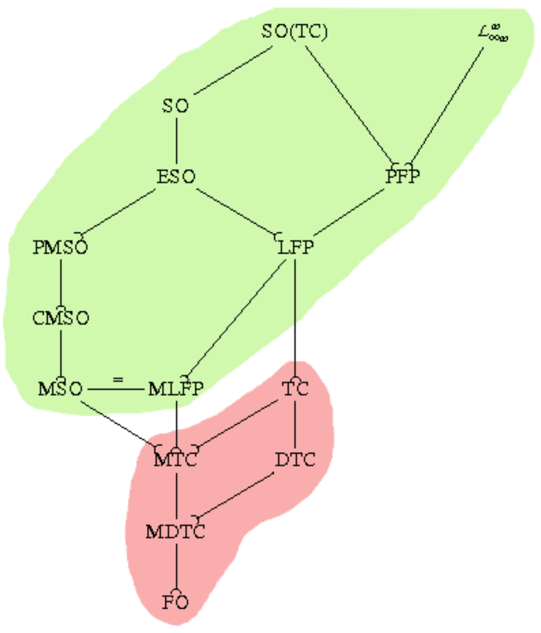


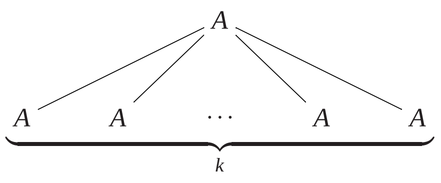
FIGURE 3 Separating automata logics from fixed-point logics.

particularly good at counting.

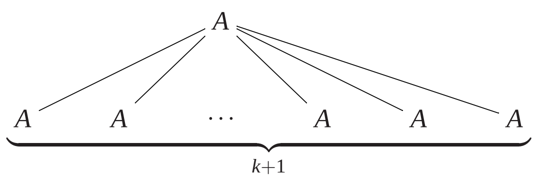
Let $\Lambda = \{A\}$. Define the tree language $L_4 = \{(V,E,\lambda) \mid |V| = 2n$ for some $n \in \mathbb{N}\}$ as the set of all tree with an even number of nodes (where each node is labelled with A). We first show that L_4 is CMSO-definable. The following formula defines L_4.

$$\exists X\,(\forall x.x \in X \wedge Mod^2_0(X))$$

We will next show that L_4 is not $\mathcal{L}^{\omega}_{\infty\omega}$-definable using infinite pebble games. For the definition of this type of games, the reader is referred to, e.g., (Libkin, 2004, Chap. 11.2). For a natural number k define $\mathfrak{A}_k$ to be



and $\mathfrak{B}_k$ to be



If k is even then $\mathfrak{A}_k$ has an odd number of nodes while $\mathfrak{B}_k$ has an even number of nodes. If k is odd then $\mathfrak{A}_k$ has an even number of nodes while $\mathfrak{B}_k$ has an odd number of nodes.

Lemma 8 *The duplicator has a winning strategy for the infinite pebble game* $\mathrm{PG}_k^\infty(\mathfrak{A}_k, \mathfrak{B}_k)$ *for every* $k \in \mathbb{N}$.

PROOF. Let $(a_1, \ldots, a_k) \mapsto (b_1, \ldots, b_k)$ be a partial isomorphism between $\mathfrak{A}_k$ and $\mathfrak{B}_k$. We assume no two pebbles are ever placed on the same node, because doing so leads to a game with less than k pebbles. We also assume that the spoiler never leaves a pebble in its place when making a move, because if he did, the duplicator would do the same and the move would be void.

Assume the spoiler chooses $\mathfrak{B}_k$ and to reposition pebble j. We distinguish the following cases.
Case 1: There is a pebble on the root of $\mathfrak{B}_k$.
Since $(a_1, \ldots, a_k) \mapsto (b_1, \ldots, b_k)$ is a partial isomorphism, there is a l with $1 \leq l \leq k$ such that b_l is the pebble on the root of $\mathfrak{B}_k$ and a_l is a pebble on the root of $\mathfrak{A}_k$.
Case 1.1: $j = l$, i.e., the spoiler chooses the pebble on the root.
Since there is now no pebble on the root of $\mathfrak{B}_k$, the substructure $(b_1, \ldots, b_k)$ is now a discrete structure of k elements. Since $\mathfrak{A}_k$ has k leaves and one pebble is placed on the root of $\mathfrak{A}_k$ there must be an unpebbled leaf of $\mathfrak{A}_k$. The duplicator places his j-th pebble on this leaf. Now $(a_1, \ldots, a_k)$ is also a discrete structure and $(a_1, \ldots, a_k) \mapsto (b_1, \ldots, b_k)$ is a partial isomorphism.
Case 1.2: $j \neq l$, i.e., the spoiler chooses a pebble on one of the leaves.
The spoiler moves pebble j onto an unpebbled leaf. The resulting substructure induced by $(b_1, \ldots, b_k)$ is obviously isomorphic to the one before the move. Actually, it is $\mathfrak{B}_{k-2} \cong \mathfrak{A}_{k-1}$. Since the substructure induced by $(a_1, \ldots, a_k)$ is also $\mathfrak{A}_{k-1}$, the duplicator leaves all his pebbles in place and $(a_1, \ldots, a_k) \mapsto (b_1, \ldots, b_k)$ is a partial isomorphism.
Case 2: There is no pebble on the root of $\mathfrak{B}_k$.
Both $(b_1, \ldots, b_k)$ and $(a_1, \ldots, a_k)$ are discrete structures.
Case 2.1: The spoiler moves pebble j onto the root of $\mathfrak{B}_k$.
The induced structure of $(b_1, \ldots, b_k)$ is now $\mathfrak{B}_{k-2} \cong \mathfrak{A}_{k-1}$. The duplicator mimics this move moving his pebble j onto the root of $\mathfrak{A}_k$. Now the induced structure of $(a_1, \ldots, a_k)$ is also $\mathfrak{A}_{k-1}$ and $(a_1, \ldots, a_k) \mapsto (b_1, \ldots, b_k)$ is a partial

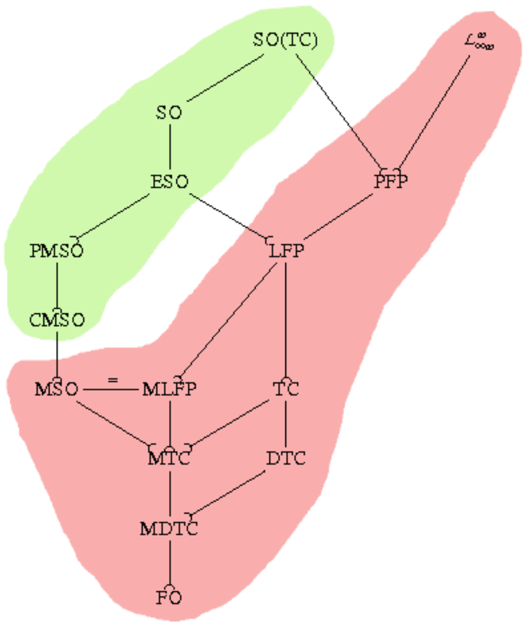


FIGURE 4 Separating fixed-point logics from second-order logics.

isomorphism.
Case 2.2: The spoiler moves pebble j onto an unpebbled leaf of $\mathfrak{B}_k$.
Then $(b_1,\dots,b_k)$ remains a discrete structure. Thus it is already isomorphic to $(a_1,\dots,a_k)$, and the duplicator leaves all his pebbles in place.

The argument for the situation where the spoiler chooses to move on structure $\mathfrak{A}_k$ is analogous, actually simpler. □

The lemma implies that $\mathfrak{A}_k \models \varphi$ iff $\mathfrak{B}_k \models \varphi$ for every $k \in \mathbb{N}$ and $\varphi \in \mathcal{L}^{k}_{\infty\omega}$.

Proposition 9 *The tree language L_4 of trees with an even number of nodes is CMSO-definable, but is* not *$\mathcal{L}^{\omega}_{\infty\omega}$-definable.*

PROOF. Suppose L_4 were $\mathcal{L}^{\omega}_{\infty\omega}$-definable, i.e, there were a formula $\varphi \in \mathcal{L}^{\omega}_{\infty\omega}$ that defined L_4. By definition of $\mathcal{L}^{\omega}_{\infty\omega}$ there is a $k \in \mathbb{N}$ such that $\varphi \in \mathcal{L}^{k}_{\infty\omega}$. By the above lemma, either $\mathfrak{A}_k \models \varphi$ and $\mathfrak{B}_k \models \varphi$ or $\mathfrak{A}_k \not\models \varphi$ and $\mathfrak{B}_k \not\models \varphi$. But one of $\mathfrak{A}_k, \mathfrak{B}_k$ has an even number of nodes, while the other has an odd number of nodes. □

The results of this section are summarised in Figure 4. Logics in the green area can define L_4 whereas logics in the red one cannot.

Theorem 10 *The following inclusions are* strict.

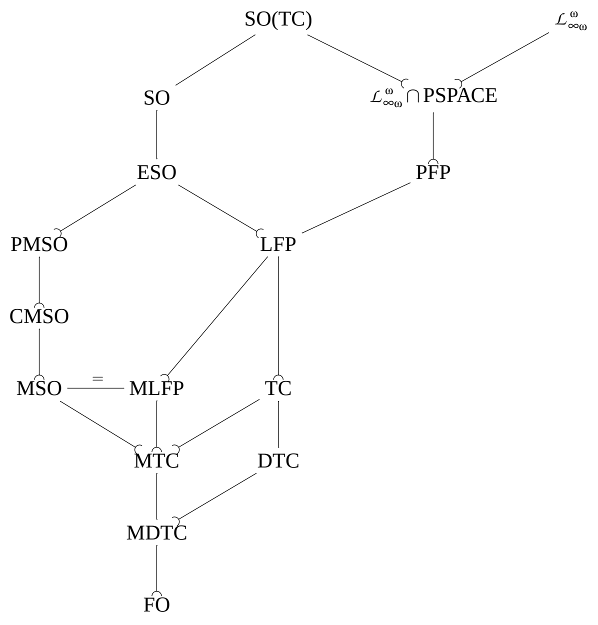

SO(TC)
$\mathcal{L}^{\omega}_{\infty\omega}$
SO
$\mathcal{L}^{\omega}_{\infty\omega} \cap$ PSPACE
ESO
PFP
PMSO
LFP
CMSO
MSO
=
MLFP
TC
MTC
DTC
MDTC
FO

FIGURE 5 Logics for finite unordered unranked trees.
$\supset\!\!-$ indicates a proper inclusion.

- PFP *is* strictly *less powerful than* SO(TC).
- LFP *is* strictly *less powerful than ESO.*
- MSO *is* strictly *less powerful than* CMSO.

The last result is already known. We just provided an alternative proof of the result.

1.7 Conclusion

Figure 5 depicts a landscape of the expressive power of different logics for finite unordered unranked trees. This includes the relationship between PFP and $\mathcal{L}^{\omega}_{\infty\omega}$, which we have not been able to present here due to space restrictions. As one can see, most inclusions between different logics turn out to be proper.

An important result one can see from this picture is that the automata logics are largely incomparable to the logics stemming from descriptive complexity theory (TC, LFP, PFP).

Most of the remaining open questions are directly related to difficult open problems in complexity theory. This is true for the second-order logics, but also concerns the transitive closure logics. Also, the separation of LFP from

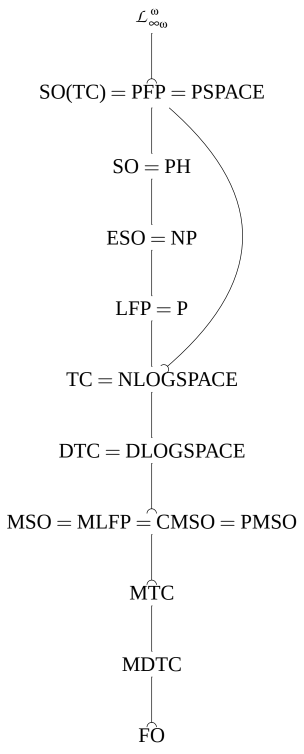


FIGURE 6 Logics for finite ordered ranked trees.
$\supset\!\!-$ indicates a proper inclusion.

PFP amounts to the separation of PTIME from PSPACE by the Abiteboul-Vianu theorem (Abiteboul and Vianu, 1995).

Appendix: The Situation for Finite Ordered Ranked Trees

For comparison we also show what is known about the expressive power of the above mentioned logics on finite ordered ranked trees. Most questions on whether or not inclusions are proper are open. This is probably due to the fact that they are directly related to famous open problems in classical complexity theory. Figure 6 summarises the results.

There are only few known non-trivial results of proper inclusion. Kolaitis and Vardi (1992) showed that PFP $\subsetneq \mathcal{L}^{\omega}_{\infty\omega}$. The proper inclusion TC $\subsetneq$ PFP follows from the space hierarchy theorem. Tiede and Kepser (2006) showed that MSO $\subsetneq$ DTC. Recently, ten Cate and Segoufin (2008) were able to show that also MTC $\subsetneq$ MSO.

References

Abiteboul, Serge, Peter Buneman, and Dan Suciu. 2000. *Data on the Web*. Morgan Kaufmann.

Abiteboul, Serge and Victor Vianu. 1995. Computing with first-order logic. *Journal of Computer and System Sciences* 50:309–335.

Boneva, Iovka and Jean-Marc Talbot. 2005. Automata and logics over unranked and unordered trees. In J. Giesl, ed., *Proceedings RTA 2005*, LNCS 3467, pages 500–515. Springer.

Courcelle, Bruno. 1990. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85:12–75.

Dawar, Anuj, Steven Lindell, and Scott Weinstein. 1995. Infinitary logic and inductive definability over finite structures. *Information and Computation* 119(2):160–175.

Ebbinghaus, Heinz-Dieter and Jörg Flum. 1995. *Finite Model Theory*. Springer-Verlag.

Fagin, Ronald. 1975. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 21:89–96.

Grohe, Martin. 1994. *The Structure of Fixed-Point Logics*. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg.

Gurevich, Yuri and Saharon Shelah. 1986. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic* 32:265–280.

Immerman, Neil. 1999. *Descriptive Complexity*. Springer.

Kolaitis, Phokion G. and Moshe Y. Vardi. 1992. Infinitary logics and 0-1 laws. *Information and Computation* 98(2):258–294.

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

Seidl, Helmut, Thomas Schwentick, and Anca Muscholl. 2003. Numerical document queries. In T. Milo, ed., *Proc. 22nd Symposium on Principles of Database Systems (PODS 2003)*, pages 155–166. ACM.

ten Cate, Balder and Luc Segoufin. 2008. XPath, transitive closure logic, and nested tree walking automata. In *Proceedings PODS 2008*.

Tiede, Hans-Jörg and Stephan Kepser. 2006. Monadic second-order logic over trees and deterministic transitive closure logics. In G. Mints, ed., *13th Workshop on Logic, Language, Information and Computation*, ENTCS 165, pages 189–199. Springer.