A Landscape of Logics for Finite Unordered Unranked Trees

Stephan Kepser
Collaborative Research Centre 441,
University of Tübingen, Germany
kepser@sfs.uni-tuebingen.de

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 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 are the underlying data structures of 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 recall the definitions of all logics of this paper in Section 3. Section 4 provides two simple results to start with. Section 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 6. A refinement of the result in this section is provided in the next section, where we consider complexity classes in addition to logics. The relationship between partial fixed-point logic and infinitary logic is closer investigated in Section 8. We close the paper with an overview of the results obtained in Section 9 and a comparison to the situation for ordered ranked trees in Section 10.

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 tree automata and their 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 multi-set of N (where N may also be a set).

The notion of a context, familiar from ordered trees, can also be extended to unordered trees. A context is a tree with a hole in it. Let $\bullet \notin \Lambda$. A context is a tree with a single occurrence of $\bullet$ and $\bullet$ occurs at a leaf. Formally, $\bullet$ is a context (the trivial context) and if C is a context, $t_1, \ldots, t_k$ are unordered trees, and $L \in \Lambda$ then $L\{C, t_1, \ldots, t_k\}$ is a context. If C is a context and t an unordered tree, we write $C(t)$ for replacing $\bullet$ in C with t.

3 The Logics

3.1 First-Order Logic

First-order logic is the weakest logic we study. 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 $v \in V$.

Definition 1 Let $\mathcal{X}_0 = \{x_0, x_1, x_2, \ldots\}$ be a denumerably infinite set of first-order variables. The first-order formulae over trees are

$$L(x) \mid E(x,y) \mid x = y \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

where $x, y \in \mathcal{X}_0$ and φ and ψ are formulae.

The semantics of first-order formulae is defined as usual. A variable assignment $\alpha : \mathcal{X}_0 \to V$ maps first-order variables to tree nodes. Let $t = (V, E, (L)_{L \in \Lambda})$ be a tree.

- $t, \alpha \models L(x)$ iff $L(\alpha(x))$,
- $t, \alpha \models E(x,y)$ iff $(\alpha(x), \alpha(y)) \in E$,
- $t, \alpha \models x = y$ iff $\alpha(x) = \alpha(y)$,
- $t, \alpha \models \neg\varphi$ iff $[\![\varphi]\!]^{\alpha} = \mathit{false}$,
- $t, \alpha \models \varphi \wedge \psi$ iff $t, \alpha \models \varphi$ and $t, \alpha \models \psi$,

- $t, \alpha \models \varphi \vee \psi$ iff $t, \alpha \models \varphi$ or $t, \alpha \models \psi$,
- $t, \alpha \models \exists x.\varphi$ iff there is a $v \in V$ such that $t, \alpha[x \mapsto v] \models \varphi$,
- $t, \alpha \models \forall x.\varphi$ iff for all $v \in V$ $t, \alpha[x \mapsto v] \models \varphi$.

3.2 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 trees we will see that different types of tree automata with different expressive power can be defined. The automata and logics definitions that follow are taken from (Boneva and Talbot, 2005) and (Seidl, Schwentick, and Muscholl, 2003).

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

Definition 2 Let $\mathcal{X}_2 = \{X_0, X_1, X_2, \dots\}$ be a denumerable infinite set of set variables. The MSO formulae are all FO formulae and

$$x \in X \mid X \subseteq Y \mid \exists X.\varphi \mid \forall X.\varphi$$

where $x \in \mathcal{X}_0, X, Y \in \mathcal{X}_1$ and φ is an MSO formula. A variable assignment α now consists of a first-order assignment $\mathcal{X}_0 \to V$ and a set assignment $\mathcal{X}_2 \to \wp(V)$.

- $t, \alpha \models x \in X$ iff $\alpha(x) \in \alpha(X)$,
- $t, \alpha \models X \subseteq Y$ iff $\alpha(X) \subseteq \alpha(Y)$,
- $t, \alpha \models \exists X.\varphi$ iff there is a set $W \subseteq V$ such that $t, \alpha[X \mapsto W] \models \varphi$,
- $t, \alpha \models \forall X.\varphi$ iff for all sets $W \subseteq V$ $t, \alpha[X \mapsto W] \models \varphi$.

Definition 3 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 for a tree t iff $|\alpha(X)| \mod i = j$, i.e., 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.

Definition 4 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. Formally we need a variable assignment $\beta : \mathcal{Y} \to \mathbb{N}$ of arithmetic variables to natural numbers. Arithmetic expressions have their natural semantics.

- $\beta \models y = n$ iff $\beta(y) = n$,
- $\beta \models x + y = z$ iff $\beta(x) + \beta(y) = \beta(z)$,
- $\beta \models \varphi \wedge \psi$ iff $\beta \models \varphi$ and $\beta \models \psi$,
- $\beta \models \neg\varphi$ iff $\beta \not\models \varphi$,
- $\beta \models \exists y.\varphi$ iff there is an $n \in \mathbb{N}$ such that $\beta[y \mapsto n] \models \varphi$.

Now

- $t, \alpha \models x/p$ iff $\beta \models p$ where
  $\beta[L] = |\{v \in V \mid E(\alpha(x), v), v \in L\}|$ and
  $\beta[X] = |\{v \in V \mid E(\alpha(x), v), v \in \alpha(X)\}|$.

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.

Definition 5 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 $t = L\{t_1, \ldots, t_k\}$ 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

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

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 Courcelle (1990), Boneva and Talbot (2005), and 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.

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

3.3 Transitive-Closure Logics

A fundamental restriction in the expressive power of first-order logic is the lack of any type of recursion mechanism. One of the simplest and most fundamental

queries that are not first-order expressible is the *transitive closure*. It assigns to a given binary relation E on a universe U its transitive closure, i.e., the set of all pairs $(x,y) \in U \times U$ such that there exist $z_0,\ldots,z_r \in U$ with $z_0 = x, z_r = y$ and $E(z_i, z_{i+1})$ for all $i < r$. It was first shown in Fagin (1975) that transitive closure is not expressible in FO.

Let M be a set and $R \subseteq M \times M$ a binary relation over M. The *transitive closure* $TC(R)$ of R is the smallest set containing R and for all $x,y,z \in M$ such that $(x,y) \in TC(R)$ and $(y,z) \in TC(R)$ we have $(x,z) \in TC(R)$, i.e.,

$$TC(R) := \bigcap \{W \mid R \subseteq W \subseteq M \times M, \forall x,y,z \in M : (x,y),(y,z) \in W \implies (x,z) \in W\}.$$

This notion can be extended to relations over tuples. Let $k \in \mathbb{N}$ and R a binary relation over k-tuples ($R \subseteq M^k \times M^k$). Then

$$TC(R) := \bigcap \{W \mid R \subseteq W \subseteq M^k \times M^k, \forall \bar{x},\bar{y},\bar{z} \in M^k : (\bar{x},\bar{y}),(\bar{y},\bar{z}) \in W \implies (\bar{x},\bar{z}) \in W\}.$$

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

$$R_D := \{(\bar{x},\bar{y}) \in R \mid \forall \bar{z} : (\bar{x},\bar{z}) \in R \implies \bar{y} = \bar{z}\}.$$

Now

$$DTC(R) := TC(R_D).$$

Since neither the transitive closure of a relation nor its deterministic counterpart are definable in FO, it makes sense to add these operators to first-order logic to extend its expressive power in moderate and controlled way.

Definition 6 The formulae of TC are defined by adding to first-order logic the transitive closure operator (TC):

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

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.

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

Definition 7 Let $t = (V, E, \lambda)$ be a tree. We define $t \models \varphi$ for TC or DTC in the usual way. To evaluate predicates defined with the transitive closure operator, we define

$$t, \alpha \models [\mathrm{TC}_{\bar{x},\bar{y}}\, \varphi](\bar{s}, \bar{t})$$

iff

$$(\alpha(\bar{s}), \alpha(\bar{t})) \in TC\{(\bar{a}, \bar{b}) \mid t, \alpha \models \varphi[\bar{a}, \bar{b}]\}.$$

And

$$t, \alpha \models [\mathrm{DTC}_{\bar{x},\bar{y}}\, \varphi](\bar{s}, \bar{t})$$

iff

$$(\alpha(\bar{s}), \alpha(\bar{t})) \in DTC\{(\bar{a}, \bar{b}) \mid t, \alpha \models \varphi[\bar{a}, \bar{b}]\}.$$

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

3.4 Fixed-Point 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 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

$$\mathrm{PFP}(F) = \begin{cases} X^n & \text{if } X^n = X^{n+1}, \\ \emptyset & \text{if } X^n \neq X^{n+1} \text{ for all } n \leq 2^{|M|}. \end{cases}$$

Definition 8 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 $\varphi(R, \bar{x})$ where $|\bar{x}| = k$ gives rise to an operator $F_\varphi : \wp(V^k) \to \wp(V^k)$ defined as

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

Now

- If $\varphi(R, \bar{x})$ is a formula where $|\bar{x}| = |\bar{t}| = k$ then $[\mathrm{IFP}_{R,\bar{x}}\, \varphi(R, \bar{x})](\bar{t})$ is a formula of IFP.
- If $\varphi(R, \bar{x})$ is a formula where R is positive in φ and $|\bar{x}| = |\bar{t}| = k$ then $[\mathrm{LFP}_{R,\bar{x}}\, \varphi(R, \bar{x})](\bar{t})$ is a formula of LFP.
- If $\varphi(R, \bar{x})$ is a formula where $|\bar{x}| = |\bar{t}| = k$ then $[\mathrm{PFP}_{R,\bar{x}}\, \varphi(R, \bar{x})](\bar{t})$ is a formula of PFP.

The semantics is defined as follows

- $t, \alpha \models [\mathrm{IFP}_{R,\bar{x}}\, \varphi(R, \bar{x})](\bar{t})$ iff $\alpha(\bar{t}) \in \mathrm{IFP}(F_\varphi)$,
- $t, \alpha \models [\mathrm{LFP}_{R,\bar{x}}\, \varphi(R, \bar{x})](\bar{t})$ iff $\alpha(\bar{t}) \in \mathrm{LFP}(F_\varphi)$,
- $t, \alpha \models [\mathrm{PFP}_{R,\bar{x}}\, \varphi(R, \bar{x})](\bar{t})$ iff $\alpha(\bar{t}) \in \mathrm{PFP}(F_\varphi)$,

Note that by the Gurevich and Shelah (1986) Theorem, IFP = LFP.

3.5 The Logic $\mathcal{L}^{\omega}_{\infty\omega}$

The 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. And tree $t \models \bigvee \Psi$ (resp. $\bigwedge \Psi$) iff $t \models \psi$ for some (resp. all) $\psi \in \Psi$. This logic is known to be much too powerful. In fact, every class of finite models of a given signature is definable in $\mathcal{L}_{\infty\omega}$ (see, e.g., (Ebbinghaus and Flum, 1995; Libkin, 2004)).

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.

Definition 9 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 $\mathcal{L}^{\omega}_{\infty\omega}$. Ehrenfeucht-Fraïssé games for $\mathcal{L}^{\omega}_{\infty\omega}$ are infinitary pebble games. These

are frequently a lot simpler to play than Ehrenfeucht-Fraïssé games for fixed-point logics. It is therefore frequently simpler to show that a certain tree language is undefinable in $\mathcal{L}^{\omega}_{\infty\omega}$ than it would be to show that this language is undefinable in a fixed-point logic.

The following inclusions are a consequence of the definitions of the logics defined in the last three subsections.

$$\begin{array}{ccccccccc} \mathrm{DTC} & \subseteq & \mathrm{TC} & \subseteq & \mathrm{LFP} & \subseteq & \mathrm{PFP} & \subseteq & \mathcal{L}^{\omega}_{\infty\omega} \\ \cup\mathrm{I} & & \cup\mathrm{I} & & \cup\mathrm{I} & & & & \\ \mathrm{MDTC} & \subseteq & \mathrm{MTC} & \subseteq & \mathrm{MLFP} & & & & \end{array}$$

Furthermore, Dawar, Lindell, and Weinstein (1995) showed that $\mathrm{LFP} \subsetneq \mathcal{L}^{\omega}_{\infty\omega}$. It is also known that on arbitrary finite unordered structures DTC and TC can be separated (Ebbinghaus and Flum, 1995, Chap. 7.6.1). But the proof does *not* extend to unordered unranked trees.

3.6 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.

Definition 10 For every positive natural number $n > 0$, let $\mathcal{R}_n = \{R_{n0}, R_{n1}, R_{n2}, \dots\}$ be a denumerable infinite set of relation variables of arity n. The following formulae are added to the first-order case.

$$R_{nk}(x_1, \dots, x_n) \mid \exists R_{nk}.\varphi \mid \forall R_{nk}.\varphi$$

for all $n > 0$ and $k \in \mathbb{N}$. A variable assignment for second-order logic extends a variable assignment for FO by assigning each relation variable of arity n a set of n-tuples of nodes. Hence $\alpha : \mathcal{X}_0 \to V \cup \bigcup_{n>0} \mathcal{R}_n \to V^n$. Let $t = (V, E, (L)_{L\in\Lambda})$ be a tree. Now

- $t, \alpha \models R_{nk}(x_1, \dots, x_n)$ iff $(\alpha(x_1), \dots, \alpha(x_n)) \in \alpha(R_{nk})$,
- $t, \alpha \models \exists R_{nk}.\varphi$ iff there is an $M \subseteq V^n$ such that $t, \alpha[R_{nk} \mapsto M] \models \varphi$,
- $t, \alpha \models \forall R_{nk}.\varphi$ iff for all $M \subseteq V^n$ it is true that $t, \alpha[R_{nk} \mapsto M] \models \varphi$.

The logic ESO (for Existential SO) is a restriction of SO. In ESO all second-order variables are globally existentially quantified. They are not involved in any quantifier alternation.

Definition 11 A first-order matrix ξ with second-order relation variables is defined as

$$L(x) \mid E(x,y) \mid x = y \mid R_{nk}(x_1,\ldots,x_n) \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

where $x, y, x_1, \ldots, x_n \in \mathcal{X}_0$, $R_{nk} \in \mathcal{R}_n$ and φ and ψ are FO-matrices. That is ξ is an FO-formula, that may contain SO-relations, but must not contain SO-quantification. Now, if ξ is an FO-matrix with SO-variables $R_{n_1k_1}, \ldots, R_{n_lk_l}$, then $\exists R_{n_1k_1} \ldots \exists R_{n_lk_l}.\xi$ is a ESO-formula.

The semantics of formulae is the same as for SO.

ESO is sometimes denoted as Σ^1_1.

The third logic of this section is SO with second-order transitive closure, denoted SO(TC). It was introduced by Immerman (1999, Chap. 10.4) as a logic that strongly captures[1] PSPACE. 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 reference above. Let φ be a formula with free variables $R_1, \ldots, R_k, R'_1, \ldots, R'_k, x_1, \ldots, x_l, x'_1, \ldots, x'_l$ where $k, l \in \mathbb{N}, x_1, \ldots, x_l, x'_1, \ldots, x'_l$ are first-order variables and $R_1, \ldots, R_k, R'_1, \ldots, R'_k$ are second-order variables such that for each $1 \leq j \leq k$ there is a $n \in \mathbb{N}$ with $R_j, R'_j \in \mathcal{R}_n$ (same arity). Then $[\mathrm{TC}_{R_1,\ldots,R_k,R'_1,\ldots,R'_k,x_1,\ldots,x_l,x'_1,\ldots,x'_l}\ \varphi]$ is a third-order $2(k+l)$-ary relation expressing the existence of a φ-path between two $(R_1, \ldots, R_k, x_1, \ldots, x_l)$ tuples.

Second-order logics are certainly full logics in their own right (see the book by Shapiro (1991)). But they also have a strong connection to complexity theory. Actually, descriptive complexity theory was initiated by Fagin's (1975) result showing that ESO strongly captures NPTIME. 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.

3.7 Overview

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

[1] A finite structure is *ordered* iff its signature contains a binary relation that is interpreted as a linear order on the universe. A logic captures a complexity class iff the logic defines the same classes of finite *ordered* structures as the complexity class. A logic captures a complexity class *strongly* iff the logic defines the same classes of finite *arbitrary* structures as the complexity class. Any logic at least as expressive as ESO captures its respective complexity class strongly, because ESO is capable of expressing the existence of a linear order on the universe.

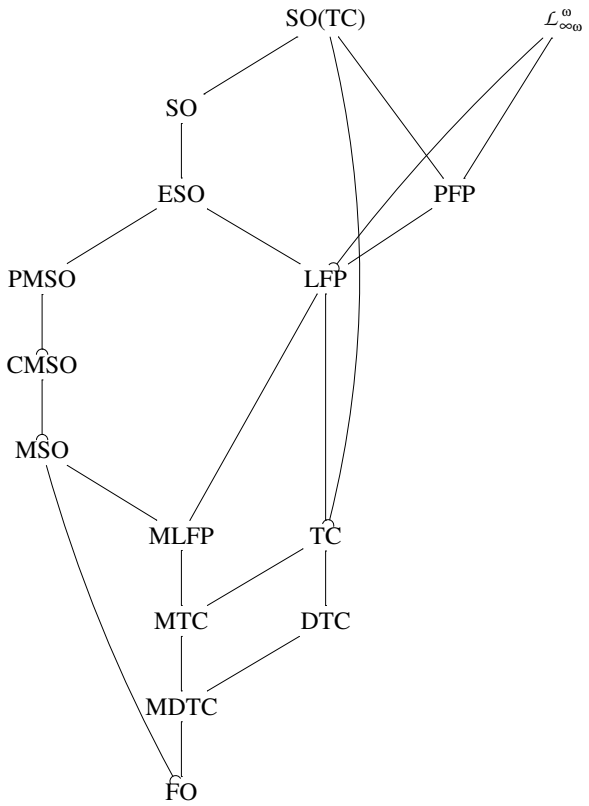


Figure 1: Logics for finite unordered unranked trees: the base $\supset\!\!-$ indicates a proper inclusion.

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, Chap. 7.5, p. 208).

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.

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

PROOF For the proper inclusion consider the tree language L_1 of trees where each leaf is at an even depth level, i.e, an even number of steps away from the root. Let

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

define the root of a tree. The formula

$$[\mathrm{DTC}_{x_3,x_1}\ \exists x_2.E(x_1,x_2) \wedge E(x_2,x_{3)}](y,x)$$

expresses that there is a path of an even number of steps from x to y. Now L_1 is defined by

$$\exists x.Root(x) \wedge \forall y.(Leaf(y) \rightarrow (y = x \vee [\mathrm{DTC}_{x_3,x_1}\ \exists x_2.E(x_1,x_2) \wedge E(x_2,x_{3)}](y,x))).$$

To show that L_1 is not FO-definable is a simple exercise using Ehrenfeucht-Fraïssé-games. For every quantifier depth k of an FO-formula potentially defining L_1 there are trees with maximal depth $2k$ and $2k+1$ such that the duplicator has a winning strategy. □

The second result concerns the expressive power of MSO and MLFP.

Theorem 13 *The logics* MSO *and* MLFP *have the same expressive power over unordered unranked trees.*

PROOF As stated above, MSO is capable of expressing monadic least fixed-points over arbitrary finite structures (see, e.g., (Ebbinghaus and Flum, 1995, Chap. 7.5, p. 208)).

The inverse direction is proven by capturing the run of a unary ordering PTA in MLFP. We use the fact that the equivalence of LFP and simultaneous LFP extends to MLFP, i.e., MLFP = $\mathrm{MLFP}^{\mathrm{simult}}$. This is proven, e.g., in (Libkin, 2004, Chap. 10.3).

Note that an atomic unary ordering constraint is equivalent to one of the four forms True, False, $y \leq n$ or $n \leq y$, where y is a Presburger variable and $n \in \mathbb{N}$, by simple arithmetics. Non-trivial atomic unary ordering constraints are expressible in FO. For $\delta(q,L) = y_p \leq n$ we have

$$(\exists x_1,\ldots,x_n : L(x) \wedge \bigwedge_{1\leq j\leq n} (E(x,x_j) \wedge x_j \in p) \wedge \forall z : z = x_1 \vee \cdots \vee z = x_n) \rightarrow x \in q$$

where x is a free variable for the root of the local tree. For $\delta(q,L) = n \leq y_p$ we have

$$(\exists x_1,\ldots,x_n : L(x) \wedge \bigwedge_{1\leq j\leq n} (E(x,x_j) \wedge x_j \in p) \wedge \bigwedge_{\substack{1\leq i,j\leq n \\ i<j}} x_i \neq x_j) \rightarrow x \in q$$

Unary ordering constraints are the closure of atomic unary ordering constraints under conjunction and negation. Hence unary ordering constraints are FO-expressible. The free variable x has to be universally quantified to complete the construction. Addionally we have to state that the set of states partitions the set of nodes in a tree. For any given fixed set of states $Q = \{q_1,\ldots,q_k\}$, this is also FO-expressible:

$$\forall x : \bigvee_{1\leq j\leq k} x \in q_j,$$

$$\forall x : \bigwedge_{1\leq j\leq k} (x \in q_j \rightarrow \bigwedge_{\substack{1\leq i\leq k \\ i\neq j}} x \notin q_i)$$

And we have to state the condition for an accepting run of the automaton, namely that the root of the tree is assigned a final state:

$$\exists x : \forall y\, \neg E(y,x) \wedge \bigvee_{q\in F} x \in q$$

The formula that expresses the run of the unary ordering PTA is the simultaneous monadic least fixed point over the set of states of the conjunction of the ordering constraints, the acceptance condition and the domain partitioning above. □

Note that for the case of *ordered ranked* trees, Potthoff (1994, p. 33f) showed an even stronger result. A (ordered ranked) tree language is regular iff it is definable in MLFP with a single (non-simultaneous) least fixed point operator. It is likely that this result can be extended to unordered unranked trees.

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.

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). The language is actually DTC^2-definable, i.e., definable in DTC where the transitive closures are taken over binary relations of pairs. It is a variation of a tree language

defined by Tiede and Kepser (2006). We have the following node labels f, g where f labelles the root, g is the label for all other nodes. The language is defined as $L_2 = \{f\{g^n, g^n\} \mid n \in \mathbb{N}^+\}$. It is the language of two g-chains of equal length below the root.

The language L_2 is definable in DTC as follows. Let

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

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_2. 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. We will show it is not even PMSO-definable.

Proposition 14 *The tree language L_2 is DTC-definable, but is* not 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.

PROOF Suppose $\mathcal{A} = (Q, \Lambda, \delta, F)$ is a tree automaton accepting L_2 and $k = |Q|$ is the number of states. Let $m > k$. Consider the tree $t = f\{g^m, g^m\} \in L_2$ 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 $q \in Q$ such that

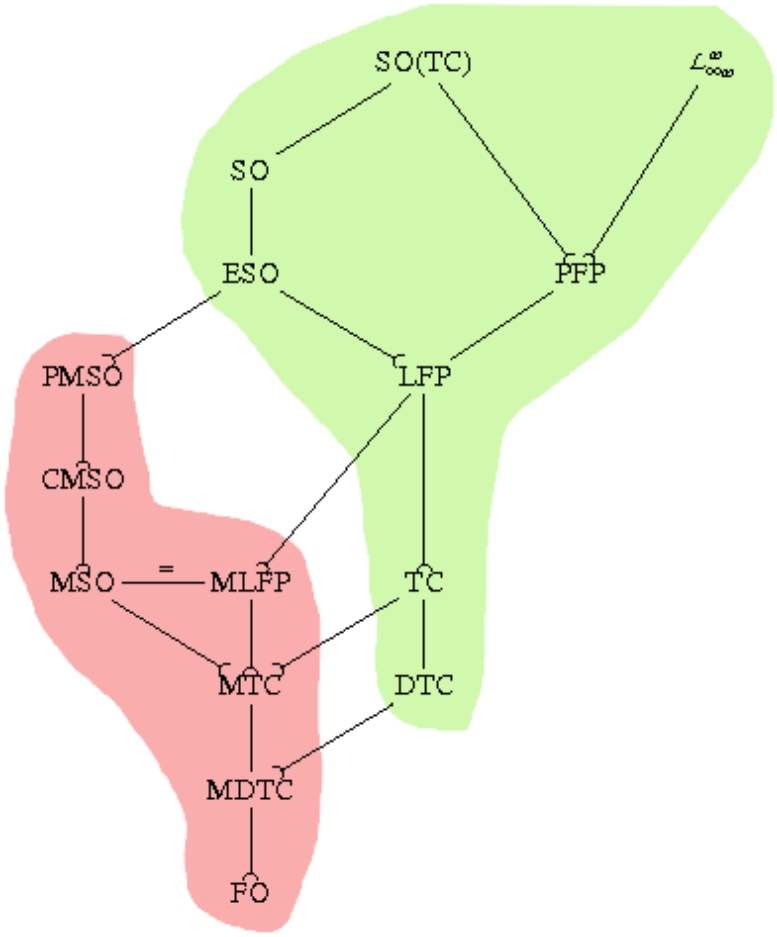


Figure 2: Separating automata logics from fixed-point logics.

$l_1 + l_2 + l_3 = m$ 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 $u \notin L_2$. □

The results of this subsection are depicted in Figure 2. Logics in the green area are capable of defining L_2, whereas logics in the red area are not.

Theorem 15 *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.*

5.2 An MSO-Definable Tree Language

Consider the following tree language. It is originally defined in (Ebbinghaus and Flum, 1995, Chap. 7.6.3) 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_3 and L_4 are the smallest sets such that

$$\begin{array}{rcl} 0 & \in & L_4 \\ 1 & \in & L_3 \\ BL' & \in & L_4 \text{ where } L' \subseteq_{mfin} L_3 \\ B(\{t\} \uplus L') & \in & L_3 \text{ where } t \in L_4 \text{ and } L' \subseteq_{mfin} L_3 \\ B(\{t,t'\} \uplus L' \uplus L'') & \in & L_4 \text{ where } t,t' \in L_4, L' \subseteq_{mfin} L_4, \text{ and } L'' \subseteq_{mfin} L_3. \end{array}$$

The tree language L_3 is recognised by the following Presburger tree automaton $\mathcal{A} = (\{q_t, q_f\}, \Lambda, \delta, \{q_t\})$ where

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

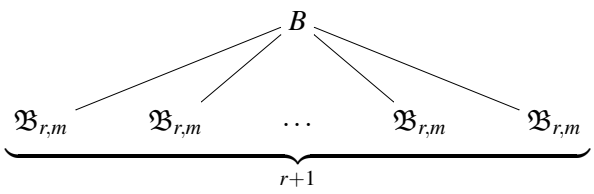
Hence L_3 is PMSO-definable. A close inspection of δ reveals that all constraints in the transitions are unary ordering constraints. Hence L_3 is even MSO-definable.

The proof that the tree language L_3 is not TC definable is an application of results by Grohe (1994). They are reprinted in the book by Ebbinghaus and Flum (1995, Chap. 7.6.3), which we will use as the source of our exposition.

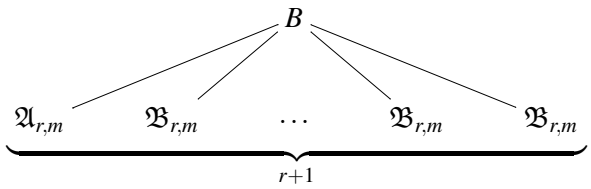
For natural numbers $r, m \in \mathbb{N}$ we define trees $\mathfrak{A}_{r,m}$ and $\mathfrak{B}_{r,m}$ inductively as follows.

$$\mathfrak{A}_{r,0}: \quad 0 \qquad\qquad \mathfrak{B}_{r,0}: \quad 1$$

Suppose $\mathfrak{A}_{r,m}$ and $\mathfrak{B}_{r,m}$ are already defined. Set $\mathfrak{A}_{r,m+1}$ to be



and $\mathfrak{B}_{r,m+1}$ to be



So $\mathfrak{A}_{r,m+1}$ contains $r+1$ copies of $\mathfrak{B}_{r,m}$ while $\mathfrak{B}_{r,m+1}$ contains one copy of $\mathfrak{A}_{r,m}$ and r copies of $\mathfrak{B}_{r,m}$. By a simple induction, it can be shown that for all $r,m \in \mathbb{N}$ the trees $\mathfrak{A}_{r,m} \in L_4$ and $\mathfrak{B}_{r,m} \in L_3$. Ebbinghaus and Flum (1995, p. 229ff) prove the following lemma. Define $P^r_k \subset \mathrm{FO(TC}^r)$ as the set of formulae of added quantifier and TC-operator nesting depth at most k and TC-operator width at most r.

Lemma 16 *Let $k \geq 0$ and φ be a* TC*-sentence in P^r_k. Then for $m > 2 \cdot k$,*

$$\mathfrak{A}_{2r,m} \models \varphi \qquad \textit{iff} \qquad \mathfrak{B}_{2r,m} \models \varphi.$$

This lemma can now be used to show that the tree language L_3 is not TC-definable.

Proposition 17 *The tree language L_3 is* MSO*-definable, but is* not TC*-definable.*

PROOF We showed above that L_3 is MSO-definable.
Suppose L_3 is TC-definable. Let φ be the TC-formula defining L_3. Then for all $k,m > 0$ we now $\varphi \models \mathfrak{B}_{r,m}$ because $\{\mathfrak{B}_{r,m} \mid r,m > 0\} \subset L_3$.
There are $k,r > 0$ such that $\varphi \in P^r_k$.
Then for all $m > 2k$ we have $\varphi \models \mathfrak{A}_{2r,m}$ by the above lemma.
But $\mathfrak{A}_{2r,m} \notin L_3$. □

The results of this subsection are depicted in Figure 3. Logics in the green area are capable of defining L_3, whereas logics in the red area are not.

Theorem 18 *The following inclusions are* strict*.*

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

Theorem 19 *The logics* (P)MSO *and* TC *are incomparable over the class of finite unordered unranked trees.*

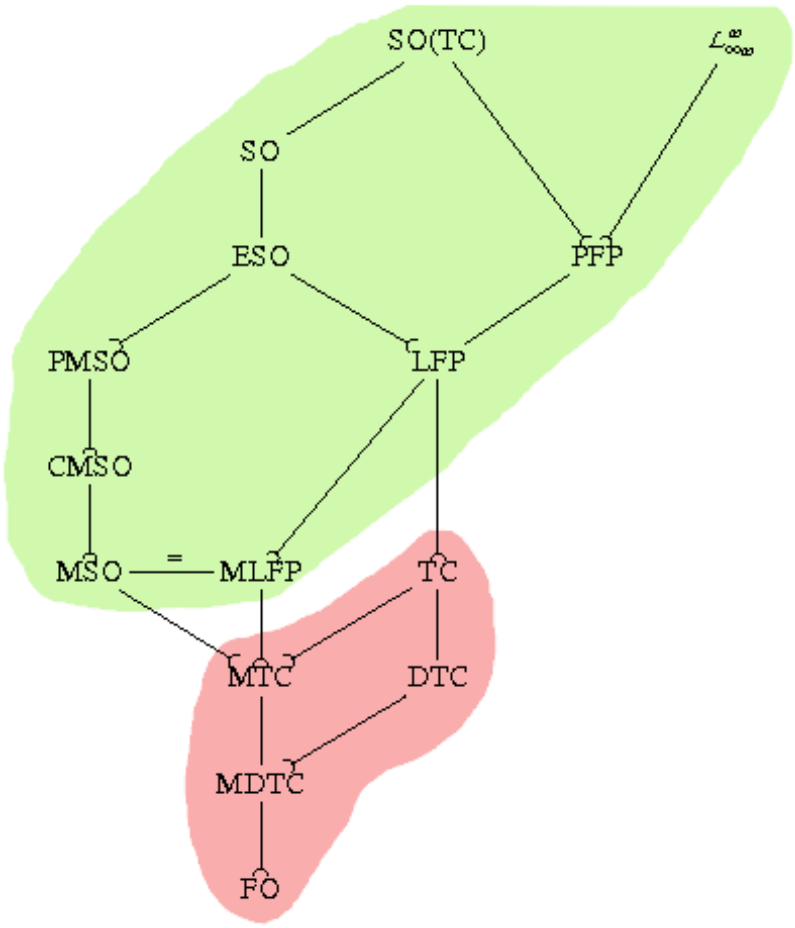


Figure 3: Separating automata logics from fixed-point logics.

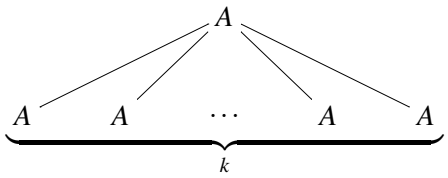
6 Separating Fixed-Point Logics and Second-Order Logics

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 particularly good at counting.

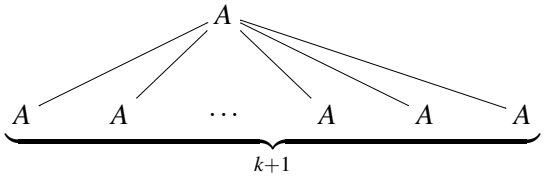
Let $\Lambda = \{A\}$. Define the tree language $L_5 = \{(V,E,\lambda) \mid |V| = 2n \text{ for some } n \in \mathbb{N}\}$ as the set of all tree with an even number of nodes (where each node is labelled with A). We first show that L_5 is CMSO-definable. The following formula defines L_5.

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

We will next show that L_5 is not $\mathcal{L}^{\omega}_{\infty\omega}$-definable using infinite pebble games. For the definition of this type of games, the reader is refered 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 20 *The duplicator has a winning strategy for the infinite pebble game* $\mathrm{PG}^{\infty}_k(\mathfrak{A}_k, \mathfrak{B}_k)$ *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 $(a_1,\dots,a_k) \mapsto (b_1,\dots,b_k)$ is a partial isomorphism.
Case 2: There is no pebble on the root of $\mathfrak{B}_k$.
Both $(b_1,\dots,b_k)$ and $(a_1,\dots,a_k)$ are discrete structures.
Case 2.1: The spoiler moves pebble j onto the root of $\mathfrak{B}_k$.
The induced structure of $(b_1,\dots,b_k)$ 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 $(a_1,\dots,a_k)$ is also $\mathfrak{A}_{k-1}$ and $(a_1,\dots,a_k) \mapsto (b_1,\dots,b_k)$ is a partial 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 21 *The language L_5 of trees with an even number of nodes is CMSO-definable, but is* not *$\mathcal{L}^\omega_{\infty\omega}$-definable.*

PROOF Suppose L_5 were $\mathcal{L}^\omega_{\infty\omega}$-definable, i.e, there were a formula $\varphi \in \mathcal{L}^\omega_{\infty\omega}$ that defined L_5. 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_5 whereas logics in the red one cannot.

Theorem 22 *The following inclusions are* strict.

- 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.

7 A Refinement Based on Complexity Classes

We showed that both PMSO and LFP are strictly weaker than ESO, and that they are incomparable. But the picture is actually a bit coarse, because ESO is already quite powerful. There seems to be no proper logic between ESO and PMSO or LFP. But one can provide a finer grained picture based on complexity classes.

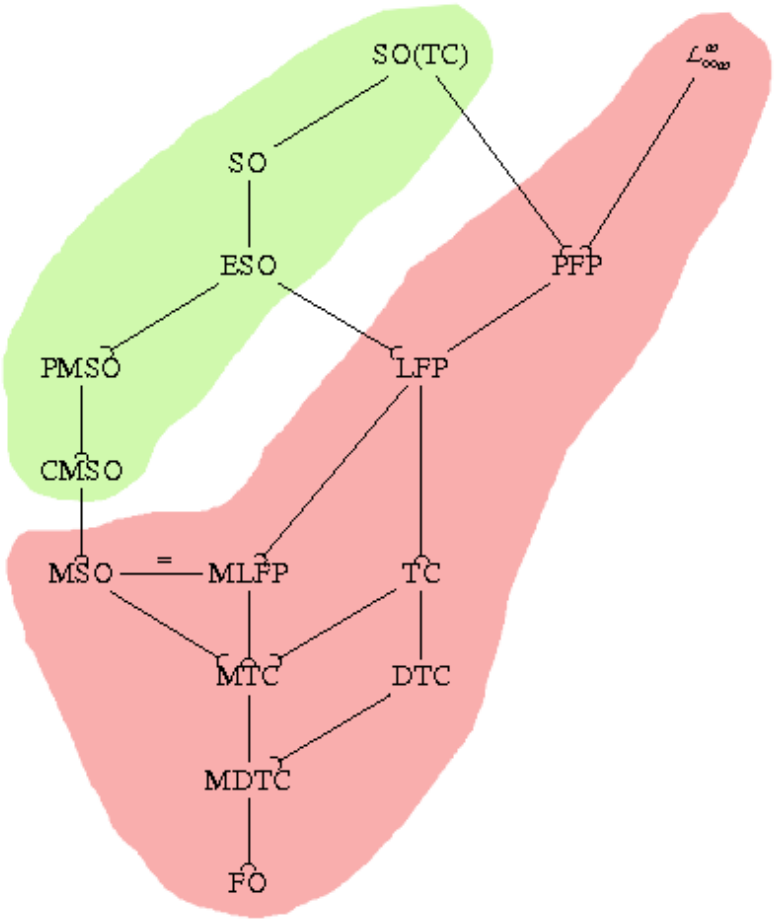


Figure 4: Separating fixed-point logics from second-order logics.

Seidl et al. (2003) show that PMSO is included in LinTIME. Now, reconsider tree language L_2 from Section 5.1. Checking that the two g-chains are of equal length can be done in linear time in the size of the input tree and requires just one counter. Therefore PMSO is strictly less powerful than LinTIME.

A known result of descriptive complexity is that LFP is included in PTIME over arbitrary finite structures. For the case of unordered unranked trees this inclusion is proper. This is demonstrated by tree language L_5 from Section 6. The language L_5 is *not* LFP-definable. But since it is CMSO-definable it is also PTIME-computabe.

The results of this section are summarised in Figure 5.

Theorem 23 *The following inclusions are* strict.

- PMSO *is* strictly *less powerful than* LinTIME.

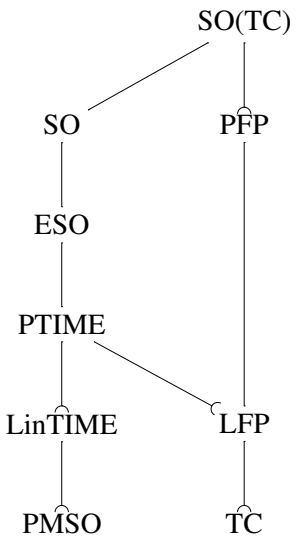


Figure 5: A closer look at PMSO, LFP, and ESO based on complexity classes. $\supset\!\!-$ indicates a proper inclusion.

- LFP *is* strictly *less powerful than* PTIME.

Note that LFP being strictly less powerful than PTIME was already shown by Lindell (1991). Our proof though is a lot simpler.

8 Separating PFP from Infinitary Logic

In this section, we consider the relationship between PFP and infinitary logic. As the first step, we show that PFP is strictly less powerful than $\mathcal{L}^{\omega}_{\infty\omega}$ on unordered unranked trees. The proof uses a method devised by Kolaitis and Vardi (1992) for separating PFP from $\mathcal{L}^{\omega}_{\infty\omega}$ on finite orderings.

Theorem 24 *The logic* PFP *is strictly less powerful than* $\mathcal{L}^{\omega}_{\infty\omega}$ *on unordered unranked trees.*

PROOF It can be expressed in FO that a tree is non-branching:
$\forall x.Leaf(x) \vee (\exists y.E(x,y) \wedge \forall z.E(x,z) \rightarrow y = z)$.
If a tree is non-branching, then for every $n \in \mathbb{N}$ one can state that the tree has n nodes using just 3 variables. For example, for $n = 4$ the formula is
$\exists x,y.Root(x) \wedge E(x,y) \wedge (\exists x.E(y,x) \wedge (\exists y.E(x,y) \wedge Leaf(y)))$.
The proof of this fact can be found in (Immerman and Kozen, 1989). So, for every

$n \in \mathbb{N}$ let φ_n be the $\mathcal{L}^{\omega}_{\infty\omega}$-formula stating that the tree is non-branching and has exactly n nodes. Now, let M be a non-recursive set of natural numbers. The following $\mathcal{L}^{\omega}_{\infty\omega}$-formula

$$\bigvee_{n \in M} \varphi_n$$

is a non-recursive property. It is true on a tree iff it is non-branching and its number of nodes is an element from M.

On the other hand, since PFP $\subseteq$ PSPACE, every PFP-definable tree language is recursive. □

Theorem 25 *The logics* $\mathcal{L}^{\omega}_{\infty\omega}$ *and* SO(TC) *are incomparable over unordered unranked trees.*

PROOF The proof of the theorem above provides a tree language that is not SO(TC)-definable. On the other hand, we saw in Section 6 that tree language L_5 is SO(TC)-definable, but not $\mathcal{L}^{\omega}_{\infty\omega}$-definable. □

We saw that PFP $\subsetneq$ SO(TC) and PFP $\subsetneq \mathcal{L}^{\omega}_{\infty\omega}$ by different tree languages. One may now ask whether PFP can define all $\mathcal{L}^{\omega}_{\infty\omega}$-definable tree languages that are computable in PSPACE. It turns out that there are $\mathcal{L}^{\omega}_{\infty\omega}$-definable PSPACE computable tree languages that cannot be defined in PFP.

Theorem 26 *The following inclusions are* strict.

- PFP *is* strictly *less powerful than* $\mathcal{L}^{\omega}_{\infty\omega} \cap$ PSPACE.
- $\mathcal{L}^{\omega}_{\infty\omega} \cap$ PSPACE *is* strictly *less powerful than* PSPACE.
- $\mathcal{L}^{\omega}_{\infty\omega} \cap$ PSPACE *is* strictly *less powerful than* $\mathcal{L}^{\omega}_{\infty\omega}$.

PROOF It is the first item that is to be shown here. The two others follow from the proofs of Proposition 21 and Theorem 24.

The following proof is an extension of the proof by Dawar et al. (1995), who show that LFP $\subsetneq$ PTIME $\cap\, \mathcal{L}^{\omega}_{\infty\omega}$. We follow their exposition closely quoting it frequently, because their proofs remain true if one replaces LFP with PFP throughout.

A binary tree is a tree in which each internal node has exactly two children. A level is a set of nodes with equal distance to the root. A binary tree is complete, if all leaves are in one level. Let the set of labels $\Lambda = \{0,1\}$. Define the class $\mathcal{T}$ of complete binary Λ-trees as those complete binary trees where for each level all nodes in this level have the same label. In other words, if two nodes have the same distance to the root then both are labelled with 0 or both are labelled with 1.

Fact 1: Every language $L \subseteq \mathcal{T}$ of complete binary Λ-trees is $\mathcal{L}^{\omega}_{\infty\omega}$-definable.

A binary string is a Λ-labelled tree where each node has at most one child. We denote the set of all binary strings by $\mathcal{B}$. Every set of binary strings is also $\mathcal{L}^{\omega}_{\infty\omega}$-definable. Actually, a complete binary Λ-labelled tree encodes the same information as a binary string, because all nodes on the same level carry the same label, a 0 or a 1. For a string b and a tree $t \in \mathcal{T}$ we write $b \sim t$ iff they encode the same information. This is expressed in the following map. For every $B \subseteq \mathcal{B}$ and $T \subseteq \mathcal{T}$ define $h(B) = \{t \in \mathcal{T} \mid \exists b \in B \text{ with } b \sim t\}$ and $h^{-1}(T) = \{b \in \mathcal{B} \mid \exists t \in T \text{ with } b \sim t\}$. Obviously $h^{-1}(h(B)) = B$.

Claim 1: If $B \in$ EXPSPACE then $h(B) \in$ PSPACE.

Given a tree $t \in h(B)$ we can extract from it a string b such that $b \sim t$ in DLOGSPACE($|t|$). String b can then be checked for acceptance in EXPSPACE($|b|$) = EXPSPACE(DLOGSPACE($|t|$)) = PSPACE($|t|$).

Claim 2: If T is PFP-definable, then $h^{-1}(T)$ is PFP-definable.

Lindell (1991) proves that if T is LFP-definable then $h^{-1}(T)$ is LFP-definable. The main part of this proof is given in Lemma 3.2 and Corollary 3.3 by means of a translation of LFP-formulae over complete binary trees into LFP-formulae over binary strings. This translation is semantics-preserving for least fixed-points, because each stage of the fixed-point construction is a so-called invariant relation. This fact extends to partial fixed-points. If no fixed point exists for a formula, then the PFP-operator yields the empty relation. And the empty relation is an invariant relation. For details, see (Lindell, 1991).[2]

Claim 3: There exists a tree language $T \subseteq \mathcal{T}$ such that $T \in$ PSPACE, but T is not PFP-definable.

By the space hierarchy theorem there are string languages in EXPSPACE which are not in PSPACE. Let B be such a string language. Then $h(B) \in$ PSPACE (claim 1). Suppose $h(B)$ were PFP-definable. Then B were PFP-definable by Claim 2. Then B were in PSPACE, contradicting the assumption that B not in PSPACE. Thus $h(B)$ is in PSPACE, but not PFP-definable.

Moreover $h(B)$ is $\mathcal{L}^{\omega}_{\infty\omega}$-definable by Fact 1. Hence $h(B) \in \mathcal{L}^{\omega}_{\infty\omega} \cap$ PSPACE, but not PFP-definable. □

9 Conclusion

Figure 6 depicts a landscape of the expressive power of different logics for finite unordered unranked trees. As one can see, most inclusions turn out to be proper. The following ones have been shown in this paper to be proper.

[2]The proof of Claim 2, in particular the insight that Lindell's translation of of LFP-formulae over complete binary trees into LFP-formulae over binary strings extends to PFP, is the main new contribution.

- $\mathcal{L}^{\omega}_{\infty\omega} \cap \text{PSPACE} \subsetneq \text{SO(TC)}$,
- $\mathcal{L}^{\omega}_{\infty\omega} \cap \text{PSPACE} \subsetneq \mathcal{L}^{\omega}_{\infty\omega}$,
- $\text{PFP} \subsetneq \text{PSPACE} \cap \mathcal{L}^{\omega}_{\infty\omega}$,
- $\text{LFP} \subsetneq \text{ESO}$, $\text{LFP} \subsetneq \text{PTIME}$,
- $\text{PMSO} \subsetneq \text{ESO}$, $\text{PMSO} \subsetneq \text{LinTIME}$,
- $\text{MSO} \subsetneq \text{CMSO}$,
- $\text{MLFP} \subsetneq \text{LFP}$,
- $\text{TC} \subsetneq \text{LFP}$,
- $\text{MTC} \subsetneq \text{MSO}$,
- $\text{MTC} \subsetneq \text{TC}$,
- $\text{MDTC} \subsetneq \text{DTC}$, and
- $\text{FO} \subsetneq \text{MDTC}$.

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 PFP amounts to the separation of PTIME from PSPACE by the Abiteboul-Vianu theorem (Abiteboul and Vianu, 1995). This makes refinement a challenging task.

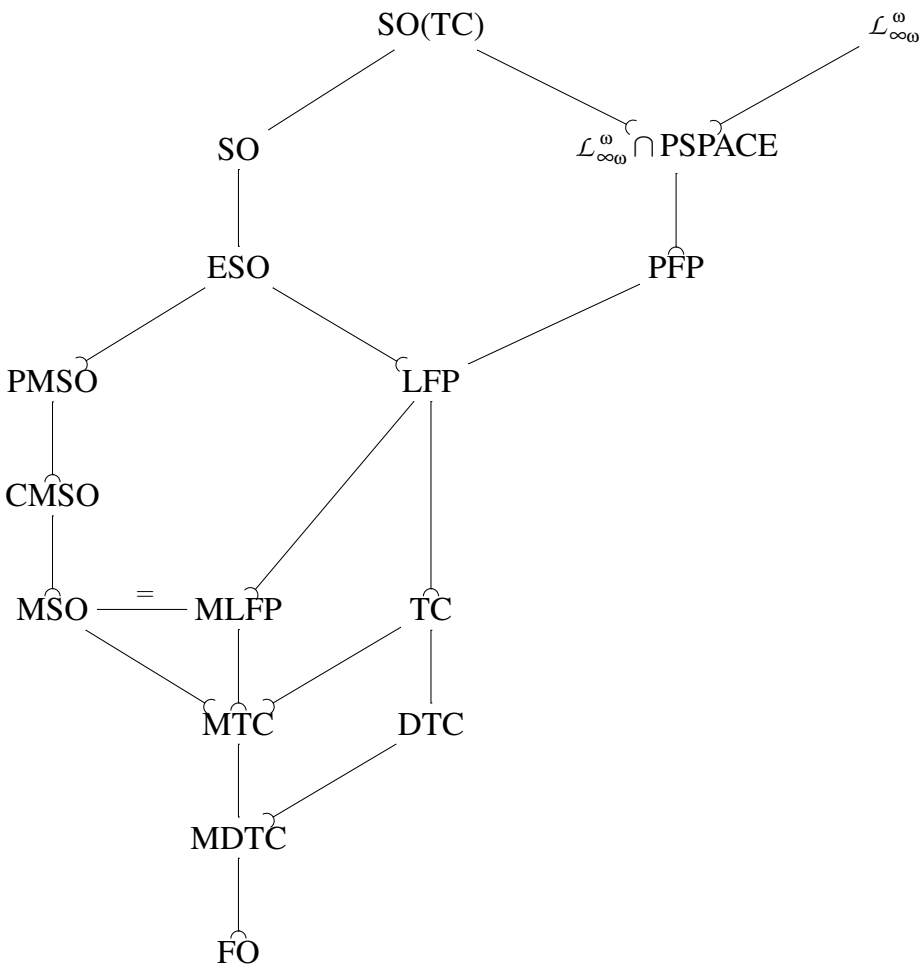


Figure 6: Logics for finite unordered unranked trees.
$\supset\!\!-$ indicates a proper inclusion.

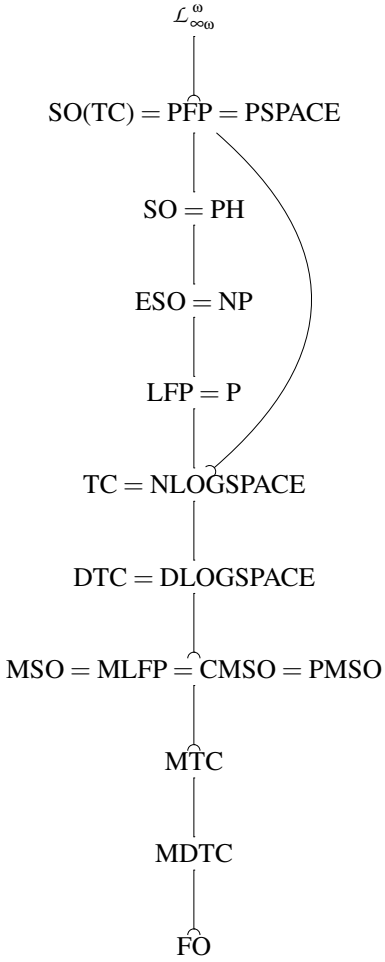


Figure 7: Logics for finite ordered ranked trees.
$\supset$— indicates a proper inclusion.

10 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 questions in classical complexity theory. In all logics at least as powerful as MDTC – that is in all logics considered in this paper with the exception of FO – it is possible to define a linear order on the nodes of a tree (see (Kepser, 2006)). Hence in these logics, ordered trees are ordered structures in the sense of descriptive complexity theory. This constitutes the relationship to classical complexity theory. Figure 7 summarises the results on the expressive power of different logics over ordered ranked trees.

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. And Bojanczyk et al. (2006) showed that MDTC $\subsetneq$ MSO. Recently, ten Cate and Segoufin (2008) were able to show that 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.

Bojanczyk, Mikołaj, Mathias Samuelides, Thomas Schwentick, and Luc Segoufin (2006). Expressive power of pebble automata. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, eds., *Automata, Languages and Programming, ICALP 2006*, volume I, pp. 157–168. Springer.

Boneva, Iovka and Jean-Marc Talbot (2005). Automata and logics over unranked and unordered trees. In Jürgen Giesl, ed., *Proceedings RTA 2005*, LNCS 3467, pp. 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*.

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

Immerman, Neil and Dexter Kozen (1989). Definability with bounded number of bound variables. *Information and Computation*, **83**:121–139.

Kepser, Stephan (2006). Properties of binary transitive closure logic over trees. In Paola Monachesi, Gerald Penn, Giorgio Satta, and Shuly Wintner, eds., *Formal Grammar 2006*, pp. 77–89.

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.

Lindell, Steven (1991). An analysis of fixed-point queries on binary trees. *Theoretical Computer Science*, **85**:75–95.

Potthoff, Andreas (1994). *Logische Klassifizierung regulärer Baumsprachen*. Ph.D. thesis, Institut für Informatik, Universität Kiel.

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

Shapiro, Stewart (1991). *Foundations without Foundationalism: A Case for Second-Order Logic*. Oxford University Press.

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 Grigori Mints, ed., *13th Workshop on Logic, Language, Information and Computation*, ENTCS 165, pp. 189–199. Springer.