ELSEVIER

Theoretical Computer Science 275 (2002) 311–346

Theoretical Computer Science

www.elsevier.com/locate/tcs

Monadic second-order logic on tree-like structures

Igor Walukiewicz

Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland

Received September 1999; received in revised form February 2001; accepted March 2001
Communicated by W. Thomas

Abstract

An operation M^* which constructs from a given structure M a *tree-like structure* whose domain consists of the finite sequences of elements of M is considered. A notion of automata running on such tree-like structures is defined. It is shown that automata of this kind characterise expressive power of monadic second-order logic (MSOL) over tree-like structures. Using this characterisation it is proved that MSOL theory of a tree-like structure is effectively reducible to that of the original structure. As another application of the characterisation it is shown that MSOL on trees of arbitrary degree is equivalent to first-order logic extended with unary least fixpoint operator.

Keywords: Monadic second-order logic; Tree automata; Decidability

1. Introduction

In [19] Shelah mentions the following construction. For a structure $M = \langle D_M, r_1, \ldots\rangle$ one considers the structure $M^\sharp = \langle D_M^*, son, r_1^*, \ldots\rangle$, where D_M^* is the set of all finite sequences of elements of D_M; relation $son(w, wd)$ holds for every $w \in D_M^*$ and $d \in D_M$; finally $r^*(wd_1, \ldots, wd_k)$ holds iff $r(d_1, \ldots, d_k)$ holds in the original structure. He refers to an unpublished paper of Stupp [21] which contains the proof of the fact that the MSO theory of $M^\sharp$ is decidable whenever the MSO theory of M is decidable.

Semenov [18] presents an extension of this result which he attributes to Muchnik. The stronger operation M^* is considered which creates a structure $M^* = \langle D_M^*, son, cl, r_1^*, \ldots\rangle$. The new clone relation, cl, is an unary relation that holds for all elements of the

E-mail address: igw@mimuw.edu.pl (I. Walukiewicz).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00185-2

form wdd where $w \in D_M^*$ and $d \in D_M$. The statement of the theorem is also strengthened. It says that for every sentence φ of MSOL over the extended signature one can effectively find an MSOL sentence $\hat{\varphi}$ over the original signature such that for every structure M of the original signature:

$$M \models \hat{\varphi} \text{ iff } M^* \models \varphi.$$

Semenov hints an idea of the proof but apparently the proof was never published.

In this paper we characterize the power of MSOL on tree-like structures by means of automata. We use this characterisation to give a proof of Muchnik's theorem. We also show that MSOL over unordered trees is equivalent to first-order logic extended with unary fixpoint operator (FPL).

One of the applications of Muchnik's theorem is proving decidability of MSO theories. For example, as the MSO theory of any finite structure is decidable it immediately follows that MSO theory of a sequence or a tree of such structures is decidable. In particular, one gets Rabin's Theorem [16] when one starts from a two element structure. A different application is presented in [4]; it gives a relation between the theory of a structure and the theory of the unwinding of the structure. A general notion of automaton that is used in this work is also useful in other contexts. Automata for the mu-calculus [9], an instance of general automata, were the main tool to show an uniform interpolation theorem for the mu-calculus [5]. The general automata were used to show the equivalence between the bisimulation invariant fragment of MSOL and the mu-calculus [10]. The equivalence of MSOL and FPL over trees can be seen as a similar kind of result but for unwinding instead of bisimulation. More precisely, it implies that a property closed under unwinding is definable in MSOL iff it is definable in FPL. Of course, both MSOL and FPL can define properties which are not closed under unwinding.

The proof presented here does not follow the outline given by Semenov although most certainly profits from his exposition. A natural way to prove Muchnik's theorem is to extend some proof of Rabin's theorem. Stupp modified the original proof of Rabin. Muchnik took as a starting point his proof of Rabin's theorem. Here we take the approach through fixpoint operators and parity automata that was discovered by Emerson and Jutla [8]. Parity automata allow considerable simplifications of the proof of Rabin's theorem. The fixpoint description of the behaviour of an automaton is essential for our proofs of the results presented here.

Let us now give an overview of the paper. After a preliminary section introducing MSOL we give a proof of the forgetful determinacy theorem for parity games. The proof gives a fixpoint operator describing the set of winning positions. In Section 3 we introduce a general notion of (alternating) automaton running on tree-like structures. In full generality these automata are much too powerful. The idea is to limit their expressive power by restricting the class of allowed transition functions. The goal of Section 3 is to show what closure properties of the class of transition functions are sufficient for the automata to be closed under sum, complementation and projection. In Section 4 we take a particular class of transition functions, namely those definable

by MSOL formulas. We show that this restricted class satisfies the closure conditions from Section 3. This gives us the automata characterisation of MSOL on tree-like structures. In Section 5 we use this characterisation and the fixpoint description of the set of winning positions in parity games to show Muchnik's theorem. In particular, the fixpoint characterisation allows us to code a run of automaton on M^* directly into M. In Section 6 we further limit the class of transition functions of our automata to those definable by some simple first-order formulas. This restriction still leaves enough power for automata to capture MSOL on trees. We get the equivalence of MSOL and FPL by coding runs of these automata in FPL.

2. Preliminaries

We will use $\mathbb{N}$ for the set of natural numbers. For a number $n \in \mathbb{N}$ we write $[n]$ for the interval $\{1,\ldots,n\}$. Let $Sig = \{r_1,\ldots\}$ be a signature containing relational symbols only. Let $Var = \{X, Y, Z, \ldots\}$ be a set of second-order variables. The set of *MSOL formulas over Sig* is the smallest set such that:

- $X \subseteq Y$ is a formula for every $X, Y \in Var$;
- $r(X_1,\ldots,X_k)$ is a formula for every $r \in Sig$ of arity k and every $X_1,\ldots,X_k \in Var$;
- if α, β are formulas then $\neg\alpha$, $\alpha \vee \beta$, $\exists X\,.\,\alpha$ are formulas.

Formulas of the first two kinds are called *atomic*.

In a model $M = \langle D_M, r_1^M, \ldots\rangle$ each relation symbol of arity k is interpreted as a k-ary relation on D_M, i.e., a subset of $(D_M)^k$. The variables from Var range over subsets of D_M. So a valuation is a function $Val : Var \to \mathcal{P}(D_M)$. The satisfiability of a formula α in a model M with a valuation Val (denoted $M, Val \models \alpha$) is defined by induction on the structure of the formula:

- $M, Val \models X \subseteq Y$ iff $Val(X) \subseteq Val(Y)$;
- $M, Val \models r(X_1,\ldots,X_k)$ iff there exist $d_1 \in Val(X_1),\ldots,d_k \in Val(X_k)$ such that $r^M(d_1,\ldots,d_k)$ holds;
- $M, Val \models \neg\alpha$ if not $M, Val \models \alpha$;
- $M, Val \models \alpha \vee \beta$ if $M, Val \models \alpha$ or $M, Val \models \beta$;
- $M, Val \models \exists X.\alpha$ if there is a set $S \subseteq D_M$ such that $M, Val[S/X] \models \alpha$; where $Val[S/X]$ denotes the valuation identical to Val on all the variables except X for which $Val[S/X]$ returns S.

Please observe that we allow slightly nonstandard formulas of the form $r(X_1,\ldots,X_k)$ where the variables are second-order variables. There are several reasonable ways to interpret such formulas. Our choice will simplify some automata constructions later on.

Many useful predicates are definable in our language. The predicate $X = \emptyset$ saying that the meaning of X is the empty set can be defined by $\forall Y.\ X \subseteq Y$. To say that the meanings of variables X and Y are the same $(X = Y)$ we can write $X \subseteq Y \wedge Y \subseteq X$. The predicate $Singl(X)$ saying that the meaning of X is a singleton can be defined by $\neg(X = \emptyset) \wedge \forall Y.\ Y \subseteq X \Rightarrow (Y = X \vee Y = \emptyset)$. This predicate allows us to make some second-order variables to behave like first-order ones by restricting their range to

singleton sets. In particular, the meaning of $r(X_1,\ldots,X_k)$ when $Val(X_1),\ldots,Val(X_k)$ are singletons is just the standard meaning from first-order logic.

By the remarks above we can simulate first-order variables by second-order variables ranging over singletons. We will use lower case letters for such variables. We will use $\exists x.\alpha$ as an abbreviation for $\exists x.Singl(x)\wedge\alpha$. Similarly $\forall x.\alpha$ stands for $\forall x.Singl(x)\Rightarrow\alpha$.

Next we present some syntax for defining sets of elements of a structure. In general, given a structure M and valuation Val, we say that a set S is *definable* by a formula $\varphi(x)$ iff $S=\{d\in D_M\colon\ M, Val[\{d\}/x]\models\varphi(x)\}$; here x is some distinguished variable. In order to make this distinguished variable explicit we will write

$$\{x:\varphi(x)\}$$

Such an expression is called *MSOL predicate*. The intended meaning of such a predicate is

$$\|\{x:\varphi(x)\}\|_M^{Val}=\{d\in D_M\colon\ M, Val[\{d\}/x]\models\varphi(x)]\}.$$

Finally, we introduce the notation for fixpoints of predicates. Suppose we have MSOL predicate θ. We want to define the meanings of predicates $\mathsf{LFP}\ Z.\theta$ and $\mathsf{GFP}\ Z.\theta$. For a given M and Val the predicate θ defines a set $\|\theta\|_M^{Val}$. Changing valuation of Z may change the set defined by θ. Hence we get a function from $\mathscr{P}(D_M)$ to $\mathscr{P}(D_M)$ given by $S\mapsto\|\theta\|_M^{Val[S/Z]}$. The meaning of the predicate $\mathsf{LFP}\ Z.\theta$ will be the least fixpoint of this function or the empty set if there is no such fixpoint. Similarly, the meaning of $\mathsf{GFP}\ Z.\theta$ will be the greatest fixpoint or the empty set if it does not exist.

Example. Let $M=\langle D,E\rangle$ be a graph with vertex set D and the edge relation E. Let $\theta=\{x:\exists y.E(x,y)\wedge(y\in Y\vee y\in Z)\}$. For a given valuation Val the meaning of θ is the set of all the elements $d\in D$ from which there is an edge to $Val(Y)\cup Val(Z)$. The meaning of the predicate $\mathsf{LFP}\ Z.\theta=\mathsf{LFP}\ Z.\{x:\exists y.E(x,y)\wedge(y\in Y\vee y\in Z)\}$ is the set of all d from which there is a path to an element in $Val(Y)$.

Remark. Fixpoint predicates are definable in our basic MSOL language. Hence adding them to the syntax does not increase the expressive power of the language.

3. Forgetful determinacy for parity conditions

In this section we define parity games and give an explicit description of winning strategies in such games. We describe the set of winning positions by a fixpoint expression and derive a winning strategy from this expression using the concept of signatures. We will extensively use this fixpoint characterisation in later sections.

The notion of signature was proposed by Büchi [2] and independently by Streett and Emerson [20]. The proof of the existence of memoryless strategies in parity games was given independently by Emerson and Jutla [8] and by Mostowski [15]. Klarlund

[12] proves a more general fact that a player has a memoryless winning strategy in a game if he has a winning strategy and his winning condition is given as a Rabin condition. In [13, 22, 23] one can find beautiful presentations of the theorem together with discussions on its applications. The approach here is based on [8]. It is more technical but gives us a fixpoint characterisation of the winning set.

A *parity game* is a tuple $G = \langle V = V_0 \cup V_1, E \subseteq V \times V, \Omega : V \to [n] \rangle$ where $V_0 \cap V_1 = \emptyset$. Function Ω assigns to each vertex a natural number called *priority* of the vertex. We say that v' is a *successor* of v if $(v, v') \in E$.

A play in the game from some vertex $v_0 \in V_0$ proceeds as follows: first player 0 chooses a successor v_1 of v_0, then player 1 chooses a successor v_2 of v_1, and so on ad infinitum unless one of the players cannot make a move. If a player cannot make a move he loses. The result of an infinite play is an infinite sequence of vertices $v_0, v_1, v_2, \ldots$ Using function Ω we obtain a sequence of natural numbers $\Omega(v_0), \Omega(v_1), \Omega(v_2) \ldots$ Player 0 wins the game iff this sequence satisfies the *parity condition*, i.e., the smallest among numbers appearing infinitely often in the sequence is even. The play from vertices of V_1 is defined similarly but this time player 1 starts.

A *strategy* σ for player 0 is a function assigning to every finite sequence of vertices $\vec{v}$ ending in a vertex $v \in V_0$ one of its successors $\sigma(\vec{v}) \in V$. A strategy is called *memoryless* iff $\sigma(\vec{v}) = \sigma(\vec{v}')$ whenever $\vec{v}$ and $\vec{v}'$ end in the same vertex. The strategy is *winning* iff it guarantees a win for player 0 whenever he follows the strategy. Similarly, we define strategies and winning strategies for player 1. Our main goal is to show that from every node of a parity game one of the players has a winning memoryless strategy.

For the rest of this section let us fix a game graph:

$$G = \langle V = V_0 \cup V_1, E, \Omega : V \to [n] \rangle$$

To avoid considering two cases we assume that the above number n is even. Clearly we can do so without a loss of generality. The graph G can be represented as a relational structure:

$$M(G) = \langle V, E, V_0, V_1, P_1, \ldots, P_n \rangle,$$

where V is the carrier, E defines an edge relation between states, and $V_0, V_1, P_1, \ldots, P_n$ are subsets of V. Each P_i (for $i = 1, \ldots, n$) denotes the set of nodes with priority i, i.e., the set $\{v\colon\ \Omega(v) = i\}$.

Consider the predicate

$$F_0(Z_1, \ldots, Z_n) = \left\{ x : \left(V_0(x) \Rightarrow (\exists y.E(x,y) \wedge \bigwedge_{i \in [n]} (P_i(y) \Rightarrow Z_i(y))) \right) \right.$$
$$\left. \wedge \left(V_1(x) \Rightarrow \forall y.\ (E(x,y) \Rightarrow \bigwedge_{i \in [n]} (P_i(y) \Rightarrow Z_i(y))) \right) \right\}. \qquad (1)$$

We will show that the set of winning positions for player 0 in game G is precisely the set

$$W_0 = \|\mathsf{LFP}\ Z_1.\mathsf{GFP}\ Z_2.\ldots.\mathsf{LFP}\ Z_{n-1}.\mathsf{GFP}\ Z_n.F_0(Z_1,\ldots,Z_n)\|^{M(G)} \tag{2}$$

(in the above formula LFP is used to close variables with odd indices and GFP is used for even indices; n is even by our assumption).

To understand some intuitions behind this formula consider the formula $\mathsf{LFP}\ Z_n.F_0(Z_1,\ldots,Z_n)$. This formula holds in a node of the structure $M(G)$ if from this node player 0 can force the play in a finite number of steps into a node of a priority $i<n$ in which Z_i holds. Similarly, the greatest fixpoint formula $\mathsf{GFP}\ Z_n.F_0(Z_1,\ldots,Z_n)$ describes that player 0 can either stay forever in nodes of priority n or he can reach a node of a priority $i<n$ in which Z_i holds. Hence choosing appropriate fixpoint we can decide whether we want to force a play to reach some smaller priority or whether we can also allow the play to stay in nodes of a given priority infinitely. As we show below, by alternating the fixpoints the way we have done in the formula we can express the parity condition.

Theorem 1 (Forgetful determinacy). *From every node of W_0 there is a winning memoryless strategy for player* 0. *From every node in $V\setminus W_0$ player* 1 *has a winning memoryless strategy.*

The idea of the proof is the following. For every vertex in W_0 we associate a *signature* which is a n-tuple of ordinals. Intuitively, a signature says how far is the vertex from something good. We use signatures to define a winning memoryless strategy for player 0 from vertices in W_0. Finally, it turns out that the complement of W_0 is defined by a formula of exactly the same shape as the one defining W_0. This gives us a memoryless winning strategy for player 1 from vertices not in W_0.

Definition 2. When applied to n-tuples of ordinals, symbols $=, <, \leqslant$ stand for corresponding relations in the lexicographical ordering (i.e., $(a_1,\ldots,a_n) < (b_1,\ldots,b_n)$ if for the smallest position i on which the elements are different we have $a_i<b_i$). For every $i\in\{1,\ldots,n\}$ we use $=_i$ to mean that both arguments are defined and when truncated to the first i positions the two tuples are equal; similarly for $<_i$ and $\leqslant_i$.

Definition 3 (*Consistent signature assignment*). A signature is a n-tuple of ordinals. Let $\mathcal{S}$ be a partial function assigning signatures to nodes of the game. Let $U\subseteq V$ be the domain of $\mathcal{S}$. For a node $v\in V$ and its successor w we say that w is *good* for v if $w\in U$ and

$$\mathcal{S}(w) \leqslant_{\Omega(w)} \mathcal{S}(v) \text{ and the inequality is strict if } \Omega(w) \text{ is odd.} \tag{3}$$

The assignment $\mathcal{S}$ is *consistent* if for every $v\in U\cap V_0$ there is a good successor and for every $v\in U\cap V_1$ every successor of v is good.

When defining MSOL we have introduced means for defining predicates and fixpoints of predicates. For the next definition we need to extend the syntax with approximations of fixpoint predicates. These approximations have the form $\mathsf{LFP}^{\tau}\ Z.\alpha(Z)$, where τ is an ordinal and $\alpha(Z)$ is a formula possibly also containing this new constructor. The semantics is defined as follows:

$$\|\mathsf{LFP}^{0}\ Z.\alpha(Z)\|^{M}_{Val} = \emptyset, \qquad \|\mathsf{LFP}^{\tau+1}\ Z.\alpha(Z)\|^{M}_{Val} = \|\alpha(Z)\|^{\mathscr{M}}_{V[\|\mathsf{LFP}^{\tau}\|Z.\alpha(Z)\|^{M}_{Val}/Z]},$$
$$\mathsf{LFP}^{\tau}\ Z.\alpha(Z)\|^{M}_{Val} = \bigcup_{\rho<\tau} \|\mathsf{LFP}^{\rho}\ Z.\alpha(Z)\|^{M}_{Val} \quad \text{(for } \tau \text{ a limit ordinal)}.$$

By the Knaster–Tarski theorem $\|\mathsf{LFP}\ Z.\alpha(Z)\|^{M}_{Val} = \bigcup_{\tau} \|\mathsf{LFP}^{\tau}\ Z.\alpha(Z)\|^{M}_{Val}$. Observe that such approximations are in general not definable in our basic MSOL language.

Definition 4 (*Canonical signatures*). A *canonical signature*, $Sig(v)$, of a vertex $v \in V$ is the smallest in the lexicographical ordering sequence of ordinals $(\tau_1, \ldots, \tau_n)$ such that

$$v \in \|F_0(U_1, \ldots, U_n)\|^{M(G)},$$

where for odd i,

$$U_i = \mathsf{LFP}^{\tau_i}\ Z_i.\mathsf{GFP}\ Z_{i+1}.\mathsf{LFP}\ Z_{i+2} \ldots \mathsf{GFP}\ Z_n.F_0(U_1, \ldots, U_{i-1}, Z_i, \ldots, Z_n)$$

and for even i,

$$U_i = \mathsf{GFP}\ Z_i.\mathsf{LFP}\ Z_{i+1} \ldots \mathsf{GFP}\ Z_n.F_0(U_1, \ldots, U_{i-1}, Z_i, \ldots, Z_n).$$

As for an even i the ordinal τ_i is not used, the definition implies that $\tau_i = 0$ for every even i. We prefer to have this redundancy rather than to multiply or divide indices by 2 again and again.

Fact 5. *A vertex v belongs to W_0 iff the canonical signature, $Sig(v)$, is defined.*

Proof. Suppose $v \in W_0$. Let τ be an ordinal of a cardinality bigger than the cardinality of $M(G)$. By the Knaster–Tarski theorem we have

$$W_0 = \|\mathsf{LFP}^{\tau}\ Z_1.\mathsf{GFP}\ Z_2 \ldots \mathsf{LFP}^{\tau}\ Z_{n-1}.\mathsf{GFP}\ Z_n.F_0(Z_1, \ldots, Z_n)\|^{M(G)}.$$

Hence $(\tau, \ldots, \tau)$ is an upper bound on the canonical signature for v. So, the signature is defined.

Conversely, suppose $Sig(v)$ is defined. For every ordinal ρ and every predicate $\alpha(X)$ we have $\|\mathsf{LFP}^{\rho}\ X.\alpha(X)\|^{M(G)} \subseteq \|\mathsf{LFP}\ X.\alpha(X)\|^{M(G)}$. Thus $v \in W_0$ by monotonicity of F_0. □

Fact 6. *The assignment $v \mapsto Sig(v)$ is a consistent signature assignment.*

Proof. We will consider only the case when $v \in V_0$. The case when $v \in V_1$ is similar. Let $(\tau_1, \ldots, \tau_n)$ be the canonical signature of v. Using definition of signature

(Definition 4), we have that $v \in \|F_0(U_1,\ldots,U_n)\|^{M(G)}$ with $U_1,\ldots,U_n$ as in that definition. Expanding the definition of F_0 we obtain

$$v \in \left|\left| \left\{ x\colon\ \exists y.E(x,y) \wedge \bigwedge_{i\in[n]} P_i(y) \Rightarrow U_i(y) \right\} \right|\right|^{M(G)}.$$

Suppose w is a node to take for the meaning of y, i.e.,

$$v \in \left|\left| \left\{ x\colon E(x,w) \wedge \bigwedge_{i\in[n]} P_i(w) \Rightarrow U_i(w) \right\} \right|\right|^{M(G)}.$$

Hence $E(v,w)$ holds and $w \in \|U_i\|^{M(G)}$ for $i = \Omega(w)$. Assume i is odd, the case when i is even is easier. By Definition 4 we know that

$$U_i = \mathsf{LFP}^{\tau_i}\ Z_i.\mathsf{GFP}\ Z_{i+1}\ldots\mathsf{GFP}\ Z_n.F_0(U_1,\ldots,U_{i-1},Z_i,\ldots,Z_n).$$

Ordinal τ_i may be a limit ordinal but still, by the definition of LFP^τ, there is a successor ordinal $\rho < \tau_i$ such that

$$w \in \|\mathsf{LFP}^{\rho}\ Z_i.\mathsf{GFP}\ Z_{i+1}\ldots\mathsf{GFP}\ Z_n.F_0(U_1,\ldots,U_{i-1},Z_i,\ldots,Z_n)\|^{M(G)}.$$

Using the definition of LFP^τ and in particular the fact that $\mathsf{LFP}^{\tau+1}\ Z.\alpha(Z)$ is equivalent to $\alpha(\mathsf{LFP}^\tau\ Z.\alpha(Z))$ we have

$$w \in \|\mathsf{GFP}\ Z_{i+1}.\mathsf{LFP}\ Z_{i+2}\ldots\mathsf{GFP}\ Z_n.F_0(U_1,\ldots,U_i',Z_{i+1},\ldots,Z_n)\|^{M(G)},$$

where $U_i' = \mathsf{LFP}^{\rho-1}\ Z_i.\mathsf{GFP}\ Z_{i+1}\ldots\mathsf{GFP}\ Z_n.F_0(U_1,\ldots,U_{i-1},Z_i,\ldots,Z_n)$. This shows that there are ordinals $\tau'_{i+1},\ldots,\tau'_n$ such that

$$w \in \|F_0(U_1,\ldots,U_i',U'_{i+1},\ldots,U'_n)\|^{M(G)}$$

where for odd $j > i$ we have

$$U_j' = \mathsf{LFP}^{\tau_j}\ Z_j.\mathsf{GFP}\ Z_{j+1}.\mathsf{LFP}\ Z_{j+2}\ldots\mathsf{GFP}\ Z_n.F_0(U_1,\ldots,U_i',\ldots,U'_{j-1},Z_j,\ldots,Z_n)$$

and for even $j > i$ we have

$$U_j' = \mathsf{GFP}\ Z_j.\mathsf{LFP}\ Z_{j+1}\ldots\mathsf{GFP}\ Z_n.F_0(U_1,\ldots,U_i',\ldots,U'_{j-1},Z_j,\ldots,Z_n).$$

This shows that $(\tau_1,\ldots,\tau_{i-1},\rho-1,\tau'_{i+1},\ldots,\tau'_n)$ is not smaller than the canonical signature of w. So we have

$$Sig(w) \leqslant (\tau_1,\ldots,\tau_{i-1},\rho-1,\tau'_{i+1},\ldots,\tau'_n) <_i (\tau_1,\ldots,\tau_{i-1},\rho,\tau_{i+1},\ldots,\tau_n) = Sig(v)$$

which means that $Sig(w) <_i Sig(v)$. □

Definition 7 (*Minimizing strategy*). A *minimizing strategy* is a strategy that takes for each node $v \in W_0 \cap V_0$ a son that has the smallest canonical signature.

Remark. There may be more than one minimizing strategy as a vertex may have many successors with the same signature.

The forgetful determinacy theorem follows from the next lemma.

Lemma 8. *The minimizing strategy is a winning memoryless strategy for player* 0 *from every node in* W_0. *From every node not in* W_0 *player* 1 *has a memoryless winning strategy.*

Proof. Suppose $v_0 \in W_0$. It should be clear that the canonical strategy is memoryless. We show that it is winning for player 0. Let $\mathscr{P} = v_0, v_1, \ldots$ be a play when player 0 uses the canonical strategy. To arrive at a contradiction assume that $\mathscr{P}$ is winning for player 1. In other words, the smallest priority appearing infinitely often on $\mathscr{P}$ is some odd number p.

Take an infinite sequence of positions $j_1 < j_2 < \cdots$ such that no vertex after v_{j_1} has priority smaller than p, and $\Omega(v_{j_k}) = p$ for $k = 1, \ldots$. From Fact 6 we obtain that $Sig(v_{j_k}) >_p Sig(v_{j_{k+1}})$. This is a contradiction because the lexicographical ordering on sequences of ordinals of bounded length is a well ordering.

To show the second statement of the theorem we use some propositional logic and dualities of the fixpoint calculus. For a predicate F one can define the predicate $\neg F$ by

$$\neg Z \equiv \{x : \neg Z(x)\}, \qquad \neg\{x : \psi(x)\} \equiv \{x : \neg\psi(x)\},$$

$$\neg \mathsf{LFP}\ Z.F(Z) \equiv \mathsf{GFP}\ Z.\neg F(\neg Z), \qquad \neg \mathsf{GFP}\ Z.F(Z) \equiv \mathsf{LFP}\ Z.\neg F(\neg Z)$$

It can be easily checked that the meaning of $\neg F$ is the complement of the meaning of F.

As W_0 is defined by (2), the complement of W_0 is defined by

$$\|\mathsf{GFP}\ Z_1.\mathsf{LFP}\ Z_2 \ldots \mathsf{GFP}\ Z_{n-1}.\mathsf{LFP}\ Z_n.\neg F_0(\neg Z_1, \ldots, \neg Z_n)\|^{M(G)}.$$

Using the propositional tautology

$$\neg((p \Rightarrow q) \wedge (\neg p \Rightarrow r)) \equiv ((p \Rightarrow \neg q) \wedge (\neg p \Rightarrow \neg r)),$$

we obtain

$$\neg F_0(\neg Z_1, \ldots, \neg Z_n) = \left\{ x : \left(V_0(x) \Rightarrow \forall y.\ E(x,y) \Rightarrow \neg \bigwedge_{i \in [n]} P_i(y) \Rightarrow \neg Z_i(y) \right) \right.$$

$$\left. \left(\neg V_0(x) \Rightarrow \exists y.\ E(x,y) \wedge \neg \bigwedge_{i \in [n]} P_i(y) \Rightarrow \neg Z_i(y) \right) \right\}.$$

Using the fact that in each vertex of G exactly one of the propositions $P_1,\dots,P_n$ holds, the formula above is equivalent to

$$\left\{x : \left(V_0(x)\Rightarrow\forall y.\ \left(E(x,y)\Rightarrow \bigwedge_{i\in[n]} P_i(y)\Rightarrow Z_i(y)\right)\right.\right.$$
$$\left.\left.\wedge\left(\neg V_0(x)\Rightarrow\exists y.\ E(x,y)\wedge \bigwedge_{i\in[n]} P_i(y)\Rightarrow Z_i(y)\right)\right)\right\}$$

Consider the game $G'=\langle V,V_1,E,\Omega'\rangle$ obtained from G by interchanging the vertices of player 0 and player 1 and letting $\Omega'(v)=\Omega(v)+1$. It is easy to see that a winning strategy for player 0 in G' translates to a strategy for player 1 in G and vice versa.

In the formulas above let us increase indices of the variables by one. Adding two dummy variables Z_1, Z_{n+2} we can see that in G' the complement of W_0 can be described by the formula

$$\mathsf{LFP}\ Z_1.\mathsf{GFP}\ Z_2\ldots.\mathsf{LFP}\ Z_{n+1}.\mathsf{GFP}\ Z_{n+2}.F_0'(Z_1,\dots,Z_{n+2}),$$

where

$$F_0'(Z_1,\dots,Z_{n+2})=\left\{x : \left(\neg V_0(x)\Rightarrow\exists y.\ E(x,y)\wedge \bigwedge_{i\in\{2,\dots,n+1\}} P_i(y)\Rightarrow Z_i(y)\right)\right.$$
$$\left.\wedge\left(V_0(x)\Rightarrow\forall y.\ \left(E(x,y)\Rightarrow \bigwedge_{i\in\{2,\dots,n+1\}} P_i(y)\Rightarrow Z_i(y)\right)\right)\right\}$$

(observe that the variables Z_1 and Z_{n+2} are not used). By the first statement of the lemma it follows that in G' there exists a memoryless winning strategy for player 0 from every node not in W_0. This strategy translates to a winning memoryless strategy for player 1 in G. □

4. Automata on trees

In this section we define automata running on trees. In full generality these automata are too powerful for applications we have in mind. In the later sections we will limit the power of the automata by restricting the class of allowed transition functions. The goal of this section is to study which closure properties of a class of transition functions are sufficient for the corresponding class of automata to be closed under sum, complementation and projection. The constructions from this section are variations on known constructions for automata on binary trees.

Let D be a nonempty set. A full D-tree is D^*, i.e., the set of all finite sequences of elements over D. For an alphabet Σ a Σ-labelled D-tree is a function $T : D^*\to\Sigma$.

Definition 9. Automaton on Σ-labelled D-trees is a tuple

$$\mathscr{A} = \langle Q, \Sigma, D, q^0, \delta : Q \times \Sigma \to (D^* \to \mathscr{P}(\mathscr{P}(Q \times D))), W \rangle, \tag{4}$$

where $W \subseteq Q^\omega$ is a set of infinite sequences over Q.

Intuitively, $\mathscr{A}$ is an alternating automaton. Being in a state q and in a node w labelled with $a = T(w)$ the automaton will choose a set $f \in \delta(q,a)(w)$. For every (q',d') in f it will send in the direction d' a copy of itself starting from the state q'. We will often write $\delta(q,a,w)$ instead of $\delta(q,a)(w)$.

A winning condition W is required to be a Borel set of paths of the tree Q^* (cf. [14, 11]). Most often W will be a parity winning condition. Recall from the previous section that such a condition is given by a function $\Omega : Q \to \mathbb{N}$. A sequence $q_0, q_1, \ldots$ satisfies the condition iff the smallest integer appearing infinitely often in the sequence $\Omega(q_0), \Omega(q_1), \ldots$ is even.

Remark. Please observe that our automata are not exactly finite. There are only finitely many states but the transition function and the winning condition are not finite. Usually, we will limit ourselves to finitely presented winning conditions, as for example parity conditions defined above. The range of the transition function δ may be infinite as soon as D is infinite. Also δ takes one argument from D^* which gives the automaton information about its position in the tree. This argument is a particular feature of our automata. It will be used in a very limited way but for now we can have it in full generality without any complications.

Definition 10. Given an automaton $\mathscr{A}$ as above and a tree $T : D^* \to \Sigma$ we define the game $G(\mathscr{A}, T)$ as follows:

- the set of player 0 vertices is $V_0 = Q \times D^*$;
- the set of player 1 vertices is $V_1 = \mathscr{P}(Q \times D) \times D^*$;
- the initial position of the game is (q^0, ε);
- there is an edge from a vertex $(q,w) \in V_0$ to a vertex $(f,w) \in V_1$ if $f \in \delta(q, T(w), w)$,
- there is an edge from a vertex $(f,w) \in V_1$ to a vertex $(q, wd) \in V_0$ if $(q,d) \in f$;
- a play $(q_1,w_1),(f_1,w_1),(q_2,w_2),(f_2,w_2),\ldots$ is winning for player 0 iff the sequence of states $q_1, q_2, \ldots$ is in W.

We say that $\mathscr{A}$ *accepts* T if player 0 has a winning strategy in the game $G(\mathscr{A}, T)$ from the initial position. The *language accepted* by $\mathscr{A}$, denoted $L(\mathscr{A})$, is the set of all Σ labelled D-trees accepted by $\mathscr{A}$.

Remark. If W is Borel then by Martin's Theorem the above game is determined. This is because whenever W is Borel in Q^* then its cylindrification is Borel in $(Q \times D)^*$ (a cylindrification of W is the set off all the sequences over $Q \times D$ such that the sequence of first components is in W)

Our goal is to show closure properties of the above automata. More precisely, we would like to study what transformations are needed to construct from a given automaton $\mathscr{A}$ automata accepting the complement, respectively projections, of $L(\mathscr{A})$. Before this we will give a simple construction of an automaton accepting the sum of the given two languages.

Let $\mathscr{A}_1$, $\mathscr{A}_2$ be two automata over the same alphabet Σ:

$$\mathscr{A}_i = \langle Q_i, \Sigma, D, q_i^0, \delta_i : Q_i \times \Sigma \to (D^* \to \mathscr{P}(\mathscr{P}(Q_i \times D))), W_i \rangle.$$

Assume that $Q_1 \cap Q_2 = \emptyset$. Take $q^0 \notin Q_1 \cup Q_2$. Consider

$$\mathscr{A}_\vee = \langle Q_\vee = Q_1 \cup Q_2 \cup \{q^0\}, \Sigma, D, q^0, \delta_\vee, W_\vee \rangle,$$

where $\delta_\vee(q^0, a, w)) = \delta_1(q_1^0) \cup \delta_2(q_2^0)$ and $\delta_\vee(q, a, w) = \delta_i(q, a, w)$ for $i = 1$ or 2 depending on whether $q \in Q_1$ or $q \in Q_2$ respectively. We also define $W_\vee$ to contain all the sequences $q_0, q_1, \ldots$ such that $q_0 = q^0$ and $q_1, q_2, \ldots$ is in $W_1 \cup W_2$. Obviously $W_\vee$ is Borel.

Lemma 11. $L(\mathscr{A}_\vee) = L(\mathscr{A}_1) \cup L(\mathscr{A}_2)$.

Proof. Let T be a tree. Observe that the game $G(\mathscr{A}, T)$ consists of the disjoint sum of the games $G(\mathscr{A}_1, T)$, $G(\mathscr{A}_2, T)$ and a new initial position (q^0, ε). From this initial position player 0 can move to every position reachable from the initial position of $G(\mathscr{A}_1, T)$ or $G(\mathscr{A}_2, T)$. Hence a strategy for player 0 in $G(\mathscr{A}, T)$ gives us either a strategy in $G(\mathscr{A}_1, T)$ or a strategy in $G(\mathscr{A}_2, T)$ depending on the first move. Similarly a strategy for player 0 in $G(\mathscr{A}_1, T)$ or $G(\mathscr{A}_2, T)$ gives rise to the strategy in $G(\mathscr{A}, T)$. □

Now we are going to describe the construction of an automaton accepting the complement of the given language.

Definition 12 (*Cover*). The *cover* of a set $F \subseteq \mathscr{P}(Q \times D)$ is the set $\bar{F} \subseteq \mathscr{P}(Q \times D)$ defined by: $\bar{f} \in \bar{F}$ iff for every $f \in F$ we have $f \cap \bar{f} \neq \emptyset$.

Lemma 13. *Let $\mathscr{A}$ be an automaton as in* (4). *Consider $\bar{\mathscr{A}} = \langle Q, \Sigma, D, q^0, \bar{\delta} : Q \times \Sigma \to \mathscr{P}(\mathscr{P}(D \times Q)), \bar{W} \rangle$ where $\bar{W} = Q^\omega \backslash W$ and $\bar{\delta}$ is defined by*

$$\bar{\delta}(q, a, w) = \bar{F} \quad \text{where } F = \delta(q, a, w).$$

For every tree T : $T \in L(\mathscr{A})$ iff $T \notin L(\bar{\mathscr{A}})$.

Proof. Suppose $T \in L(\mathscr{A})$. This means that there is a winning strategy σ for player 0 in the game $G(\mathscr{A}, T)$. We will show how to use this strategy to construct a winning

strategy for player 1 in $G(\bar{\mathcal{A}}, T)$. This implies that player 0 does not have a winning strategy in $G(\bar{\mathcal{A}}, T)$, hence T is not accepted by $\bar{\mathcal{A}}$.

The initial position in both plays is (q^0, ε). Suppose more generally that the plays continued for some time and the results of the finite plays so far are p in $G(\mathcal{A}, T)$ and $\bar{p}$ in $G(\bar{\mathcal{A}}, T)$. We assume that the play p on $G(\mathcal{A}, T)$ followed the strategy σ. We also assume that the projections of p and $\bar{p}$ on V_0 are the same (note that the sets of player 0 positions in the two games are the same). Let (q, w) be the last position in p as well as in $\bar{p}$. Now player 0 chooses in $G(\bar{\mathcal{A}}, T)$ a vertex $(\bar{f}, w)$ which is a successor of (q, w). We are going to consult the strategy σ to find an appropriate answer for player 1.

As p was obtained using strategy σ we know that $\sigma(p)$ is defined. Let $\sigma(p) = (f, w)$. By the definition of the game $G(\bar{\mathcal{A}}, T)$ we know that $f \cap \bar{f} \neq \emptyset$. Let $(q', d') \in f \cap \bar{f}$. Player 1 in $G(\bar{\mathcal{A}}, T)$ should choose (q', wd'). So we put $\bar{\sigma}(\bar{p}(q, w)(\bar{f}, w)) = (q', wd')$. This is also a possible choice for player 1 in $G(\mathcal{A}, T)$. The initial parts of the two plays become $p(q, w)(f, w)(q', wd')$ and $p'(q, w)(\bar{f}, w)(q', wd')$. From this point we can repeat the argument.

Whenever $\bar{p}$ is a play in $G(\bar{\mathcal{A}}, T)$ according to the strategy described above then we have a play p in $G(\mathcal{A}, T)$ such that the projections of $\bar{p}$ and p on V_0 are the same. We know that p is winning for player 0 in $G(\mathcal{A}, T)$ because p was played according to the winning strategy σ. Hence, by the definition of $\bar{W}$ play $\bar{p}$ is winning for player 1 in $G(\bar{\mathcal{A}}, T)$.

For the implication in the other direction suppose $T \notin L(\mathcal{A})$. Let σ_1 be a winning strategy for player 1 in $G(\mathcal{A}, T)$. Such a strategy exists because $G(\mathcal{A}, T)$ is determined. We are going to construct a winning strategy $\bar{\sigma}_0$ for player 0 in $G(\bar{\mathcal{A}}, T)$. As in the previous case we assume that the plays continued for some time and the results of the plays so far are p in $G(\mathcal{A}, T)$ and $\bar{p}$ in $G(\bar{\mathcal{A}}, T)$. We assume that the play p in $G(\mathcal{A}, T)$ followed the strategy σ_1. We also assume that the projections of p and $\bar{p}$ on V_0 are the same. Let (q, w) be the last position in p as well as in p'. We must now define $\bar{\sigma}_0(\bar{p})$. For every choice (f, w) of player 0 from (q, w) the strategy $\sigma_1(p(f, w))$ is defined. Let $(q_f, d_f) = \sigma_1(p(f, w))$. Let $\bar{f} = \{(q_f, d_f) : (f, w) \text{ is a successor of } (q, w)\}$. By the definition of automaton $\bar{\mathcal{A}}$ we know that $(\bar{f}, w)$ is a successor of (q, w) in game $\mathcal{G}(\bar{\mathcal{A}}, T)$. We put $\bar{\sigma}_0(\bar{p}) = (\bar{f}, w)$. Now player 1 chooses some $(q', d') \in \bar{f}$. By definition of $\bar{f}$ we can find f such that $(q', d') = (q_f, d_f)$. We make player 0 in $G(\mathcal{A}, T)$ to choose this f and then use strategy σ_1 to get to (q', wd'). The positions in the two games become $p(f, w)(q', wd')$ and $\bar{p}(\bar{f}, w)(q', wd')$. From these positions we can repeat the argument. By the similar reasoning as before, the strategy $\bar{\sigma}_0$ is winning for player 0. □

Definition 14. We say that an automaton as in (4) is *nondeterministic* iff for every $(q, a) \in Q \times \Sigma$, $w \in D^*$, $f \in \delta(q, a, w)$ and $d \in D$ there is at most one q' such that $(q', d) \in f$. In this case δ is called *nondeterministic transition function*.

Nondeterministic automata are interesting because projection operation is easy for them as we will now describe. The *projection* of a language L of trees over an alphabet

of the form $\Sigma \times \{0,1\}$ is the set of trees over the alphabet Σ defined by

$$\Pi_\Sigma(L) = \{T : \exists_{T' \in L}.\ \forall_{w \in D^*}.\ \exists_{b \in \{0,1\}}.\ T'(w) = (T(w), b)\}.$$

Let $\mathscr{A}$ be a nondeterministic automaton over the alphabet $\Sigma \times \{0,1\}$, i.e.

$$\mathscr{A} = \langle Q, \Sigma \times \{0,1\}, D, q^0, \delta : Q \times (\Sigma \times \{0,1\}) \to (D^* \to \mathscr{P}(\mathscr{P}(Q \times D))), W \rangle,$$

where δ is a nondeterministic transition function. We are going to define an automation for $\Pi_\Sigma(L(\mathscr{A}))$.

$$\mathscr{A}_\pi = \langle Q, \Sigma, D, q^0, \delta_\pi : Q \times \Sigma \to (D^* \to \mathscr{P}(\mathscr{P}(Q \times D))), W \rangle,$$

where $\delta_\pi(q,a,w) = \delta(q,(a,0),w) \cup \delta(q,(a,1),w)$ for all q and w.

Lemma 15. *If $\mathscr{A}$ is a nondeterministic automaton then $L(\mathscr{A}_\pi)$ accepts a projection of the language accepted by $L(\mathscr{A})$, i.e., $L(\mathscr{A}_\pi) = \Pi_\Sigma(L(\mathscr{A}))$.*

Proof. If σ is a winning strategy for player 0 in $G(\mathscr{A}, T')$ then the same strategy is also winning in $G(\mathscr{A}_\pi, T)$. This is because $G(\mathscr{A}_\pi, T)$ is obtained from $G(\mathscr{A}, T')$ by adding some more choices for player 0.

Suppose σ_π is a winning strategy for player 0 in $G(\mathscr{A}_\pi, T)$. Because $\mathscr{A}_\pi$ is a nondeterministic automaton, for every node $w \in D^*$ there is at most one state q such that a play respecting σ_π reaches the position (q,w). Let p be the history of this play. We can put $T'(w) = (T(w),0)$ if $\sigma(p) \in \delta(q,(T(w),0))$ and $T'(w) = (T(w),1)$ otherwise. This way we define the tree $T' : D^* \to \Sigma \times \{0,1\}$. Now σ_π becomes a winning strategy for player 0 in $G(\mathscr{A}, T')$. □

In the rest of this section we will describe how for a given parity automaton to find a nondeterministic automaton accepting the same language. We prefer to give the construction in several steps. This will be helpful for the next sections where we will code transition functions by formulas. First, we present some operations on transition functions.

Definition 16 (*Shift*). Let Q' be a finite set and let $q' \in Q'$. The *shift* by q' of a set $F \subseteq \mathscr{P}(Q \times D)$ is $F\mapsto^{q'}_{Q'} \subseteq \mathscr{P}((Q' \times Q) \times D)$ given by $f' \in F\mapsto^{q'}_{Q'}$ iff there is $f \in F$ such that $f' = \{((q',q),d) : (q,d) \in f\}$.

Definition 17 (*Join*). The *join* of two sets $F_1, F_2 \subseteq \mathscr{P}(Q \times D)$ is the set $F \subseteq \mathscr{P}(D \times Q)$ defined by $F = \{f_1 \cup f_2 : f_1 \in F_1 \wedge f_2 \in F_2\}$.

Definition 18 (*Collection*). The *collection* of a set $f \subseteq Q \times D$ is the set $col(f) \subseteq \mathscr{P}(Q) \times D$ defined by $(S,d) \in col(f)$ iff $S = \{q : (q,d) \in f\}$. A *collection* of a set $F \subseteq \mathscr{P}(Q \times D)$ is the set $col(F) \subseteq \mathscr{P}(\mathscr{P}(Q) \times D)$ defined by $col(F) = \{col(f) :\ f \in F\}$.

Let $\mathscr{A}$ be an automaton as in (4). We are going to define a nondeterministic automaton $\mathscr{A}_n$ accepting the same language. We let

$$\mathscr{A}_n = \langle Q_n = \mathscr{P}(Q \times Q), \Sigma, D, q_n^0 = \{(q^0, q^0)\}, \delta_n, W_n \rangle,$$

where it remains to define δ_n and W_n. Before doing this let us describe an intuition behind the construction. It is very similar to the subset construction used to determinize automata on finite words. The meaning of a state $S \in Q_n$ is that if $(q, q') \in S$ then there is a run reaching q' and q is a state just before q' in this run. This information about the predecessor is what makes the construction different from the standard one. It is necessary to define the acceptance condition. It is needed because we are interested in infinite runs and not in finite ones as it is the case of automata on finite words.

In order to define δ_n and W_n let us introduce an abbreviation. For a state $S \in Q_n$ we write $S \downarrow 2$ for the set of states appearing as the second components of the elements of S, i.e., $\{q' : \exists q.\ (q, q') \in S\}$.

We first define $\delta_n : Q_n \times \Sigma \to (D^* \to \mathscr{P}(\mathscr{P}(Q_n \times D)))$. Take $S \in Q_n$ and $a \in \Sigma$. Let $S \downarrow 2 = \{q_1, \ldots, q_k\}$. Let $F_i = \delta(q_i, a, w) \upharpoonright_Q^{q_i}$ for $i = 1, \ldots, k$. Take the join F' of $F_1, \ldots, F_k$. Put $\delta_n(S, a, w) = col(F')$.

Finally we want to define W_n, so we are going to say what infinite sequences $S_0, S_1, \ldots$ of subsets of $Q \times Q$ are winning for player 0. Let a *trace* in such a sequence be a sequence of states $q_0, q_1, q_2 \ldots$ such that $(q_i, q_{i+1}) \in S_i$ for all $i = 0, 1, 2, \ldots$. We put the sequence $S_0, S_1, \ldots$ into W_n iff all the traces in this sequence belong to W.

Lemma 19. *If $\mathscr{A}$ is a parity automaton then $L(\mathscr{A}) = L(\mathscr{A}_n)$ and $\mathscr{A}_n$ is a nondeterministic automaton.*

Proof. To see that $\mathscr{A}_n$ is a nondeterministic automaton observe that for every q, a, w, we have $\delta_n(q, a, w) = col(F)$ for some F. By definition of col operation for every $f \in col(F)$ and $d \in D$ there is exactly one state $S \in Q_n$ such that $(S, d) \in f$.

Now we want to show that $L(\mathscr{A}) \subseteq L(\mathscr{A}_n)$. Let $T : D^* \to \Sigma$ be a tree from $L(\mathscr{A})$ and let σ be a winning memoryless strategy for player 0 in $G(\mathscr{A}, T)$. Such a strategy exists by Theorem 1 because we have assumed that $\mathscr{A}$ is a parity automaton hence $G(\mathscr{A}, T)$ is a parity game. We define a memoryless strategy σ_n for player 0 from every position $(S, w) \in G(\mathscr{A}_n, T)$ such that $\sigma(q, w)$ is defined for every $q \in S \downarrow 2$. Given such (S, w) we take $(f_q, q) = \sigma(q, w)$ for every $q \in S \downarrow 2$. Let $f' = \bigcup_{q \in S \downarrow 2} \{((q, q'), d) : (q', d) \in f_q\}$. We put $\sigma(S, w) = (col(f'), w)$.

Because σ is a winning strategy in $G(\mathscr{A}, T)$ we know that $\sigma_n(\{(q_0, q_0)\}, \varepsilon)$ is defined and whenever $(f, w) \in \sigma_n(S, w)$ and $(S', d) \in f$ then $\sigma_n(S', wd)$ is defined. Moreover we have that if $(q, q') \in S'$ then $(q', d) \in f_q$ so player 1 can reach the position (q', wd) from $(\sigma(q, w), w)$. This shows that if $S_0, S_1, \ldots$ is a result of a play according to the strategy σ_n then every trace in this sequence is in W (because it is a play in $G(\mathscr{A}, T)$ according to the strategy σ). Hence the play is winning for player 0 and σ_n is a winning strategy.

For the other direction, let σ_n be a (not necessarily memoryless) winning strategy for player 0 in $G(\mathscr{A}_n, T)$. We will describe how player 0 can win every play in $G(\mathscr{A}, T)$.

The initial positions of the two games are (q^0,ε) and $(\{(q^0,q^0)\},\varepsilon)$. Suppose more generally that after some moves we have a history p_n of a play in $G(\mathscr{A}_n,T)$ and a history p of a play in $G(\mathscr{A},T)$. Let (S,w) be the last position in p_n. Assume that the last position in p is of the form (q,w) with $q\in S\downarrow 2$. Let $(f_n,w)=\sigma_n(p)$. Define $f=\{(q',d):\exists S'.\ (S',d)\in f_n\wedge((q,q'),d)\in S'\}$. By the definition of δ_n, we have $f\in\delta(q,T(w))$. We put $\sigma(p)=(f,w)$. Now player 1 chooses some position (q',wd) with $(q',d)\in f$. We transfer this move to $G(\mathscr{A}_n,T)$ by making player 1 choose (S_d,wd) where S_d is (unique) such that $(S_d,d)\in f_n$. Clearly $(q,q')\in S_d$ and we can repeat the whole procedure.

This way whenever we play according to the above strategy in $G(\mathscr{A},T)$, with a result p, we also obtain a play p_n in $G_n(\mathscr{A}_n,T)$ and p_n is winning as it follows σ_n. Moreover we have that p is trace in p_n. Hence p is winning for player 0 by the definition of W_n. □

The drawback of $\mathscr{A}_n$ is its quite complicated acceptance condition. Assuming that $\mathscr{A}$ is a parity automaton we are going to describe how to transform $\mathscr{A}_n$ into a nondeterministic parity automaton. The important observation is that the set of sequences in W_n is a regular language (of infinite words over the alphabet $\mathscr{P}(Q\times Q)$). Hence we can take a deterministic parity automaton accepting this language and "synchronize" it with the run of $\mathscr{A}_n$.

A parity automaton on infinite words over the alphabet $\Sigma=\mathscr{P}(Q\times Q)$ has the form

$$\mathscr{B}=\langle Q_b,\Sigma=\mathscr{P}(Q\times Q),q_b^0,\delta_b:(Q_b\times\Sigma)\to Q_b,\Omega_b\rangle$$

Recall that an infinite word over Σ is a function $s:\mathbb{N}\to\Sigma$. A run of $\mathscr{B}$ on such a word is a sequence of states $r:\mathbb{N}\to Q_b$ such that $r(0)=q_b^0$ and $r(i+1)\in\delta(r(i),s(i))$ for all $i\in\mathbb{N}$. A run is accepting iff it satisfies the parity condition given by Ω_b.

Suppose that $\mathscr{B}$ accepts exactly those sequences in which all traces satisfy the parity condition W. It is not difficult to convince oneself that such an automaton $\mathscr{B}$ exists. For this one can first consider the dual property: there is a trace in the sequence which does not satisfy the parity condition W. It is easy to construct a nondeterministic automaton accepting the sequences with this property. Using the fact that word automata with parity conditions are closed under complementation (cf. [22]) we obtain the desired automaton $\mathscr{B}$.

Define

$$\mathscr{A}_p=\langle Q_b\times\mathscr{P}(Q\times Q),\Sigma,D,(q_b^0,q_n^0),\delta_p,\Omega_p\rangle,$$

where $\delta_p((q_b,S),a,w)=\delta_n(S,a,w)\mapsto_{Q_b}^{q_b'}$ with $q_b'=\delta_b(q_b,S)$ and $\Omega_p(q_b,S)=\Omega_b(q_b)$.

Lemma 20. *$L(\mathscr{A}_p)=L(\mathscr{A}_n)$ and $\mathscr{A}_p$ is a parity automaton.*

Proof. Let σ be a winning memoryless strategy for player 0 in $G(\mathscr{A}_n,T)$. If $\sigma(S,w)$ is defined then we define $\sigma'((q_b,S),w)$ for every $q_b\in Q_b$. We put $\sigma'((q_b,S),w)=(f',w)$ where $f'=\sigma(S,w)\mapsto_{Q_b}^{q_b'}$ and $q_b'=\delta_b(q_b,S)$. With this definition we have that whenever

$(S_0, w_0)(S_1, w_1), \ldots$ is a play according to σ then the sequence $((q_b^0, S_0), w_0)((q_b^1, S_1), w_1)$, $\ldots$ is a play according to σ', where $q_b^0, q_b^1, \ldots$ is the run of $\mathscr{B}$ on $S_0, S_1, \ldots$. By the definition of $\mathscr{B}$, the first play is winning for player 0 according to W_n iff the second play is winning for player 0 according to the parity condition defined by Ω_p. This shows that σ' is winning. The other direction is analogous. □

Summarizing, we obtain the following corollary of the above constructions.

Corollary 21. *Let D be a nonempty set. Suppose that for every n we are given a subset $Def(D,[n])$ of the function space $D^* \to \mathscr{P}(\mathscr{P}([n] \times D))$ (recall that $[n]$ stands for $\{1,\ldots,n\}$). Suppose that whenever $\mathscr{F}_1, \mathscr{F}_2, \mathscr{F} \in Def(D,[n])$ then:*

- *(sum) $\mathscr{F}_\cup \in Def(D,[n])$, where $\mathscr{F}_\cup(w) = \mathscr{F}_1(w) \cup \mathscr{F}_2(w)$;*
- *(cover) $\bar{\mathscr{F}} \in Def(D,[n])$, where $\bar{\mathscr{F}}(w) = \overline{\mathscr{F}(w)}$ (cf. Definition 12);*
- *(shift) for every m and $k \in [m]$, $\mathscr{F} \upharpoonright^k_{[m]} \in Def(D,[n * m])$, where $\mathscr{F} \upharpoonright^k_{[m]}(w) = (\mathscr{F}(w)) \upharpoonright^k_{[m]}$ (cf. Definition 16);*
- *(join) $\mathscr{F}_\wedge \in Def(D,[n])$, where $\mathscr{F}_\wedge(w) = \{f_1 \cup f_2 : f_1 \in \mathscr{F}_2(w), f_2 \in \mathscr{F}_2(w)\}$ is the join of $\mathscr{F}_1(w)$ and $\mathscr{F}_2(w)$ (cf. Defnition 17);*
- *(collection) $col(\mathscr{F}) \in Def(D,[2^n])$, where $col(\mathscr{F})(w) = col(\mathscr{F}(w))$ (cf. Definition 18).*

Consider the class of automata with ranges of transition functions restricted to $\bigcup_{n=1,2,\ldots} Def(D,[n])$, i.e., automata of the form

$$\mathscr{A} = \langle [n], \Sigma, D, q^0, \delta : [n] \times \Sigma \to Def(D,[n]), \Omega : [n] \to \mathbb{N} \rangle$$

The set of languages recognised by such restricted automata is closed under sum, complementation and projection. Moreover for every restricted automaton there is an equivalent nondeterministic restricted automaton.

5. MSOL on tree-like structures

In this section we define certain tree-like structures and characterize the power of MSOL on such structures in terms of some restriction of automata from the previous section.

Let $Sig = \{r_1, \ldots\}$ be a signature containing relational symbols only. We start with the definition of an operation constructing tree-like structures. We call such structures iterated structures.

Definition 22 (*Iterated structure*). Let $M = \langle D_M, r_1, \ldots \rangle$ be a structure of the signature Sig. An *iterated structure* over the *extended signature* $Sig^* = Sig \cup \{son, cl\}$ is $M^* = \langle D_M^*, son\ cl, r_1^*, \ldots \rangle$ where D_M^* is the set of all finite sequences over D_M and the relations are defined by

$$son = \{(w, wd) : w \in D_M^*, d \in D_M\},$$

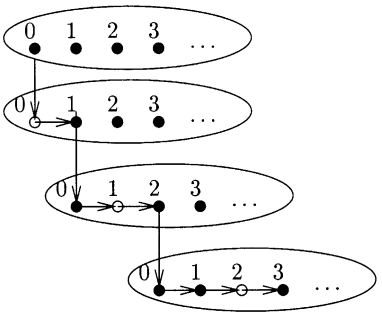

Fig. 1. Part of structure M^*.

$$cl = \{wdd : w \in D_M^*, d \in D_M\},$$

$$r^+ = \{(wd_1, \ldots, wd_k) : w \in D_M^*, (d_1, \ldots, d_k) \in r_M\} \quad (r \in Sig, k\text{-}ary).$$

Example. Consider the structure $M = \langle \{0,1\}, l, r\rangle$, where l holds only for 0 and r only for 1. The structure M^+ has as a domain the set $\{0,1\}^*$. The relation $son^M(w,w')$ holds if $w' = w0$ or $w' = w1$. The relation $l(w)$ holds iff w ends with 0, similarly $r(w)$ holds if w ends with 1. The clone relation holds for elements of the form $w00$ and $w11$. Hence M^* is the full binary tree with additional relations saying if a node is a left or a right son. The clone relation is not very useful in this example.

Example. Consider the structure $M = \langle \mathbb{N}, \leqslant\rangle$, where $\mathbb{N}$ is the set of natural numbers and $\leqslant$ is the usual order. $\mathbb{N}^*$ with son^* relation gives a tree of infinite countable branching. The relation $\leqslant^*$ linearly orders all siblings, i.e., nodes with the same father. The clone relation holds for a node w iff w is nth son of a father that is itself nth son (more formally if $w = w'nn$ for some n). This clone relation may be already useful. For example we can define in M^* the structure $\langle \mathbb{N}, \leqslant, P\rangle$ where P holds for numbers of the form $l(l+1)/2$. Elgot and Rabin [7] show decidability of the monadic theory of $\langle \mathbb{N}, <, P\rangle$ using different arguments.

The structure M^* and the construction is presented in Fig. 1. The empty circles show the clone relation. The arrows show the path isomorphic to the structure $\langle \mathbb{N}, \leqslant, P\rangle$. Predicate P holds in targets of downwards arrows.

We define in M^* the set of nodes $\{v_i\colon\ i \in \mathbb{N}\}$ where v_i is a sequence of subsequent numbers from 0 to i (that is $v_i = 012\ldots i$). This is easy using the clone relation because given v_i the node $v_i i$ (the sequence v_i extended with number i) is the son of v_i which is the clone. Then v_{i+1} is the next son of v_i. On the picture vertices v_i are the sources of downwards arrows. Having defined $\{v_i\colon\ i \in \mathbb{N}\}$ we can take a path starting from v_0 and going from each v_i to $v_i 0$ then $v_i 1$ up to $v_{i+1} = v_i i$ and then to its son $v_{i+1}0$ and so on. It is easy to see that this path is definable by a formula. Our path and the monadic predicate which interpretation is the set $\{v_i' : v_i' \text{ is a successor of some } v_i\}$ form a structure isomorphic to $\langle \mathbb{N}, \leqslant, P\rangle$.

The goal of this section is to give an automata characterisation of MSOL on tree like structures. For this we need a correspondence between models of formulas and labelled trees as inputs of automata.

Let M be a structure. Recall that a Σ-labelled tree over D_M is a function $T : D_M^* \to \Sigma$. An MSOL formula $\varphi(Z_1,\ldots,Z_k)$ defines the set of valuations $Val : \{Z_1,\ldots,Z_k\} \to \mathscr{P}(D_M^*)$ such that $M^*, Val \models \varphi(Z_1,\ldots,Z_k)$. A valuation Val defines a tree $T_{Val} : D_M^* \to \mathscr{P}(\{1,\ldots,k\})$ where $T_{Val}(w) = \{i : w \in Val(Z_i)\}$. Conversely, every such tree determines a valuation.

We say that an automaton $\mathscr{A}$ over the alphabet $\mathscr{P}(\{1,\ldots,k\})$ is *equivalent* to a formula $\varphi(Z_1,\ldots,Z_k)$ if

$$L(\mathscr{A}) = \{T_{Val} : M^*, Val \models \varphi(Z_1,\ldots,Z_k)\}.$$

Fact 23. *For every MSOL formula one can effectively construct an equivalent automaton.*

Proof. The construction of the automaton proceeds by induction on the structure of φ. By Corollary 21 our automata are closed under sum, complementation and projection. Hence it is enough to show how to handle atomic formulas: $Z_i \subseteq Z_j$, $\rho(Z_{i_1},\ldots,Z_{i_k})$, $son(Z_i,Z_j)$, and $cl(Z_i)$. All four automata for these four kinds of formulas will have the form

$$\mathscr{A}_m = \langle Q_m, \Sigma = \mathscr{P}(\{1,\ldots,k\}), D_M, q, \delta_m, \Omega_m\rangle$$

with Q_m, δ_m and Ω_m different for the four cases. Before giving the definitions below let us observe that if $\delta(q,a,w) = \emptyset$ then the automaton cannot accept in this configuration. If $\delta(q,a,w) = \{\emptyset\}$ then the automaton unconditionally accepts from this configuration. Now let us consider the atomic formulas one by one.

- For $Z_i \subseteq Z_j$ we take $Q_1 = \{q\}$, $\Omega_1(q) = 0$ and

$$\delta_1(q,a,w) = \begin{cases} \{\{(q,d) : d \in D\}\} & \text{if } i \notin a \text{ or } j \in a, \\ \emptyset & \text{otherwise.} \end{cases}$$

  This automaton checks whether labels of all the nodes satisfy the condition $i \notin a$ or $j \in a$.

- For $r(Z_{i_1},\ldots,Z_{i_k})$ we put $Q_2 = \{q, t_1,\ldots,t_k\}$ and $\Omega(q) = \Omega(t_1) = \cdots = \Omega(t_k) = 1$. We define the transition function by

$$\delta_2(q,a,w) = \{\{(q,d)\} : d \in D\} \cup \{\{(t_1,d_1),\ldots,(t_k,d_k)\} : r^M(d_1,\ldots,d_k)\},$$

$$\delta_2(t_j,a,w) = \begin{cases} \{\emptyset\} & \text{if } j \in a, \\ \emptyset & \text{otherwise.} \end{cases}$$

  This automaton finds nondeterministically a vertex d and its sons $d_1,\ldots,d_k$ such that $r(d_1,\ldots,d_k)$ and $d_1,\ldots,d_k$ are labelled by the appropriate letters.

- For $son(Z_i,Z_j)$ we put $Q_3=\{q,t\}$, and $\Omega_3(q)=\Omega_3(t)=1$. The transition function is defined by

$$\delta_3(q,a,w)=\begin{cases}\{\{(q,d)\}:d\in D\} & \text{if } i\notin a,\\ \{\{(q,d)\}:d\in D\}\cup\{\{(t,d)\}:d\in D\} & \text{otherwise.}\end{cases}$$

$$\delta_3(t,a,w)=\begin{cases}\{\emptyset\} & \text{if } j\in a,\\ \emptyset & \text{otherwise.}\end{cases}$$

  This automaton nondeterministically finds a vertex w with $i\in T(w)$ and its son wd with $j\in L(wd)$.
- For $cl(Z_i)$ we put $Q_4=Q_3$ and $\Omega_4=\Omega_3$. To define the transition function we let

$$\delta_4(q,a,w)=\begin{cases}\{\{(q,d)\}:d\in D\} & \text{if } w=\emptyset,\\ \{\{(q,d)\}:d\in D\}\cup\{\{(t,d^0)\}\} & \text{if } w=w'd^0,\end{cases}$$

$$\delta_4(t,a,w)=\begin{cases}\{\emptyset\} & \text{if } i\in a,\\ \emptyset & \text{otherwise.}\end{cases}$$

  Observe that this is the only case when the transition function depends on its third parameter. □

Of course, there are automata that are not equivalent to any formula. To see this it is enough to take a subset $R\subseteq D_M$ not definable by a MSOL formula in M^* (this is possible as soon as M is infinite and the signature is at most countable). It is straightforward to construct an automaton $\mathscr{A}_R$ recognizing all the trees $T:D_M^*\to\{0,1\}$ such that $T(w)=1$ iff $w\in R$. Hence $\mathscr{A}_R$ is not equivalent to any MSOL formula over M^*.

5.1. Automata characterisation of MSOL

The goal of this subsection is to limit the power of our automata so that for every automaton there will be a formula equivalent to it. The idea is to use MSOL formulas in the definition of transition functions.

In order to define a function $D_M^*\to\mathscr{P}(\mathscr{P}([n]\times D_M))$ for some $n\geqslant 1$ we can use a MSOL formula $\varphi(C,X_1,\ldots,X_n)$ with free second-order variables as displayed. Actually for all the functions $f:D_M^*\to\mathscr{P}(\mathscr{P}([n]\times D_M))$ the value $f(w)$ will depend only on the last element of the sequence w.

Definition 24. Let M be a structure. A MSOL formula $\varphi(C,X_1,\ldots,X_n)$ determines a function $\langle\!\langle\varphi(C,X_1,\ldots,X_n)\rangle\!\rangle_M$ from D_M^* to $\mathscr{P}(\mathscr{P}([n]\times D_M))$ defined by

- $f\in\langle\!\langle\varphi(C,X_1,\ldots,X_n)\rangle\!\rangle_M(wd)$ iff there exist $S_1,\ldots,S_n\subseteq D_M$ such that $M\models\varphi(\{d\},S_1,\ldots,S_n)$ and $f=\{(q,d):d\in S_q\}$.
- $f\in\langle\!\langle\varphi(C,X_1,\ldots,X_n)\rangle\!\rangle_M(\varepsilon)$ iff there exist $S_1,\ldots,S_n\subseteq D_M$ such that $M\models\varphi(\emptyset,S_1,\ldots,S_n)$ and $f=\{(q,d):d\in S_q\}$.

Let $Def(D_M,[n])$ be the set of all functions determined by MSOL formulas with free variables $C,X_1,\ldots,X_n$. A *restricted automaton* over M^* has the form

$$\mathscr{A} = \langle [n], \Sigma, D_M, \delta : [n] \times \Sigma \to Def(D_M,[n]), \Omega : [n] \to \mathbb{N} \rangle.$$

Suppose in the above $\Sigma = \mathscr{P}(\{1,\ldots,k\})$. We say that a restricted automaton $\mathscr{A}$ is equivalent over a structure M^* to a MSOL formula $\psi(Z_1,\ldots,Z_k)$ if for every valuation $Val : \{Z_1,\ldots,Z_k\} \to \mathscr{P}(D_M^*)$ we have

$$M^*,\ Val \models \psi(Z_1,\ldots,Z_k) \text{ iff } T^M_{Val} \in L(\mathscr{A}_\varphi).$$

Theorem 25. *Let M be a structure of signature Sig. For every MSOL formula there is an equivalent over M^* restricted automaton. For every MSOL automaton there is an equivalent over M^* MSOL formula.*

Proof. Let us consider the first statement. We construct an automaton by induction on the structure of the formula. For the cases of atomic formulas we show that the transition functions described in the proof of Fact 23 can be defined by MSOL formulas. This is done as follows:

$$\delta_1(1,a) = \begin{cases} \forall z.\ Y_1(z) & \text{if } j \in a \text{ or } i \notin a, \\ false & \text{otherwise,} \end{cases}$$

$$\delta_2(1,a) = (\exists z.\ Y_1(z)) \vee \left(\exists z_1,\ldots,z_k.\ r(z_1,\ldots,z_k) \wedge \bigwedge_{i=1,\ldots,k} Y_{i+1}(z_i) \right),$$

$$\delta_2(j+1,a) = \begin{cases} true & \text{if } i_j \in a, \\ false & \text{otherwise,} \end{cases}$$

$$\delta_3(1,a) = \begin{cases} \exists z.\ Y_1(z) & \text{if } i \notin a, \\ (\exists z.\ Y_1(z)) \vee (\exists z.\ Y_2(z)) & \text{otherwise,} \end{cases}$$

$$\delta_3(2,a) = \begin{cases} true & \text{if } j \in a, \\ false & \text{otherwise,} \end{cases}$$

$$\delta_4(1,a) = (\exists z. Y_1(z)) \vee (\exists z.\ C(z) \wedge Y_2(z)),$$

$$\delta_4(2,a) = \begin{cases} true & \text{if } i \in a, \\ false & \text{otherwise.} \end{cases}$$

For more complex formulas we use Corollary 21. For this we need to show that all the constructions required in the corollary are definable by MSOL formulas. Let φ, φ_1 and φ_2 be MSOL formulas with free variables among $C,X_1,\ldots,X_n$. We construct

- sum: $\varphi_\vee = \varphi_1 \vee \varphi_2$;
- cover: $\bar{\varphi} = \forall Y_1,\ldots,Y_n.\ \varphi(C,Y_1,\ldots,Y_n) \Rightarrow \bigvee_{i=1,\ldots,n} Y_i \cap X_i \neq \emptyset$;
- join: $\exists Y_1,\ldots,Y_n,Y'_1,\ldots,Y'_n.\varphi_1(C,Y_1,\ldots,Y_n.) \wedge \varphi_2(C,Y'_1,\ldots,Y'_n) \wedge \bigwedge_{i=1,\ldots,n} Y_i \cup Y'_i = X_i$;
- shift for some m and $k \in [m]$: $\varphi(C,X_{k*n+1},\ldots,X_{k*n+n})$;

- collect: $\exists Y_1,\ldots,Y_n.\varphi(C,Y_1,\ldots Y_n)\wedge\bigwedge_{S\subseteq[n]}\forall z.\ (z\in X_S)\Leftrightarrow(\bigvee_{i\in S} z\in Y_i)$.

For the second statement of the theorem observe that by Corollary 21 every restricted automaton is equivalent to a nondeterministic restricted automaton. It is easy to write a formula guessing a run of a nondeterministic automaton on M^*. □

Until now we have limited our considerations to an arbitrary but fixed structure of the signature *Sig*. It is useful to observe that the constructions from the above proof did not depend on this structure. Hence we get a more general notion of an automaton capable of running on every structure M^* with M a structure of the signature *Sig*.

Definition 26. For a fixed signature *Sig* an *MSOL automaton* has the form

$$\mathscr{A}=\langle Q=\{1,\ldots,n\},\Sigma,q^0,\delta:Q\times\Sigma\to Form(n),\Omega:Q\to\mathbb{N}\rangle,$$

where $Form(n)$ is the set of all MSOL formulas in the signature *Sig* with the free variables $C,X_1,\ldots,X_n$. For a given structure M with the carrier D_M such an automaton instantiates to an automaton on M^*:

$$\mathscr{A}_M=\langle Q,\Sigma,D_M,q^0,\delta_M:Q\times\Sigma\to(D_M^*\to\mathscr{P}(\mathscr{P}(Q\times D_M))),\Omega:Q\to\mathbb{N}\rangle,$$

where δ_M is defined by $\delta_M(q,a)=\langle\langle\delta(q,a)\rangle\rangle_M$ (cf. Definition 24).

The following corollary follows directly from the inspection of the proof of the previous theorem.

Corollary 27. *Fix a relational signature Sig. For every MSOL formula* $\psi(Z_1,\ldots,Z_k)$ *there is an MSOL automaton* $\mathscr{A}_\psi$ *over the alphabet* $\Sigma=\mathscr{P}(\{1,\ldots,k\})$ *such that for every structure M of signature Sig and every valuation* $Val:\{Z_1,\ldots,Z_k\}\to\mathscr{P}(D_M^*)$:

$$M^*,Val\models\psi(Z_1,\ldots,Z_k)\quad\textit{iff}\quad T^M_{Val}\in L(\mathscr{A}_\varphi).$$

From the proof of Theorem 25 we get also the following corollary showing how limited is our use of variable C (or equivalently D^* parameter) in transition functions.

Corollary 28. *If a formula does not use the clone relation then in the automaton obtained from the translation given in Theorem* 25 *no formula in the range of the transition function refers to the free variable C.*

A formula φ is called *monotone* in variables $X_1,\ldots,X_n$ if for every structure M and valuations Val, Val' such that $Val(X_i)\subseteq Val'(X_i)$ for all $i=1,\ldots,n$ we have that $M,Val\models\varphi$ implies $M,Val'\models\varphi$. With our definition of the run it is not necessary that the formulas in the image of the transition function are monotone. On the other hand, we can require that without a loss of expressive power.

Corollary 29. *Restricting the range of transition functions of automata to formulas monotone in the variables $X_1,\ldots,X_n$ (but not necessarily monotone in C) does not limit the power of MSOL automata.*

Proof. Let $\mathscr{A}$ be an automaton and suppose that for some state q and letter a the formula $\delta(q,a)=\alpha(C,X_1,\ldots,X_n)$ is not monotone. We can take its closure $\alpha'=\exists Y_1,\ldots,Y_n.\ Y_1\subseteq X_1\wedge\cdots\wedge Y_n\subseteq X_n\wedge\alpha(C,Y_1,\ldots,Y_n)$. Clearly $\alpha'(C,X_1,\ldots,X_n)$ is monotone in $X_1,\ldots,X_n$. We can put $\delta(q,a)=\alpha'$. The automaton $\mathscr{A}'$ obtained after the change is equivalent to $\mathscr{A}$. This follows directly from the definition of acceptance. Intuitively, this holds because the smaller the chosen sets in transitions are the easier it is to construct an accepting run. □

6. Muchnik's theorem

Recall that *Sig* is our arbitrary but fixed signature and Sig^* is its extension with *son* and *cl* relations. In this section we want to give a proof of Muchnik's theorem which is:

Theorem 30. *For every MSOL sentence φ over the signature Sig^* one can effectively find a MSOL sentence $\hat{\varphi}$ over the signature Sig, s.t. for every Sig-structure M:*

$$M\models\hat{\varphi}\quad \textit{iff}\quad M^*\models\varphi.$$

Take a MSOL sentence φ. By Corollary 27 we have a MSOL-automaton equivalent to φ over all iterated structures of the signature Sig^*. The input alphabet of this automaton has just one letter as φ is a sentence. To prove the above theorem it remains to show:

Lemma 31. *For every MSOL automaton $\mathscr{A}$ over one letter alphabet one can effectively find a MSOL sentence $\hat{\varphi}$ such that for every structure M,*

$$M\models\hat{\varphi}\ \textit{iff}\ M^*\ \textit{is accepted by}\ \mathscr{A}$$

Here, slightly overloading the notation, M^* stands for the unique Σ-labelled D_M-tree with Σ a singleton alphabet.

The proof of this lemma consists of several steps which fill the rest of this section.

MSOL transductions: First we will need a tool for constructing another structure from a given one. It is called the method of interpretations or transductions [17, 3]. We will need a quite simple instance of the general method.

Let Sig' be a relational signature. A (Sig,Sig')-transduction is a tuple

$$\Delta=\langle k,\{\theta_\chi\}_{\chi\in Sig'*[k]}\rangle$$

where

- $k>0$ is a natural number,
- $Sig' * [k] = \{(r', \vec{c}) : r' \in Sig', \vec{c} \in [k]^{\rho(r')}, \rho(r') \text{ arity of } r'\}$,
- every $\theta_{(r',\vec{c})}$ is a MSOL formula with as many free (second-order) variables as the arity of r'.

Let $M = \langle D_M, r_1, \ldots\rangle$ be a Sig-structure. The structure $\Delta(M)$ defined by the transduction Δ has the form $\langle D_M \times [k], r'_1, \ldots\rangle$ where

$$r'((d_1,i_1),\ldots,(d_l,i_l)) \text{ iff } M,[\{d_1\}/X_1]\cdots[\{d_l\}/X_l] \models \theta_\chi(X_1,\ldots,X_l)$$
$$\text{for } \chi = (r',(i_1,\ldots,i_l))$$

A well-known important property of such transductions [3] is

Fact 32. *For every MSOL sentence φ' of signature Sig' one an effectively find an MSOL sentence φ of signature Sig such that for every Sig-structure M we have*

$$M \models \varphi \quad \textit{iff} \quad \Delta(M) \models \varphi'.$$

Constructing structure $\mathcal{M}(\mathcal{A})$: Let us fix an automaton:

$$\mathcal{A} = \langle Q = [n], q^0, \delta : Q \to Form(n), \Omega : Q \to \mathbb{N}\rangle,$$

where we have intentionally omitted the alphabet (as being a singleton it plays no role in the behaviour of the automaton). Recall that $Form(n)$ is the set of MSOL formulas with free variables among $C, X_1, \ldots, X_n$. Without loss of generality we can assume that n is even and that $\Omega(i) = i$.

Take a structure $M = \langle D_M, r_1^M, \ldots\rangle$ of the signature Sig. Although our construction will be independent from the choice of a structure it is easier to describe it having a structure at hand.

Recall that we also use M^* to denote the unique D_M-tree over one letter alphabet. Consider the game $G(\mathcal{A}, M^*)$ as described in Definition 10. An important observation is that the game looks the same from every two positions (q, wd) and $(q, w'd)$, i.e., positions whose second components end in the same element. This allows us to identify all positions of the form (q, wd) and take (q, d) as the representative of all of them. In the result we obtain a smaller game $G'(\mathcal{A}, M)$ defined by

- the set of vertices of player 0 is $V_0 = Q \times D_M$;
- the set of vertices of player 1 is $V_1 = \mathcal{P}(Q \times D_M)$;
- the initial position of the game is (q^0, ε);
- there is an edge from a vertex $(q, d) \in V_0$ to $f \in V_1$ if $f \in \langle\langle \delta(q)\rangle\rangle_M(d)$ (cf. Definition 24)
- there is an edge from a vertex $f \in V_1$ to $(q, d) \in V_0$ if $(q, d) \in f$;
- a play $(q_1, d_1), f_1, (q_2, d_2), f_2, \ldots$ is winning for player 0 iff the sequence of states $q_0, q_1, \ldots$ satisfies the parity condition Ω.

Lemma 33. *Player* 0 *has a winning strategy from a vertex* (q, wd) *in game* $G(\mathscr{A}, M^*)$ *iff he has a winning strategy from the vertex* (q, d) *in game* $G'(\mathscr{A}, M)$.

Proof. The unwindings of $G(\mathscr{A}, M^*)$ from (q, wd) and of $G'(\mathscr{A}, M)$ from (q, d) are isomorphic. □

The game $G'(\mathscr{A}, M)$ can be presented as the structure

$$M(\mathscr{A}) = \langle V_0 \cup V_1, E, eq_2, V_0, V_1, s_1, \ldots, s_n, r_1, \ldots \rangle,$$

where

- $M(\mathscr{A}) \models E(e, e')$ iff there is an edge from e to e' in G';
- $M(\mathscr{A}) \models eq_2(e, e')$ iff there are q, q', d such that $e = (q, d)$ and $e' = (q', d)$, i.e., the second components of e and e' are the same;
- $M(\mathscr{A}) \models V_i(e)$ iff $e \in V_i$;
- $M(\mathscr{A}) \models s_q(e)$ iff $e \in V_0$, $e = (q, d)$ for some $d \in D_M$;
- $M(\mathscr{A}) \models r_j(e_1, \ldots, e_l)$ iff $M \models r_j(d_1, \ldots, d_l)$ and there is $q \in Q$ such that $e_1 = (q, d_1)$, $\ldots, e_l = (q, d_l)$.

Let $M(\mathscr{A})|V_0$ denote the restriction of $M(\mathscr{A})$ to V_0, i.e., the structure

$$M(\mathscr{A})|V_0 = \langle V_0, eq_2, s_1, \ldots, s_n, r_1, \ldots \rangle.$$

Observe that all the above relations were defined only on V_0 also in $M(\mathscr{A})$. The relation E is missing as it relates elements of V_0 with the elements of V_1.

Lemma 34. *For a given automaton* $\mathscr{A}$ *there is a transduction* Δ *defining from every Sig-structure* M *the structure* $\Delta(M) = M(\mathscr{A})|V_0$ *described above.*

Proof. Recall that the set of states of $\mathscr{A}$ is $Q = \{1, \ldots, n\}$. We have $Sig' = \{eq_2, s_1 \ldots, s_n\} \cup Sig$. We define (Sig, Sig') transduction $\Delta = \langle n, \{\theta_w\}_{Sig' \times [n]} \rangle$ where

$$\begin{aligned}
&\theta_{eq_2,(i,j)} = (X_1 = X_2) \quad \text{for all } i, j \in [n],\\
&\theta_{s_i,i} = true \quad \text{for all } i \in [n],\\
&\theta_{s_i,j} = false \quad \text{for } i \neq j,\\
&\theta_{r,(i_1,\ldots,i_l)} = r(X_1, \ldots, X_l) \quad \text{if } i_1 = i_2 = \ldots = i_l,\\
&\theta_{r,(i_1,\ldots,i_l)} = false \quad \text{if not the case above.}
\end{aligned}$$ □

Describing the winning set in $M(\mathscr{A})$: Theorem 1 gives us an MSOL predicate defining the set of winning positions for player 0. Rewriting this predicate for the special case of our structure $M(\mathscr{A})$ we obtain

$$F'_0(Z_1, \ldots, Z_n) = \left\{ x : \left(V_0(x) \Rightarrow \exists y.(xEy \wedge y \in Z_n) \right) \wedge \left(V_1(x) \Rightarrow \forall z.(xEz \Rightarrow \bigwedge_{i \in [n]} (s_i(z) \Rightarrow z \in Z_i)) \right) \right\}. \tag{5}$$

The simplification in the first line is possible because whenever $x \in V_0$ and xEy then $y \in V_1$ and $s_n(y)$ holds. The set of winning vertices for player 0 is given by

$$\mathscr{S} = \|\mathsf{LFP}\ Z_1 \ldots \mathsf{GFP}\ Z_n.F_0'(Z_1,\ldots,Z_n)\|^{M(\mathscr{A})}. \tag{6}$$

We are interested in the set $\mathscr{S} \cap V_0$ so we can simplify the formula even further. We use the fact that our graph is bipartite so every edge from a vertex in V_1 leads to a vertex in V_0. This allows to replace F_0' with the predicate

$$F_0''(Z_1,\ldots,Z_n) = \left\{ x : V_0(x) \wedge \right.$$
$$\left. \exists y \in V_1.\left(xEy \wedge \left(\forall z.yEz \Rightarrow \bigwedge_{i\in[n]} (s_i(z) \Rightarrow z \in Z_i) \right) \right) \right\}. \tag{7}$$

It is straightforward to check:

Lemma 35. *For every valuation Val the predicates* $\mathsf{GFP}\ Z_n.F_0'$ *and* $\mathsf{GFP}\ Z_n.F_0''$ *define the same subset of* V_0 *in the structure* $M(\mathscr{A})$.

Hence in (6) instead of $\mathsf{GFP}\ Z_n.F_0'$ we can take $\mathsf{GFP}\ Z_n.F_0''$ and we will obtain the same set of vertices from V_0.

Restricting to elements from V_0: The above tells us how to describe the winning positions for player 0 in the acceptance game $G(\mathscr{A}, M^*)$ as a subset of $M(\mathscr{A})$. The problem is that we cannot define $M(\mathscr{A})$ from M by means of a transduction. What we can do is to define $M(\mathscr{A})|V_0$ by a transduction. So we will refine the formula (7) to define the set of winning positions in $M(\mathscr{A})|V_0$.

As $V_1 = \mathscr{P}(Q \times D_M) = \mathscr{P}(V_0)$ an element $e \in V_1$ can be represented by a set of elements of V_0. Hence instead of writing $\exists y \in V_1$ in (7) we can write $\exists Y \subseteq V_0$. Then we also need to express the edge relation somehow. We are going to write a formula $\alpha(x, Y)$ such that for every valuation Val we have

$$M, Val \models xEy \text{ iff } M, Val[S/Y] \models \alpha(x, Y)$$
$$\text{where } S = \{(q,d) \in V_0 : (q,d) \in Val(y)\}.$$

and moreover in $\alpha(x, Y)$ all quantifiers will be restricted to V_0.

Suppose we have such a formula. Then we could define.

$$F_0^{(3)}(Z_1,\ldots,Z_n) = \left\{ x : V_0(x) \wedge \right.$$
$$\left. \exists Y \subseteq V_0.\ \alpha(x, Y) \wedge \left(\forall z.\ z \in Y \Rightarrow \bigwedge_{i\in[n]} (s_i(z) \Rightarrow z \in Z_i) \right) \right\}.$$

This predicate is equivalent to F_0'' and refers only to the vertices from V_0. Hence it defines the same set of vertices in $M(\mathscr{A})$ and in $M(\mathscr{A})|V_0$.

Lemma 36. *For every valuation Val*:

$$\|F_0^{(3)}(Z_1,\ldots,Z_n)\|_{M(\mathscr{A})|V_0}^{Val|V_0} = \|F_0''(Z_1,\ldots,Z_n)\|_{M(\mathscr{A})}^{Val}$$

where $Val|V_0$ *is the valuation defined by* $Val|V_0(X) = \{e \in V_0 : e \in Val(X)\}$. *In the consequence*:

$$\|\mathsf{LFP}\ Z_1 \ldots \mathsf{GFP}\ Z_n.F_0^{(3)}(Z_1,\ldots,Z_n)\|_{M(\mathscr{A})|V_0} = \|\mathsf{LFP}\ Z_1 \ldots \mathsf{GFP}\ Z_n.F_0''(Z_1,\ldots,Z_n)\|_{M(\mathscr{A})}.$$

Constructing $\alpha(x,Y)$: To write the formula $\alpha(x,Y)$ we must recall how the edge relation of the game is defined. From a node (q,d) there is an edge to $f \in \mathscr{P}(Q \times D_M)$ if $f \in \langle\!\langle \delta(q) \rangle\!\rangle_M(d)$. Denoting $\delta(q)$ by $\beta_q(C,X_1,\ldots,X_n)$ and using Definition 24 we have

$$f \in \langle\langle \delta(q) \rangle\rangle_M(d) \text{ iff } M \vDash \beta_q(\{d\},\{d : (1,d)\,\mathrm{inf}\},\ldots,\{d : (n,d) \in f\}).$$

Let $S_1 = \{(1,d) : d \in D_M\}$. Clearly S_1 is a subset of V_0. Let $In_1(X)$ be the predicate which holds iff the value of X is a subset of S_1. Next we construct $\beta_q^{(1)}$ which is β_q with all quantifiers relativized to S_1 (i.e., to sets X satisfying $In_1(X)$)

$$\beta_q'(C,X_1,\ldots,X_n) = In_1(C) \wedge In_1(X_1) \wedge \cdots \wedge In_1(X_n) \wedge \beta_q^{(1)}(C,X_1,\ldots,X_n).$$

Hence if $M(\mathscr{A}), Val \vDash \beta_q'$ then $Val(C), Val(X_1),\ldots,Val(X_n)$ are subsets of S_1 and moreover for every $Val : Var \to \mathscr{P}(D_M)$ we have

$$M, Val \vDash \beta_i \text{ iff } M(\mathscr{A}), Val' \vDash \beta_i',$$

where $Val'(Y) = \{(1,d) : d \in Val(Y)\}$ for every variable Y.

Let us write the formula

$$\gamma_q(x,Y) = \exists C, Y_1,\ldots,Y_n.\ \beta_q'(C,Y_1,\ldots,Y_n) \wedge (Singl(C) \wedge eq_2(C,x)) \wedge Collapse(Y,Y_1,\ldots,Y_n),$$

where the second part of this formula says that $C = \{(1,d)\}$ for d such that $x = \{(q,d)\}$ for some q. The formula $Collapse(Y,Y_1,\ldots,Y_n)$ says that $Y = \{(q,d) : (1,d) \in Y_q\}$; it can be written as

$$\forall y. \bigwedge_{q\in[n]} ((y \in Y \wedge s_q(y)) \Leftrightarrow (\exists y'.\ y' \in Y_i \wedge eq_2(y,y') \wedge In_1(y'))).$$

The formula $\alpha(x,Y)$ to be constructed is

$$\bigwedge_{q\in[n]} s_q(x) \Rightarrow \gamma_q(x,Y).$$

Putting it all together (Proof of Theorem 30). By Lemma 36

$$W_0 = \|\mathsf{LFP}\ Z_1 \ldots \mathsf{GFP}\ Z_n.F_0^{(3)}(Z_1,\ldots,Z_n)\|^{M(\mathscr{A})|V_0}$$

is the set of winning positions from V_0 in the game $G'(\mathscr{A}, M)$. More precisely (q,d) belongs to W_0 iff player 0 can win from position (q,d) in $G'(\mathscr{A}, M)$. By Lemma 33 it means that he can also win from (q,d) in the acceptance game $G(\mathscr{A}, M^*)$. Player 0 can win in the game $G(\mathscr{A}, M^*)$ from the initial position (q^0, ε) iff he can find $f \in \langle\langle \delta(q^0) \rangle\rangle_M$ such that for every $(q,d) \in f$ he can win from (q,d). This can be expressed in $M(\mathscr{A})|V_0$ by the sentence

$$\exists C, Y_1, \ldots, Y_n, Y, Z.\ C = \emptyset \wedge \beta_0^{(1)}(C, Y_1, \ldots, Y_n) \wedge \mathit{Collapse}(Y, Y_1, \ldots, Y_n) \wedge Y \subseteq W_0, \tag{8}$$

where $\beta_0^{(1)}$ is a formula obtained by relativizing all quantifiers in $\beta_0 = \delta(q^0)$ to S_1, exactly as in the construction of $\alpha(x, Y)$ above. The above sentence holds in $M(\mathscr{A})|V_0$ iff player 0 has a winning strategy in the acceptance game $G(\mathscr{A}, M^*)$. Because $M(\mathscr{A})|V_0$ is definable by a transduction from M, there is a sentence $\hat{\varphi}$ such that $M \models \hat{\varphi}$ iff formula (8) is satisfied in $M(\mathscr{A})|V_0$ which is exactly when $\mathscr{A}$ accepts M^*.

7. MSOL on trees of arbitrary degree

In this section we will show an automata characterisation of MSOL over trees of arbitrary degree. From this characterisation we will deduce that MSOL over trees is equivalent to first-order logic with a unary least fix point operator (FPL for short).

A tree can be represented as a structure over the signature consisting of one binary relation *son*. A tree is a structure satisfying the axioms: there is unique root; from every node there is unique finite path from the root to the node; every element has a son. Let *Trees* stand for the class of trees. Observe that we only consider trees were every node has finitely many predecessors. Equivalently we could say that each of our trees is a prefix closed subset of A^* for some set A; where A^* is the set of all finite sequences over A.

To define the syntax of first-order logic extended in with unary fixpoint operator (FPL) fix sets $\{x_1, x_2, \ldots\}$ and $\{X_1, X_2, \ldots\}$ of first-order and second-order variables respectively. The atomic formulas of FPL are of the form $\mathit{son}(x_i, x_j)$ and $X_i(x_j)$. The set of formulas is closed under boolean connectives, first-order quantification and the following rule for constructing fixpoints:

if $\psi(X,z)$ is a formula of FPL with variable X occurring only positively in ψ then

$[\mathsf{LFP}\ X(z).\psi(X,z)](z)$ and $[\mathsf{GFP}\ X(z).\psi(X,z)](z)$ are formulas of FPL.

Recall that an occurrence of X in a formula is *positive* if it is preceded by an even number of negations.

To define the meaning of a formula in a tree T we need a valuation Val assigning a node of T to every first-order variable and a set of nodes of T to every second-order variable. Then we have:

- $T, \mathit{Val} \models X_i(x_j)$ if $\mathit{Val}(x_i) \in \mathit{Val}(X_j)$,
- $T, \mathit{Val} \models \mathit{son}(x_i, x_j)$ if $\mathit{Val}(x_j)$ is a son of $\mathit{Val}(x_i)$ in T,
- all first-order constructs are interpreted in the standard way,

- $T, Val \models [\mathsf{LFP}\ X(z).\psi(X,z)](z)$ iff $Val(z)$ belongs to the least fixpoint of the monotone operator that for a set $S \subseteq T$ returns the set $\{c : T, Val[S/X][c/z] \models \psi(X,z)\}$,
- $T, Val \models [\mathsf{GFP}\ X(z).\psi(X,z)](z)$ is defined similarly to the previous clause but taking the greatest fixpoint this time.

There is an easy translation giving for every FPL formula an equivalent MSOL formula (translating first-order variables to second-order variables ranging over singletons). The goal of this section is to show that there is a translation in the opposite direction giving an equivalent formula over *Trees*.

Theorem 37. *For every MSOL formula φ there is a FPL formula ψ s.t., $Trees \models \varphi \Leftrightarrow \psi$.*

Let *Set* be the class of structures over the empty signature. These are just plain sets considered as relational structures. Consider Set^*, the class of structures M^* for M in *Set*. A structure M^* from Set^* is a structure of the signature $\{son, cl\}$. It is the full tree over M with not very useful clone relation.

We have a characterisation of the expressive power of MSOL over such structures in terms of MSOL automata. Our goal is to show that we can "translate" these automata into FPL formulas. Recall that in MSOL automata the transition function has the from $\delta : Q \times \Sigma \to Form(|Q|)$ where $Form(|Q|)$ are all MSOL formulas with free variables in $C, X_1, \ldots, X_{|Q|}$. Here we are interested only in the formulas not referring to the clone relation. Hence by Corollary 28 we can assume that variable C does not appear in the formulas defining the transition function of an automaton. By Corollary 29 we can assume that all the formulas in the range of the transition function are monotone.

To get an even simpler form of automata we now show that every monotone MSOL formula over *Set* is equivalent to a first-order formula of a very simple form.

For every $n > 0$ we define *n-type* to be a formula of the form

$$\tau(z) = \left(\bigwedge_{i \in S} X_i(z) \right) \wedge \left(\bigwedge_{i \in [n] \setminus S} \neg X_i(z) \right)$$

for some $S \subseteq [n]$. For a tuple of variables $y_1, \ldots, y_k$ let $\mathrm{diff}(y_1, \ldots, y_k)$ be a formula saying that the meanings of the variables are pairwise different.

Let $\mathrm{BF}(n)$, the set of basic formulas with n variables, be the set of sentences of the form

$$\exists y_1, \ldots, y_k.\ \mathrm{diff}(y_1, \ldots, y_k) \wedge \theta_1(y_1) \wedge \cdots \wedge \theta_k(y_k) \wedge \forall z.\ \mathrm{diff}(z, y_1, \ldots, y_k) \Rightarrow \bigvee_{j=1,\ldots,l} \theta'_j(z), \qquad (9)$$

where $k, l \in \mathbb{N}$, and for every $i = 1, \ldots, k$ and $j = 1, \ldots, l$ the formulas $\theta_i(z)$ and $\theta'_j(z)$ are positive parts of n-types, i.e., are of the form $\bigwedge_{s \in S} X_s(z)$ for some $S \subseteq [n]$. A $\mathrm{DBF}(n)$ formula is a finite disjunction of $\mathrm{BF}(n)$ formulas.

Lemma 38. *Every monotone MSOL formula with free variables $X_1,\ldots,X_n$ is equivalent over Set to a* DBF(n) *formula.*

Proof. Let φ be a monotone MSOL formula with free variables $X_1,\ldots,X_n$. By a standard argument using Ehrenfeucht–Fraïssé games (cf. [6]) φ is equivalent to a finite disjunction $\alpha=\bigvee_{i\in I}\alpha_i$ with each α_i of the form

$$\exists y_1,\ldots,y_k.\ \mathrm{diff}(y_1,\ldots,y_k)\wedge\tau_1(y_1)\wedge\ldots\wedge\tau_k(y_k)\\ \wedge\forall z.\ \mathrm{diff}(z,y_1,\ldots,y_k)\Rightarrow\bigvee_{j=1,\ldots,l}\tau'_j(z)$$

for some types $\tau_1,\ldots,\tau_k,\tau'_1,\ldots,\tau'_l$.

If we assume that the initial formula was monotone then we can "forget" about the negative parts of types. More formally, consider $\bar{\alpha}$ obtained from α by deleting from each τ_i and τ_j all negative atoms, i.e., formulas of the form $\neg X_s(x)$. It is easy to check that if for some structure $M\in Set$ and valuation Val we have $M,Val\models\bar{\alpha}$ then there is Val' such that $M,Val'\models\alpha$ and $Val'(X)\subseteq Val(X)$ for all $X\in Var$. □

Now, from Corollary 27 we obtain:

Lemma 39. *For every MSOL formula over the signature $\{son\}$ with free variables $Z_1,\ldots,Z_k$ there is an equivalent, over Set^*, automaton of the form*

$$\langle Q=[n],\Sigma,q^0,\delta:Q\times\Sigma\to\ \mathrm{DBF}(n),\Omega:Q\to\mathbb{N}\rangle \tag{10}$$

where $\Sigma=\mathscr{P}(\{1,\ldots,k\})$.

To prove equivalence of MSOL and FPL over Set^* it is enough to give a translation of automata of the above kind into FPL formulas. This is essentially what we are going to do but first we have to overcome one technical difficulty. We want to show that the logics are equivalent not over Set^* but over a slightly bigger class *Trees* of structures. The difference is that Set^* contains only full trees while in *Trees* there can be trees with varied degrees of nodes. Of course every tree $T\in Trees$ can be represented in some $M^*\in Set^*$ for sufficiently big M. This is formalized in the definition below.

Definition 40. Let $T\in Trees$ be a tree and $Val:Var\to\mathscr{P}(T)$ a valuation. Let $\check{Y}$ be a distinguished variable. We say that (M^*,Val') *represents* (T,Val) if $Val':Var\cup\{\check{Y}\}\to\mathscr{P}(T)$ and

- there is an isomorphism h from T to $Val'(\check{Y})$,
- for every $X\in Var$ we have $Val'(X)=h(Val(X))$.

Lemma 41. *For every MSOL formula φ there is an MSOL formula φ^* such that for every tree (T,Val) and every (M^*,Val') representing (T,Val) we have $T\models\varphi$ iff $(M^*,Val')\models\varphi^*$.*

Proof. Let $\check{Y}$ be the distinguished variable defining T in M^*. We assume that $\check{Y}$ does not appear in φ. Formula φ^* is obtained from φ by relativizing all the quantifiers to $\check{Y}$. □

Putting together the above lemma and Lemma 39 we obtain for every MSOL formula over *Trees* an automaton over Set^* accepting exactly the representations of trees satisfying the formula. The next step is to obtain automata running directly on *Trees*. The only problem here is to define what happens if the automaton reaches a leaf. The simplest solution is to extend the definition of the automaton by adding a set F of terminal states:

$$\mathscr{A} = \langle Q = [n], \Sigma, q^0, \delta : Q \times \Sigma \to \mathrm{DBF}(n), F \subseteq Q, \Omega : Q \to \mathbb{N} \rangle \tag{11}$$

The notion of acceptance for such an automaton is essentially the same as for MSOL automata except for the fact that player 0 wins if he reaches a leaf with a state from F. Let us nevertheless spell out the definition more formally to avoid possible confusion and profit from some simplifications coming from the fact that the automaton runs on trees.

Definition 42. Given an automaton $\mathscr{A}$ as above (with $\Sigma = \mathscr{P}(\{1,\dots,k\})$) and a tree T with a valuation $Val : \{Z_1,\dots,Z_k\} \to \mathscr{P}(T)$ we define the game $G(\mathscr{A},(T,Val))$ as follows:

- The set of player 0 vertices is $V_0 = Q \times T$.
- The set of player 1 vertices is $V_1 = (\{X_1,\dots,X_n\} \to \mathscr{P}(T))$.
- The initial position of the game is (q^0, ε).
- Let M_w be the set of sons of w. If $M_w \neq \emptyset$ then there is an edge from a vertex $(q,w) \in V_0$ to $f \in V_1$ if $M_w, f \models \delta(q, Val(w))$, where $f : \{X_1,\dots,X_n\} \to \mathscr{P}(M_w)$ is a valuation and $Val(w)$ denotes the letter $\{i : w \in Val(Z_i)\} \in \Sigma$.
- There is an edge from a vertex $(f,w) \in V_1$ to $(q,w') \in V_0$ if $w' \in f(X_q)$.
- A finite play terminating in a position (q,w) with w a leaf is winning for player 0 iff $q \in F$.
- An infinite play $(q_1,w_1),(f_1,w_1),(q_2,w_2),(f_2,w_2),\dots$ is winning for player 0 iff the sequence $\Omega(q_0),\Omega(q_1),\dots$ satisfies the parity condition.

We say that $\mathscr{A}$ *accepts* (T,Val) if player 0 has a winning strategy in the game $G(\mathscr{A},(T,Val))$ from the initial position.

Remark. The only important difference between the above definition and automata on iterated structures (cf. Definition 10) is that we have to take care of the situation when a node of a tree has no sons. This is done by putting no edges from vertices (q,w) with w a leaf and adding a new winning condition for such vertices.

Lemma 43. *For every MSOL formula φ with free variables $Z_1,\dots,Z_k$ there is an automaton $\mathscr{A}_\varphi$ over the alphabet $\Sigma = \mathscr{P}(\{1,\dots,k\})$ such that for every tree $T \in Trees$ and valuation Val we have $T, Val \models \varphi$ iff $\mathscr{A}$ accepts (T,Val).*

Proof. Let $\varphi(Z_1,\ldots,Z_k)$ be a MSOL formula in the signature $\{son\}$. By Lemma 41 there is a formula $\varphi^*(Z_1,\ldots,Z_k,\check{Y})$ such that for every tree T with a valuation Val and for every structure M^* with a valuation Val' representing $(T,Val):T,Val\models\varphi$ iff $M^*,Val'\models\varphi^*$. By Lemma 39 there is an automaton

$$\mathscr{A}^*=\langle Q=[n],\Sigma=\mathscr{P}(\{1,\ldots,k+1\}),q^0,\delta:Q\times\Sigma\to \mathrm{DBF}(n),\Omega\rangle$$

equivalent to φ^*. Observe that whenever (M^*,Val') represents (T,Val) then for every $v\in D_M\setminus Val'(\check{Y})$ we have that $v\notin Val'(Z)$ for every Z. Hence the tree from every node not in $Val'(\check{Y})$ is the same: it is isomorphic to $(M^*,Val_\emptyset)$ where $Val_\emptyset$ is the valuation assigning the empty set to every variable. Let F be the set of states from which the automaton accepts infinitely many structures of the form $(M^*,Val_\emptyset)$. Consider then the tree automaton with F as the set of finial states:

$$\mathscr{A}^*=\langle Q=[n],\Sigma=\mathscr{P}(\{1,\ldots,k\}),q^0,\delta^*:Q\times\Sigma\to \mathrm{DBF}(n),F,\Omega\rangle,$$

where $\delta^*(q,a)=\delta(q,a\cup\{n+1\})$. This is the automaton required by the lemma. □

It remains to translate tree automata to FPL formulas.

Lemma 44. *For every automaton $\mathscr{A}$ as in* (11) *over an alphabet $\Sigma=\mathscr{P}(\{1,\ldots,k\})$ there is an FPL formula ψ with free variables $\{Z_1,\ldots,Z_k\}$, such that, for every tree T and valuation $Val:\{Z_1,\ldots,Z_k\}\to\mathscr{P}(T)$:*

$$(T,Val)\ \textit{is accepted by}\ \mathscr{A}\ \textit{iff}\ T,Val\models\psi(Z_1,\ldots,Z_k)$$

Proof. Let $\mathscr{A}$ be a tree automaton

$$\mathscr{A}=\langle Q=[n],\Sigma,q^0,\delta:Q\times\Sigma\to \mathrm{DBF}(n),F\subseteq Q,\Omega:Q\to\mathbb{N}\rangle$$

with $\Sigma=\mathscr{P}(\{1,\ldots,k\})$. For simplicity of the notation we assume that $\Omega(i)=i$. This way the state itself carries information about its priority.

The game $G(\mathscr{A},(T,Val))$ can be represented as a relational structure $M_G=\langle V_0\cup V_1, E,V_0,V_1,s_1,\ldots,s_n\rangle$ where

- $M_G\models E(e,e')$ iff there is an edge from e to e' in $G(\mathscr{A},(T,Val))$;
- $M_G\models V_i(e)$ iff $e\in V_i$;
- $M_G\models s_q(e)$ iff $e\in V_0$ and the first component of e is q.

By Theorem 1 the set of winning positions in the game $G(\mathscr{A},(T,Val))$ can be described in M_G by the MSOL predicate

$$\mathsf{LFP}\ X_1.\mathsf{GFP}\ X_2\ldots\mathsf{LFP}\ X_{n-1}.\mathsf{GFP}\ X_n.F_0(X_1,\ldots,X_n),$$

where

$$F_0(X_1,\ldots,X_n)=\left\{x:\left(V_0(x)\Rightarrow\exists y.\ E(x,y)\wedge\bigwedge_{i\in[n]}s_i(y)\Rightarrow y\in X_i\right)\right.$$
$$\left.\wedge\left(V_1(x)\Rightarrow\forall y.\ E(x,y)\Rightarrow\bigwedge_{i\in[n]}s_i(y)\Rightarrow y\in X_i\right)\right\}.$$

Using the fact that s_n holds for all the elements of V_1 and that the game graph is bipartite with the partition (V_0,V_1) we can simplify F_0 to F_0':

$$F_0'(X_1,\ldots,X_n)=\left\{x:\left(\exists y.\ E(x,y)\wedge\left(\forall z.\ E(y,z)\Rightarrow\bigwedge_{i\in[n]}s_i(z)\Rightarrow z\in X_i\right)\right)\right\}.$$

Next we will use the fact that E is defined by first-order formulas and translate this formula to a FOL formula over trees.

First, for every $q\in Q$ we define a projection function $h_q:\mathscr{P}(V_0)\to\mathscr{P}(T)$ by $h_q(S)=\{w:(q,w)\in S\}$. In particular, player 0 wins in $G(\mathscr{A},(T,Val))$ iff the root of T is in $h_{q^0}(W_0)$ where q^0 is the initial state of $\mathscr{A}$ and W_0 the set of winning positions for player 0 in the game. To define $h_{q^0}(W_0)$ in (T,Val) we will construct formulas γ_q^i for all $i\in[n+1]$ and $q\in Q$. The formulas will have the property that for every valuation $Val:Var\to M_G$ we have

$$\{w:T,Val\models\gamma_q^i(w,h_1(Val(X_1)),\ldots,h_{i-1}(Val(X_{i-1})))\}$$
$$=h_q\left(\left|\left|\begin{matrix}\mathsf{LFP}\\ \mathsf{GFP}\end{matrix}X_i\ldots\ \mathsf{GFP}\ X_n.F_0'(X_1,\ldots,X_n)\right|\right|_{M_G}^{Val}\right),\qquad(12)$$

where $\begin{matrix}\mathsf{LFP}\\ \mathsf{GFP}\end{matrix}X_i$ is $\mathsf{LFP}\ X_i$ or $\mathsf{GFP}\ X_i$ depending on whether i is odd or even respectively.

As a base case for the induction we define γ_q^{n+1} for all $q\in Q$,

$$\gamma_q^{n+1}(x,X_1,\ldots,X_n)=\bigwedge_{a\in\Sigma}a(x)\Rightarrow[\delta(q,a)]_x,\qquad(13)$$

where we must still define $a(x)$ and $[\delta(q,a)]_x$. For a letter $a\in\Sigma=\mathscr{P}(\{1,\ldots,k\})$ the formula $a(x)$ is $(\bigwedge_{i\in a}X_i(x))\wedge(\bigwedge_{i\notin a}\neg X_i(x))$. The formula $[\delta(q,a)]_x$ is obtained from $\delta(q,a)$ by relativizing all the quantifiers to the sons of x. Recall that $\delta(q,a)$ is a disjunction of formulas of the form $\exists y_1,\ldots,y_k.\beta_1\wedge\forall z.\beta_2$. In $[\delta(a,q)]_x$ each disjunct is changed to $\exists y_1,\ldots,y_k.son(x,y_1)\wedge\cdots\wedge son(x,y_k)\wedge\beta_1\wedge\forall z.son(x,z)\Rightarrow\beta_2$.

For $i\leqslant n$ we define γ_q^i assuming all $\gamma_{q'}^{i+1}$ are defined. For γ_i^i we put

$$\gamma_i^i=\left[\begin{matrix}\mathsf{LFP}\\ \mathsf{GFP}\end{matrix}X_i(x).\gamma_i^{i+1}\right](x)$$

and for γ_q^i with $q\neq i$ we put

$$\gamma_q^i=\bigwedge_{a\in\Sigma}a(x)\Rightarrow[\delta(q,a)(X_1,\ldots,X_{i-1},\gamma_i^i,\gamma_{i+1}^{i+1}[\gamma_i^i/X_i],\ldots,\gamma_n^{i+1}[\gamma_i^i/X_i])]_x.$$

The fact that γ_q^{n+1} satisfies the property (12) follows from the inspection of the definition of M_G. To see that γ_q^i satisfies the property (12) observe that for every ordinal τ we have

$$\left\{ w : T, V \vDash \left[\begin{matrix} \mathsf{LFP} \\ \mathsf{GFP} \end{matrix}^{\tau} X_i.\gamma_q^{i+1}(x, h_1(Val(X_1)), \ldots h_{i-1}(Val(X_{i-1}))) \right](w) \right\}$$

$$= h_q \left(\left\| \begin{matrix} \mathsf{LFP} \\ \mathsf{GFP} \end{matrix}^{\tau} X_i. \begin{matrix} \mathsf{LFP} \\ \mathsf{GFP} \end{matrix} X_{i+1} \ldots M_G \right\|_{M_G}^{Val} \right)$$

(here $\begin{matrix} \mathsf{LFP} \\ \mathsf{GFP} \end{matrix}^{\tau} X_i$ stands for the τ-th approximation $\mathsf{LFP}^{\tau} X_i$ or $\mathsf{GFP}^{\tau} X_i$ depending on whether i is odd or even respectively). This observation follows from the induction hypothesis saying that the property (12) holds for γ_q^{i+1} and from the distributivity of h_q over unions and intersections, i.e., $h_q(\cup B) = \cup h_q(B)$ and $h_q(\cap B) = \cap h_q(B)$. To show that γ_q^i with $i \neq q$ satisfies the property (12) one can use the induction hypothesis and the unwinding rule which says that $\mathsf{LFP}\ Z.F(Z)$ is equivalent to $F(\mathsf{LFP}\ Z.F(Z))$ (and analogously for the GFP operator). □

The proof of Theorem 37 follows from Lemmas 43 and 44. By the first lemma, for every formula of MSOL there is an equivalent tree automaton. By the second lemma, there is a FPL formula equivalent to this automaton.

7.1. Concluding remarks

We have considered an operation M^* of constructing tree-like structures and defined automata working on the structures of this kind. These automata are parametrised by a class of allowed transition functions. We have given some conditions on the class of transition functions which guarantee that automata are closed under sum, complement and projection. We use this parametrisation to give two different results using different classes of transition functions.

Other classes of transition functions are possible. One can consider transition functions of alternating automata on binary trees or the ones corresponding to the mu-calculus on arbitrary trees [9]. In all these cases transition functions can be defined by some special first-order formulas. One can also consider transition functions defined by a superset of MSOL formulas, as for example MSOL with counting modulo predicates. It would be interesting to know the connection of these automata to MSOL with counting and to the fixpoint logic with counting. One can also try to use the automata on more complicated structures than just tress. For example, one can consider structures M^* for M being a countable well-order. A conjecture is that MSOL on such structures is equivalent to the unary fixpoint logic. For the proof of it one should presumably use Büchi's automata on countable ordinals [1] and characterize their expressive power using fixpoint logic. Then the machinery presented in this paper can probably be used. Another question concerning the operation M^* is to investigate to which extent the

first-order counterpart of Muchnik's Theorem holds. Suppose we know that the first-order theory of M is decidable, what can be said about decidability of the first-order theory of M^*?

Acknowledgements

I would like to thank Bruno Courcelle, David Janin, Damian Niwiński and Wieslaw Zielonka for stimulating discussions and comments. I am also grateful to anonymous referees for helpful comments and to D. Berwanger and A. Blumensath for pointing out many misprints.

References

[1] J.R. Büchi, Decision methods in the theory of ordinals, Bull. Amer. Math. Soc. 71 (1965) 767–770.
[2] J.R. Buchi, State strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$, J. Symbolic Logic 48 (1983) 1171–1198.
[3] B. Courcelle, Monadic second-order graph transductions: a survey, Theoret. Comput. Sci. 126 (1994) 53–75.
[4] B. Courcelle, I. Walukiewicz, Monadic second-order logic, graphs and unfoldings of transition systems, Ann. Pure Appl. Logic 92 (1998) 35–62.
[5] G. D'Agostino, M. Hollenberg, Uniform interpolation, automata and the modal mu-calculus, in: M. Kracht, M. de Rijke, H. Wansing, M. Zakharyaschev (Eds.), Advances in Modal Logic, Vol. 1, CSLI Publishers, Stanford, 1996.
[6] H.-D. Ebbinghaus, J. Flum, Finite Model Theory., Springer, Berlin, 1995.
[7] C.C. Elgot, M.O. Rabin, Decidability and undecidability of extensions of second (first) order theory of (generalized) successor, J. Symbolic Logic 31 (1966) 169–181.
[8] E.A. Emerson, C.S. Jutla, Tree automata, mu-calculus and determinacy, Proceedings of the FOCS 91 (1991) 368–377.
[9] D. Janin, I. Walukiewicz, Automata for the μ-calculus and related results, Proceedings of the MFCS '95 Lecture Notes in Computer Science Vol. 969 (1995) 552–562.
[10] D. Janin, I. Walukiewicz, On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic, Proceedings of the CONCUR'96 Lecture Notes in Computer Science Vol. 1119 (1996) 263–277.
[11] A.S. Kechris, in: Classical Descriptive Set Theory, Graduate Texts in Mathematics, Vol. 156, Springer, Berlin, 1995.
[12] N. Klarlund, Progress measures, immediate determinacy and a subset construction for tree automata, Proceedings of the LICS '92 (1992) 382–393.
[13] R. McNaughton, Infinite games played on finite graphs, Ann. Pure Appl. Logic 65 (1993) 149–184.
[14] Y. Moschovakis, in: Descriptive Set Theory, Studies in Logic, Vol. 100, North-Holland, Amsterdam, 1980.
[15] A.W. Mostowski, Games with forbidden positions. Tech. Report No. 78, University of Gdansk, 1991.
[16] M. Rabin, Decidability of second-order theories and automata on infinite trees, Trans. Amer. Math. Soc. 141 (1969) 1–35.
[17] M. Rabin, Decidable theories, in: J. Barwise (Ed.), Handbook of Mathematical Logic, Elsevier, Amsterdam, 1977.
[18] A. Semenov, Decidability of monadic theories, Proceedings of the MFCS'84, Lecture Notes in Computer Science, Vol. 176, Springer, Berlin, 1984, pp. 162–175.
[19] S. Shelah, The monadic second order theory of order, Ann. Math. 102 (1975) 379–419.
[20] R.S. Streett, E.A. Emerson, An automata theoretic procedure for the propositional mu-calculus, Inform. Comput. 81 (1989) 249–264.
[21] J. Stupp, The lattice-model is recursive in the original model, Institute of Mathematics, The Hebrew University, Jerusalem, January 1975.

[22] W. Thomas, Languages, automata, and logic, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 3, Springer, Berlin, 1997.
[23] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theoret. Comput. Sci. 200 (1998) 135–183.