An extension of Rabin's tree theorem to tree-like structures

Igor Walukiewicz
Institute of Informatics
Warsaw University
Banacha 2, 02-097 Warsaw
Poland
igw@mimuw.edu.pl

Abstract

A tree *like structure* M^+ is the structure obtained from M by taking a copy of M for every finite sequence of elements in M and putting a tree structure over these copies. We consider automata on tree-like structures. We show sufficient conditions for these automata to be closed under sum, complementation and projection. We give three applications of the result.

1 Synopsis

Given a relational structure M, a *tree-like* structure M^+ is the structure containing a copy of M for every finite sequence of elements of M together with a tree structure on these copies. The notion of tree-like structure generalises the notion of a binary tree. The later can be obtained by taking for M a two element structure over the empty signature.

We study a possibility of extending automata characterisation of monadic second order logic (MSOL) from binary trees to tree-like structures. We describe a form of automaton running on tree-like structures. The definition of these automata is parametrised by a set of *basic formulas*. Taking different sets of basic formulas it is possible to obtain automata of different expressive power. We give a sufficient conditions on a set of basic formulas for automata with these formulas to be closed under sum, complementation and projection.

We show three applications of our automata characterisation. We prove Muchnik's Theorem. The theorem says that for every MSOL formula φ one can effectively find a MSOL formula $\widehat{\varphi}$, s.t., for every structure M: $M^+ \vDash \varphi$ iff $M \vDash \widehat{\varphi}$. This can be seen as a generalisation of Rabin's theorem about decidability of S2S to a larger class of structures.

We show an automata characterisation of MSOL over trees of arbitrary degree. This characterisation allows us to deduce that MSOL over trees is equivalent to first order logic extended with unary least fixpoint operators.

Finally, we show that every MSOL sentence defining a bisimulation closed class of structures is equivalent to a μ-calculus sentence.

The material presented here is based on [3, 4, 12]. Muchnik Theorem was first presented in [9]. Very much recommended background reading is [11].

2 Preliminaries

Let $Sig = \{r_1, \dots\}$ be a signature containing only relational symbols, let $Var_1 = \{x, y, z, x_1, \dots\}$ and $Var_2 = \{X, Y, Z, X_1, \dots\}$ be the sets of first and second order variables respectively. The set of *predicates* and *formulas* of MSOL is defined by mutual recursion as follows:

- every variable from Var_2 is a predicate; if $\varphi(x)$ is a formula and $F(X)$ is a predicate with X occurring only positively in F then $\{x : \varphi(x)\}$, $\mathsf{LFP}\, X.F(X)$, $\mathsf{GFP}\, X.F(X)$ are predicates.
- If $P_1, \dots, P_n$ are predicates, $x \in Var_1$, $X \in Var_2$ and α, β are formulas then: $x \in P_1$, $r(P_1, \dots, P_k)$, $\neg\alpha$, $\alpha \vee \beta$, $\exists x.\alpha$, $\exists X.\alpha$ are formulas.

In a model $M = \langle D_M, r_1^M, \dots \rangle$ each relation symbol is interpreted as a set of k-tuples of elements of D_M; k being the arity of the relational symbol. The variables from Var_1 range over elements of D^M and the variables from Var_2 over all subsets of D^M. The meaning of a predicate P in a given model M and a valuation Val is the set of elements of D_M denoted $\| P \|^M_{Val}$. The meaning of a formula is either true or false. If it is true we denote this fact by $M, Val \vDash \varphi$.

The clauses for the meanings of disjunction, negation and existential quantification are standard. In particular we have existential quantification over sets of elements of D_M. Slightly abusing the syntax we will write formulas $r(X_1, \dots, X_k)$. Such a formula will be true iff each second order variable X_i is interpreted as a singleton set $\{d_i\}$ and $r^M(d_1, \dots, d_k)$ holds.

The meaning of a predicate $\{x : \varphi(x)\}$ in a given model $\mathcal{M}$ and a valuation Val is the set of elements of D_M that when taken for the meaning of x make the formula $\varphi(x)$ true. Formally we have:

$$\| \{x : \varphi(x)\} \|^M_{Val} = \{d \in D^M : M, Val[d/x] \vDash \varphi(x)\}$$

Observe that there can be free second order variables in $\varphi(x)$. Varying the meaning of such a free variable Z the meaning of the whole predicate $F(Z) = \{x : \varphi(x, Z)\}$ varies. Hence $F(Z)$ defines an operator form $\mathcal{P}(D^M)$ to $\mathcal{P}(D^M)$. As Z appears only positively in F, this operator is monotone and has the least and the greatest fixpoints. These will be the meanings of $\mathsf{LFP}\, X.F(X)$ and $\mathsf{GFP}\, X.F(X)$ respectively.

$$\mathsf{LFP}\, X.F(X) = \bigcap \{S \subseteq D^M : \| F(X) \|^M_{\mathit{Val}[S/X]} \subseteq S\}$$
$$\mathsf{GFP}\, X.F(X) = \bigcup \{S \subseteq D^M : S \subseteq \| F(X) \|^M_{\mathit{Val}[S/X]}\}$$

Remark: For convenience we did not to take a minimal syntax for MSOL but it should be clear that for every formula in our elaborate language is equivalent to a formula using only second order variables, relational symbols, inclusion relation, negation, disjunction and existential quantification. In other words all the other constructs are definable in this core language. We will use this fact when making induction on the structure of the formulas.

3 Parity games and canonical strategies

In this section we recall the definition of parity games. We give an explicit description of winning strategies in *parity 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*.

The notion of signature was proposed by Streett and Emerson [10]. The proof of the existence of memoryless strategies in parity games was given independently by Mostowski [8] and by Emerson and Jutla [2]. Klarlund [5] 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 conditions are given as a Rabin condition. In [6, 11, 13] one can find beautiful presentations of the theorem together with discussions on its applications. The approach here is based on [2]. It is more technical but gives us a fixpoint characterisation of the winning set. We will extensively use this characterisation in the later sections.

Let $G = \langle V, V_0 \subseteq V, E \subseteq V \times V, \Omega : V \to \{1, \ldots, n\}\rangle$ be a graph with vertices labelled by *priorities* from $\{1, \ldots, n\}$. The set $V_0 \subseteq V$ describes the positions from which player 0 is to move. We let $V_1 = V \setminus V_0$. For convenience we assume that G is bipartite with edges between V_0 and V_1.

A *game* from some vertex $v_0 \in V_0$ is played as follows: first player 0 chooses a vertex v_1 with $E(v_0, v_1)$, then player 1 chooses a vertex v_2 with $E(v_1, v_2)$, and so on ad infinitum unless one of the players cannot make a move. If a player cannot make a move he looses. The result of an infinite play is an infinite path $v_0, v_1, v_2, \ldots$ This path is *winning for player* 0 if in the sequence $\Omega(v_1), \Omega(v_2), \Omega(v_3), \ldots$ the smallest number appearing infinitely often is even. Otherwise the path is *winning for player* 1. 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 sequence of vertices $\vec{v}$ ending in a vertex from V_0 a vertex $\sigma(\vec{v}) \in V_1$, such that, $E(v, \sigma(\vec{v}))$. A strategy is *memoryless* iff $\sigma(\vec{v}) = \sigma(\vec{w})$ whenever $\vec{v}$ and $\vec{w}$ end in the same vertex. A strategy is *winning* iff it guarantees a win for player 0 whenever he follows the strategy. Similarly we define a strategy for player 1.

We will often consider strategies that are partial functions. To fit our definition one can assume that these are total functions which values for some elements don't matter.

Our main goal in this section is the following theorem:

Theorem 1 (Forgetful determinacy)
Let G be a parity game. From every node of G one of the players has a forgetful winning strategy.

The idea of the proof is the following. First we define a set of nodes $\mathcal{W}_0$ of G by a special fixpoint formula. Having this formula, for every vertex in $\mathcal{W}_0$ we associate a *signature* which intuitively says how far is the vertex from something good. We use signatures to define a winning memoryless strategy for player 0 from vertices in $\mathcal{W}_0$. Finally it turns out that the complement of $\mathcal{W}_0$ is defined by a formula of exactly the same shape as the one defining $\mathcal{W}_0$. This gives us a memoryless winning strategy for player 1 from vertices not in $\mathcal{W}_0$.

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

$$G = \langle V, V_0, E, \Omega : V \to \{1, \ldots, n\}\rangle$$

To avoid considering two cases we assume that the range of Ω is $\{1, \ldots, n\}$ and that n is even. Clearly we can do so without a loss of generality. The graph G can be represented as a relational structure $\langle V, E^G, V_0^G, P_1^G, \ldots, P_n^G\rangle$, where: V is the carrier; E^G defines an edge relation between states and $V_0^G, P_1^G, \ldots, P_n^G$ are subsets of V. Each P_i^G (for $i = 1, \ldots, n$) denotes the set of nodes with priority i, i.e., the set $\{v : \Omega(v) = i\}$.

Consider the predicate:

$$\begin{aligned} F_0(Z_1, \ldots, Z_n) = \big\{x :& \big(V_0(x) \Rightarrow \exists y.\ E(x,y) \wedge \mathit{Next}(y, Z_1, \ldots, Z_n)\big) \\ &\wedge \big(\neg V_0(x) \Rightarrow \forall y.\ E(x,y) \Rightarrow \mathit{Next}(y, Z_1, \ldots, Z_n)\big)\big\} \end{aligned} \tag{1}$$

where

$$Next(y, Z_1, \ldots, Z_n) = \bigwedge_{i \in \{1,\ldots,n\}} P_i(y) \Rightarrow Z_i(y)$$

We will be interested in the set:

$$\mathcal{W}_0 = \| \ \mathsf{LFP}\, Z_1.\, \mathsf{GFP}\, Z_2. \ldots \mathsf{LFP}\, Z_{n-1}.\, \mathsf{GFP}\, Z_n.F_0(Z_1, \ldots, Z_n) \ \|^G \tag{2}$$

(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).

Definition 2 When applied to n-tuples of ordinals, symbols $=, <, \leq$ stand for corresponding relations in the lexicographical ordering. For every $i \in \{1, \ldots, n\}$ we use $=_i$ to mean that both arguments are defined and when truncated to first i positions the two vectors are equal; similarly for $<_i$ and $\leq_i$.

Definition 3 (Consistent signature assignment) A signature is a n-tuple of ordinals. An assignment $\mathcal{S}$ of signatures to nodes from some set $U \subseteq V$ will be called *consistent* if for every $v \in U \cap V_0$ there is a vertex $w \in U$ with $E(v, w)$ and such that:

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

similarly, if $v \in U \cap V_1$ then for all vertices w with $E(v, w)$ we have $w \in U$ and the condition (3) holds.

We extend the syntax of the formulas by allowing constructions of the form $\mathsf{LFP}^\tau\, Z.\alpha(Z)$, where τ is an ordinal and $\alpha(Z)$ is a formula from the extended syntax. The semantics is defined as follows:

$$\| \ \mathsf{LFP}^0\, Z.\alpha(Z) \ \|^{\mathcal{M}}_V = \emptyset \qquad \| \ \mathsf{LFP}^{\tau+1}\, Z.\alpha(Z) \ \|^{\mathcal{M}}_V = \| \ \alpha(Z) \ \|^{\mathcal{M}}_{V[\| \mathsf{LFP}^\tau\, Z.\alpha(Z) \|^{\mathcal{M}}_V / Z]}$$

$$\| \ \mathsf{LFP}^\tau\, Z.\alpha(Z) \ \|^{\mathcal{M}}_V = \bigcup_{\rho < \tau} \| \ \mathsf{LFP}^\rho\, Z.\alpha(Z) \ \|^{\mathcal{M}}_V \quad (\tau \text{ a limit ordinal})$$

By Knaster-Tarski theorem $\| \ \mathsf{LFP}\, Z.\alpha(Z) \ \|^{\mathcal{M}}_V = \bigcup_\tau \| \ \mathsf{LFP}^\tau\, Z.\alpha(Z) \ \|^{\mathcal{M}}_V$.

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:

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

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 calculate right indices each time.

Fact 5 A vertex v belongs to $\mathcal{W}_0$ iff the canonical signature, $\mathit{Sig}(v)$, is defined.

Proof
Suppose $v \in \mathcal{W}_0$. Let τ be an ordinal of a cardinality bigger than the cardinality of G. By Knaster-Tarski theorem we have:

$$\mathcal{W}_0 = \|\ \mathsf{LFP}^{\tau}\, Z_1.\, \mathsf{GFP}\, Z_2 \ldots \mathsf{LFP}^{\tau}\, Z_{n-1}.\, \mathsf{GFP}\, Z_n.F_0(Z_1, \ldots, Z_n)\ \|^G$$

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

Conversely, suppose $\mathit{Sig}(v)$ is defined. For every ordinal ρ and every predicate $\alpha(X)$ we have $\|\ \mathsf{LFP}^{\rho}\, X.\alpha(X)\ \|^G \subseteq \|\ \mathsf{LFP}\, X.\alpha(X)\ \|^G$. Thus $v \in \mathcal{W}_0$ by monotonicity. □

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

Proof
We will consider only the case when $v \in V_0$. Let $(\tau_1, \ldots, \tau_n)$ be the canonical signature of v. By the definition of signature $v \in \|\ F_0(U_1, \ldots, U_n)\ \|^G$. Expanding the definition of F_0 we obtain:

$$v \in \|\ \{x : \exists y.\ E(x, y) \wedge \mathit{Next}(y, U_1, \ldots, U_n)\}\ \|^G$$

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

$$v \in \|\ \{x : E(x, y) \wedge \mathit{Next}(y, U_1, \ldots, U_n)\}\ \|^G_{[w/y]}$$

Hence $E(v, w)$ and $w \in \|\ U_i\ \|^G$ for $i = \Omega(w)$. Assume i is odd, the case when i is even is easier. By the definition

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

If τ_i is a limit ordinal then, by the definition of LFP^τ, there is a successor ordinal $\rho < \tau_i$ such that:

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

Once again referring to the definition of LFP^τ we have:

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

where $U_i' = \mathsf{LFP}^{\rho-1} Z_i. \mathsf{GFP}\, Z_{i+1} \dots \mathsf{GFP}\, Z_n. F_0(U_1, \dots, U_{i-1}, Z_i, \dots, Z_n)$. This shows that the canonical signature of w cannot be bigger than $(\tau_1, \dots, \rho - 1)$ on the first i positions, for some $\rho \leq \tau_i$. Hence $Sig(w) <_i Sig(v)$. $\square$

Definition 7 (Canonical strategy) A *canonical strategy* is a strategy taking for each node $v \in \mathcal{W}_0 \cap V_0$ a son that has the smallest canonical signature.

Remark: Despite the name, canonical strategies may not be uniquely determined because a node may have many sons with the same signature.

The forgetful determinacy theorem follows from the next lemma.

Lemma 8 The canonical strategy is a winning memoryless strategy for player 0 from every node in $\mathcal{W}_0$. From every node not in $\mathcal{W}_0$ player 1 has a memoryless winning strategy.

Proof

Suppose $v_0 \in \mathcal{W}_0$. It should be clear that the canonical strategy is memoryless. We show that it is winning for player 0. Let $\mathcal{P} = v_0, v_1, \dots$ be a history of a play when player 0 uses the canonical strategy. To arrive at a contradiction assume that $\mathcal{P}$ is winning for player 1. In other words, that the smallest priority appearing infinitely often on $\mathcal{P}$ is some odd number p.

Take an infinite sequence of positions $j_1 < j_2 < \dots$ such that: no vertex after v_{j_1} has priority smaller than p, and $\Omega(v_{j_k}) = p$ for $k = 1, \dots$ From Fact 6 we obtain that $Sig(v_{j_{k+1}}) <_p Sig(v_{j_k})$. 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

$$\begin{aligned} \neg Z &\equiv \{x : \neg Z(x)\} & \neg\{x : \psi(x)\} &\equiv \{x : \neg p(x)\} \\ \neg\, \mathsf{LFP}\, Z.F(Z) &\equiv \mathsf{GFP}\, .\neg F(\neg Z) & \neg\, \mathsf{GFP}\, Z.F(Z) &\equiv \mathsf{LFP}\, .\neg F(\neg Z) \end{aligned}$$

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

As $\mathcal{W}_0$ is defined by (2), the compliment of $\mathcal{W}_0$ is defined by:

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

Using the propositional tautology

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

we obtain

$$\begin{aligned}\neg F_0(\neg Z_1, \dots, \neg Z_n) = \big\{x : &\big(V_0(x) \Rightarrow \forall y.\ E(x,y) \Rightarrow \neg \mathit{Next}(y, \neg Z_1, \dots, \neg Z_n)\big) \\ &\big(\neg V_0(x) \Rightarrow \exists y.\ E(x,y) \wedge \neg \mathit{Next}(y, \neg Z_1, \dots, \neg Z_n)\big)\big\}\end{aligned}$$

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:

$$\begin{aligned}&\big\{x : \big(V_0(x) \Rightarrow \forall y.\ (E(x,y) \Rightarrow \mathit{Next}(y, Z_1, \dots, Z_n)\big) \\ &\quad \wedge \big(\neg V_0(x) \Rightarrow \exists y.\ E(x,y) \wedge \mathit{Next}(y, Z_1, \dots, Z_n)\big)\big\}\end{aligned}$$

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 the 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 $\mathcal{W}_0$ can be described by the formula:

$$\mathsf{LFP}\, Z_1.\, \mathsf{GFP}\, Z_2 \dots .\, \mathsf{LFP}\, Z_{n+1}.\, \mathsf{GFP}\, Z_{n+2}. F'_0(Z_1, \dots, Z_{n+2})$$

where

$$\begin{aligned}F'_0(Z_1, \dots, Z_{n+2}) = \big\{x : &\big(\neg V_0(x) \Rightarrow \exists y.\ E(x,y) \wedge \mathit{Next}(y, Z_2, \dots, Z_{n+1})\big) \\ &\wedge \big(V_0(x) \Rightarrow \forall y.\ (E(x,y) \Rightarrow \mathit{Next}(y, Z_2, \dots, Z_{n+1})\big)\big\}\end{aligned}$$

(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 $\mathcal{W}_0$. This strategy translates to a winning memoryless strategy for player 1 in G. $\square$

4 Automata on tree-like structures

In this section we will define an operation of constructing tree-like structures. We will also describe automata running on tree-like structures.

Let us fix a relational signature $Sig = \{r_1, \dots\}$. Let $Sig^+ = Sig \cup \{son, cl\}$ be the extension of Sig with one binary relation son and one unary relation cl (short for clone). D_M will always stand for the carrier of the structure M. D_M^+ will denote the set of nonempty sequences of elements of D_M. ε will stand for the empty sequence.

Definition 9 Let $M = \langle D_M, r_1, \dots\rangle$ be a structure of the signature Sig. A *tree-like structure* M^+ is a structure $M^+ = \langle D_M^+, son, cl, r_1^+, \dots\rangle$, where the relations are defined by:

$$\begin{aligned} son =& \{(w, wd) : w \in D_M^+, d \in D_M\} \\ cl =& \{wdd : w \in D_M^+ \cup \{\varepsilon\}, d \in D_M\} \\ r^+ =& \{(wd_1, \dots , wd_k) : w \in D_M^+ \cup \{\varepsilon\}, (d_1, \dots , d_k) \in r_M\} \\ & (r \in Sig,\ k\text{-array}) \end{aligned}$$

Remark: One can think of an element wd as of an element d in w copy of M. The original relations from Sig are extended so that they relate only elements in the same copies. The son relation puts a tree structure on the copies of the structure. The cl relation describes the special elements that are exactly as their fathers.

Example: Consider the structure $M = \langle\{0, 1\}, l, r\rangle$, where l holds only for 0 and r only for 1. The domain of M^+ is the set $\{0, 1\}^+$. Hence M^+ is the disjoint sum of two full binary trees, one consisting of descendants of 0 and one consisting of descendants of 1. The son relation is just the son relation over trees. The relations l^+ and r^+ determine whether a node is a left or a right son. The cl relation has rather strange meaning, it says that a node is either a left son of a left son or a right son of a right son. $\Box$

In the rest of this section we are going to define a notion of automaton running on structures of the form M^+. The problem we face here is how to describe a transition relation when the number of sons of a node depends on the input structure M and is not given in advance. Actually the sons of a node form a copy of M itself so there can be also some relations between them.

In the full binary tree each node has two successors, and we describe possible assignments of states to the successors by giving a set of pairs of states. This is just another way of describing a set of functions from the two successors to the set of states. Hence the values of a transition function are subsets of $\{0, 1\} \to Q$.

In the structure M^+ the successors of a node form a copy of the structure M. The first approximation may be that the values of a transition function can be all the subsets of $D_M \to Q$, i.e., sets of functions from the domain of M to Q. It should be clear that for some M not all such sets of functions are definable in MSOL or any other interesting logic. Hence in order to capture the expressive power of some logic we should limit the class of allowed subsets. A simple way to do this is to use some kind of logic to describe sets of functions. As in different applications we would need different expressive power, we do not fix the logic in advance. We just state some closure properties of the logic to ensure that it is expressive enough for our constructions to go through. One more abstraction step is that we may want to work not over one particular structure M but over a class of structures *Struct*.

Let $Q = \{q_1, \ldots, q_n\}$ be a set of states. For describing a set of functions of type $D_M \to Q$ we will use a formula $\varphi(Q_1, \ldots, Q_n)$ with free second order variables $Q_1, \ldots, Q_n$. Such a formula determines the set of markings $m : D_M \to Q$ such that $M, [m^{-1}(q_1)/Q_1] \ldots [m^{-1}(q_n)/Q_n] \vDash \varphi$. Hence the range of the transition function of an automaton will be some set of formulas with as many free second order variables as there are states in the automaton. We will call these formulas *basic formulas*. The set of basic formulas with n free set variables will be denoted by $\mathrm{BF}(n)$. In general $\mathrm{BF}(n)$ need not be the set of all MSOL formulas with n free variables and sometimes it pays of to work with smaller sets. On the other hand, if we want our automata to have some reasonable closure properties we need to put some restrictions on sets $\mathrm{BF}(n)$. These are given in the definition below. One more complication is that in order to represent the clone relation, basic formulas need also to have one free first order variable.

Definition 10 (Basic formulas) Let *Struct* be some class of structures of the signature *Sig*. Let $\{\mathrm{BF}(n)\}_{n \in \mathbb{N}}$ be a family of sets of formulas over *Sig*. Formulas from this family are called *basic formulas*. The only free variables in formulas from $\mathrm{BF}(n)$ can be second order variables $Q_1, \ldots, Q_n$ and an individual variable x. We require that the family of formulas $\{\mathrm{BF}(n)\}_{n \in \mathbb{N}}$ has the following closure conditions:

- C1 There must be a tautologically true formula. For every n there is $tt \in \mathrm{BF}(n)$, s.t., $\mathit{Struct} \vDash tt$.
- C2 Closure under disjunction. If there are formulas $\varphi(x, Q_1, \ldots, Q_n)$ and $\psi(x, Q_1, \ldots, Q_n)$ in $\mathrm{BF}(n)$ then there is a formula $\theta(x, Q_1, \ldots, Q_n) \in$

BF(n) s.t.:

$$Struct \models \theta(x, Q_1, \ldots, Q_n) \Leftrightarrow \big(\varphi(x, Q_1, \ldots, Q_n) \vee \psi(x, Q_1, \ldots, Q_n)\big)$$

C3 Closure under renaming. Given a 1-1 function $\rho : \{1, \ldots, m\} \to \{1, \ldots, n\}$ for some $m, n \in \mathbb{N}$ and $\varphi(x, Q_1, \ldots, Q_m) \in$ BF(m) there is $\psi(x, Q_1, \ldots, Q_n) \in$ BF(n) with

$$Struct \models \psi(x, Q_1, \ldots, Q_n) \Leftrightarrow \varphi(x, Q_{\rho(1)}, \ldots, Q_{\rho(m)})$$

C4 Closure under product. If there are $\varphi(x, P_1, \ldots, P_m) \in$ BF(m) and $\psi(x, Q_1, \ldots, Q_n) \in$ BF(n) then there is $\theta(x, R_1, \ldots, R_{mn}) \in$ BF(mn) s.t.:

$$Struct \models \theta(x, R_1, \ldots, R_{mn}) \Leftrightarrow$$
$$\big[\varphi(x, \bigcup_{j=1}^{n} R_{1,j}, \ldots, \bigcup_{j=1}^{n} R_{m,j}) \wedge \psi(x, \bigcup_{i=1}^{m} R_{i,1}, \ldots, \bigcup_{i=1}^{m} R_{i,n})\big]$$

C5 Closure under choice operation. If $\varphi(x, Q_1, \ldots, Q_n) \in$ BF(n) then there is $\overline{\varphi}(x, R_{\{1\}}, \ldots, R_{\{1,\ldots,n\}}) \in$ BF($2^n - 1$) with the property:

$$Struct \models \overline{\varphi}(x, R_{\{1\}}, \ldots, R_{\{1,\ldots,n\}}) \Leftrightarrow \big[\forall Z_1, \ldots, Z_n.\ \varphi(x, Z_1, \ldots, Z_n) \Rightarrow$$
$$\bigvee_{i=1,\ldots,n} \bigcup \{R_S : i \in S\} \cap Z_i \neq \emptyset\big]$$

In the above, we have indexed the variables R with nonempty subsets of $\{1, \ldots, n\}$ to shorten the description.

Remark: Condition (C4) allows to construct a product automaton. Condition (C5) is needed for closure under complementation. In a structure $M \in Struct$ a formula from BF(n) defines a set of tuples $R \subseteq D_M \times \mathcal{P}(D_M)^n$. Formula $\overline{\varphi}$ defines roughly the set of those tuples of sets of states that have nonempty intersection with every tuple satisfying $\varphi(x, Q_1, \ldots, Q_n)$.

Definition 11 (Initial formulas) We will also need a family of *initial formulas* $\{I(n)\}_{n\in\mathbb{N}}$. This is a family satisfying the same conditions as basic formulas do and additionally we require that the individual variable x should not appear in initial formulas. This means that initial formulas from $I(n)$ describe subsets of $\mathcal{P}(D_M)^n$ and not subsets of $D_M \times \mathcal{P}(D_M)^n$ as basic formulas do.

Remark: Please observe that we state just closure conditions without requiring that actually the set of basic formulas is nonempty. The reason is that we are interested only in closure properties of automata. Usually one would need to express some atomic relations and this would force the families $\{\mathrm{BF}(n)\}_{n\in\mathbb{N}}$, $\{I(n)\}_{n\in\mathbb{N}}$ to be nonempty.

Definition 12 (Marking) Let M be a structure with the domain D_M. A *marking* of D_M with the states from Q is a function $m : D_M \to Q$.

For a formula $\varphi \in BF(|Q|)$, marking $m : D_M \to Q$ and an element $d \in D_M$ we write $M, d, m \vDash \varphi$ for

$$M, [d/x], [m^{-1}(q_1)/Q_1], \ldots, [m^{-1}(q_n)/Q_n] \vDash \varphi(x, Q_1, \ldots, Q_n)$$

Similarly, for a formula $\iota \in I(n)$ we write $M, m \vDash \iota$.

Definition 13 (Automaton) An automaton is a tuple

$$\mathcal{A} = \langle Q, \Sigma, \iota \in I(|Q|), \delta : Q \times \Sigma \to \mathrm{BF}(|Q|), \Omega : Q \to \mathbb{N}\rangle \tag{4}$$

where $Q = \{q_1, \ldots, q_{|Q|}\}$ is a finite set of states, Σ is a finite alphabet (usually $\{0,1\}^k$), and $\mathbb{N}$ is the set of natural numbers.

Such an automaton is intended to run on structures of the form M^+. As we will be interested in characterising the expressive power of formulas with free variables, we account for this by introducing an alphabet Σ and making the automaton run not on M^+ itself, but on M^+ together with a *labelling function* $L : D_M^+ \to \Sigma$. This function describes a valuation of free variables. Given $Val : Var_2 \to \mathcal{P}(D_M^+)$ we can represent Val restricted to some finite set of variables $Z_1, \ldots, Z_k$ by a labelling $L : D_M^+ \to \mathcal{P}(\{1, \ldots, k\})$ defined by $L(w) = \{i : w \in Val(Z_i)\}$. We define accepting process of an automaton in terms of games.

Definition 14 (Acceptance) Given a structure $M = \langle D_M, r_1, \ldots\rangle$ and a labelling function $L : D_M^+ \to \Sigma$, an automaton $\mathcal{A}$ as above defines a graph of the game $G(\mathcal{A}, M, L) = \langle V_0 \cup V_1, V_0, E, \Omega_G\rangle$ in the following way:

- Positions for player 0 are $V_0 = Q \times D_M^+$.
- Positions for player 1 are $V_1 = (Q \to D_M) \times (D_M^+ \cup \{\varepsilon\})$.
- There is an edge from $(m, w) \in V_1$ to $(q, wd) \in V_0$ iff $m(d) = q$.
- There is an edge from $(q, wd) \in V_0$ to $(m, wd) \in V_1$ iff m is a marking such that $M, d, m \vDash \varphi$, where φ is the basic formula $\delta(q, L(wd))$.

- If a vertex v belongs to V_0 then it is labelled by some (q, w) and we let $\Omega_G(v) = \Omega(q)$. If $v \in V_1$ then $\Omega_G(v) = \max\{\Omega_G(q) : q \in Q\}$.

A vertex of such a game is called *initial* iff it is a vertex (m, ε) of V_1 with ε an empty sequence and m a marking such that $M, m \vDash \iota$.

We say that the structure M^+ with a labelling L is *accepted* by $\mathcal{A}$ if there is a winning strategy for player 0 from some initial vertex.

There is an alternative definition of acceptance in terms of runs, i.e., labellings of elements of D_M^+ with states. We will freely switch between the two definitions whenever convenient.

Definition 15 (Acceptance in terms of runs) Let $M = \langle D_M, r_1, \ldots \rangle$ be a structure and $\mathcal{A}$ an automaton as above. For a function $r : D_M^+ \to Q$ and an element $w \in D_M^+ \cup \{\varepsilon\}$ we let $r \downarrow w$ be a restriction of r to the elements $\{wd : d \in D_M\}$. As before, we use ε for the empty sequence.

A *run* of $\mathcal{A}$ on M^+ with a labelling $L : D_M^+ \to \Sigma$ is a function $r : D_M^+ \to Q$ such that: $M, r \downarrow \varepsilon \vDash \iota$ and for every $wd \in D_M^+$ we have $M, d, r \downarrow wd \vDash \delta(r(wd), L(wd))$.

A run is *successful* if for every infinite sequence $d_0, d_1, \ldots$ of elements of D_M the smallest number among the numbers appearing infinitely often in $\Omega(r(d_0)), \Omega(r(d_0 d_1)), \Omega(r(d_0 d_1 d_2)) \ldots$ is even.

Fact 16 There is a successful run of $\mathcal{A}$ on M^+ with a labelling L iff $\mathcal{A}$ accepts M^+ with the labelling L.

Example: Let us show in what sense automata on binary trees are a special case of our definitions. Consider an "ordinary" parity automaton on binary trees:

$$\mathcal{A}_b = \langle Q_b, \Sigma, q_b^1 \in Q_b, \delta_b : Q_b \times \Sigma \to \mathcal{P}(Q_b \times Q_b), \Omega_b : Q \to \mathbb{N} \rangle$$

Consider the structure $M = \langle \{0, 1\}, l, r \rangle$ as in the example on page 9. Recall that l and r are unary relations holding only for 0 and 1 respectively. We define a *canonical binary tree in* M^+ to be the set of all descendants of the element 0.

We would like to show how to translate $\mathcal{A}_b$ into an automaton $\mathcal{A}$ accepting M^+ with a labelling L iff the canonical tree in M^+ with L restricted to this tree is accepted by $\mathcal{A}_b$.

Automaton $\mathcal{A}$ will have the same set of states, the same alphabet, and the same indexing function Ω_b as $\mathcal{A}_b$. The initial condition ι will say that the element 0 must be labelled by q_b^1. Hence ι is $\forall x.\ l(x) \Rightarrow Q_1(x)$. For every

$q \in Q_b$ and $a \in \Sigma$ the transition $\delta_b(q,a) = \{(q_{1,l}, q_{1,r}), \ldots, (q_{1,l}, q_{1,r})\}$ of the original automaton is translated into the transition:

$$\delta(q,a) = \big(l(x) \Rightarrow \bigvee_{i=1,\ldots,n} Q_{i,l}(x) \wedge \exists y.\ x \neq y \wedge Q_{i,r}(y)\big)$$
$$\wedge \big(r(y) \Rightarrow \bigvee_{i=1,\ldots,n} Q_{i,l}(y) \wedge \exists y.\ x \neq y \wedge Q_{i,r}(x)\big)$$

Please note that if $M, d, m \vDash l(x) \wedge Q_{i,l}(x) \wedge \exists y.\ x \neq y \wedge Q_{i,r}(y)$ then we must have $m(0) = q_{i,l}$ and $m(1) = q_{i,r}$. Even though we do not require it explicitly in the formula, the meanings of x and y must be different, this is implied by the definition of formula satisfying by a marking $\square$

5 Closure properties of the automata

Let us fix a signature *Sig*, a class of structures of this signature *Struct* and sets of basic and initial formulas $\{\mathrm{BF}(n)\}_{n\in\mathbb{N}}$, $\{I(n)\}_{n\in\mathbb{N}}$. These sets of formulas define a family of automata as in (4) running on structures of the form M^+ for $M \in \mathit{Struct}$.

We say that this family of automata is *closed under sums* if for every two automata $\mathcal{A}, \mathcal{B}$ over some alphabet Σ there is an automaton $\mathcal{A}_\vee$ in this family, s.t, for every $M \in \mathit{Struct}$ and every $L : D_M^+ \to \Sigma$ we have:

> (M^+, L) is accepted by $\mathcal{A}_\vee$ iff (M^+, L) is accepted either by $\mathcal{A}$ or by $\mathcal{B}$.

Similarly, we say that this family is *closed under complementation* if for every automaton $\mathcal{A}$ there is an automaton $\overline{\mathcal{A}}$, s.t., for every $M \in \mathit{Struct}$ and every $L : D_M^+ \to \Sigma$ we have:

> (M^+, L) is accepted by $\overline{\mathcal{A}}$ iff (M^+, L) is not accepted by $\mathcal{A}$.

Finally, we say that this family is *closed under projection* if for every automaton $\mathcal{A}$ over the alphabet $\{0,1\} \times \Sigma$ there is an automaton $\mathcal{A}_\exists$ over Σ, s.t., for every $M \in \mathit{Struct}$ and every $L : D_M^+ \to \Sigma$ we have:

> (M^+, L) is accepted by $\mathcal{A}_\exists$ iff (M^+, L') accepted by $\mathcal{A}$, for some $L' : D_M^+ \to \{0,1\} \times \Sigma$ a labelling, such that, for every $w \in D_M^+$ we have $L'(w) = (i, L(w))$, for $i = 0$ or $i = 1$.

The goal of this section is the following theorem:

Theorem 17
Let Struct be a class of structures and $\{BF(n)\}_{n\in\mathbb{N}}$, $\{I(n)\}_{n\in\mathbb{N}}$ be sets of basic and initial formulas respectively. The class of languages accepted by automata as in (4) *is closed under sum, complementation and projection.*

The theorem will be proved by the following lemmas.

Lemma 18 The family of automata is closed under sum.

Proof
Let us take two automata:

$$\begin{aligned}\mathcal{A}_1 =&\langle Q_1, \Sigma, \iota_1 \in I(|Q_1|), \delta_1 : Q_1 \times \Sigma \to \mathrm{BF}(|Q_1|), \Omega_1 : Q_1 \to \mathbb{N}\rangle\\ \mathcal{A}_2 =&\langle Q_2, \Sigma, \iota_2 \in I(|Q_2|), \delta_2 : Q_2 \times \Sigma \to \mathrm{BF}(|Q_2|), \Omega_2 : Q_2 \to \mathbb{N}\rangle\end{aligned}$$

We construct an automaton

$$\mathcal{A} = \langle Q, \Sigma, \iota \in I(|Q|), \delta : Q \times \Sigma \to \mathrm{BF}(|Q|), \Omega : Q \to \mathbb{N}\rangle$$

accepting the sum of the languages accepted by $\mathcal{A}_1$ and $\mathcal{A}_2$. The idea is straightforward and is exactly the same as in the case of ordinary automata on strings or trees. The automaton $\mathcal{A}$ will contain a copy of $\mathcal{A}_1$ and a copy of $\mathcal{A}_2$. The initial condition will allow $\mathcal{A}$ to nondeterministically choose which copy to use on a given input.

The states of $\mathcal{A}$ will be $Q = Q_1 \cup Q_2$ (we may assume without a loss of generality that $Q_1 \cap Q_2 = \emptyset$). Let $n_1 = |Q_1|$, $n_2 = |Q_2|$, and $n = |Q_1| + |Q_2|$. Given a formula $\varphi(x, Q_1, \ldots, Q_{n_2}) \in \mathrm{BF}(n_2)$, we will write $\Rsh(\varphi)$ for the formula $\varphi(x, Q_{n_1+1}, \ldots, Q_{n_1+n_2})$. This is a formula of $\mathrm{BF}(n)$ by the condition (C3).

The initial condition of $\mathcal{A}$ is $\iota = \iota_1 \vee \Rsh(\iota_2)$. This is a formula in $I(n)$ because initial formulas are closed under disjunction and renaming. Indexing function $\Omega : Q_1 \cup Q_2 \to \mathbb{N}$ is defined by $\Omega(q) = \Omega_1(q)$ if $q \in Q_1$ and $\Omega(q) = \Omega_2(q)$ otherwise.

It remains to define the transition function δ. For a given state $q \in Q_1 \cup Q_2$ and a letter a we let $\delta(q, a) = \delta_1(q, a)$ if $q \in Q_1$ and $\delta(q, a) = \Rsh(\delta_2(q, a))$ if $q \in Q_2$. It is easy to see that $\delta(q, a) \in \mathrm{BF}(n)$.

It is simple to check that for every structure M and every labelling L of M^+, there is a successful run $r : D_M^+ \to Q$ of $\mathcal{A}$ iff there is a successful run of $\mathcal{A}_1$ or $\mathcal{A}_2$. $\Box$

Lemma 19 The family of automata is closed under projection.

Proof
Let us take an automaton:

$$\mathcal{A} = \langle Q, \{0,1\} \times \Sigma, \iota \in I(|Q|), \delta : Q \times (\{0,1\} \times \Sigma) \to \mathrm{BF}(|Q|), \Omega : Q \to \mathbb{N} \rangle$$

We construct the automaton:

$$\mathcal{A}_\exists = \langle \{0,1\} \times Q, \Sigma, \iota_\exists, \delta_\exists, \Omega_\exists \rangle$$

where the meanings of the components are defined below.

It is the simplest to define the priority function: $\Omega_\exists((i,q)) = \Omega(q)$.

The initial formula ι of $\mathcal{A}$, is of the form $\varphi(Q_1, \ldots, Q_n)$. We let $\iota_\exists = \varphi(Q_{(0,1)} \cup Q_{(1,1)}, \ldots, Q_{(0,n)} \cup Q_{(1,n)})$. Let us show that it is a formula of $\mathrm{BF}(2n)$. Let tt be a tautologically true formula in $\mathrm{BF}(2)$. Its existence is guaranteed by the condition (C1). By the condition (C4) there is a formula $\theta(Q_{(0,1)}, Q_{(1,1)}, \ldots, Q_{(0,n)}, Q_{(1,n)})$ equivalent to:

$$\varphi(Q_{(0,1)} \cup Q_{(1,1)}, \ldots, Q_{(0,n)} \cup Q_{(1,n)}) \wedge tt$$

This is the formula we take for $\iota_\exists$.

For every $q \in Q$ and $(i,a) \in \{0,1\} \times \Sigma$ the transition function δ gives us a formula $\delta(q,(i,a)) \in \mathrm{BF}(n)$. This formula has the form $\varphi(x, Q_1, \ldots, Q_n)$. We let

$$\delta_\exists((i,q),a) = \varphi(x, Q_{(0,1)} \cup Q_{(1,1)}, \ldots, Q_{(0,n)} \cup Q_{(1,n)})$$

This is a formula of $\mathrm{BF}(2n)$ for the similar reasons as above.

The proof that $\mathcal{A}_\exists$ is as required is similar to the case of automata on binary trees. Let us denote by $\downarrow_1$ and $\downarrow_2$ the projections onto first and second components of $\{0,1\} \times \Sigma$.

Suppose we have a structure M^+ with the carrier D_M^+. Assume that we also have a labelling function $L : D_M^+ \to \{0,1\} \times \Sigma$, and a successful run $r : D_M^+ \to Q$ of $\mathcal{A}$. We construct a successful run $r_\exists : D_M^+ \to \{0,1\} \times Q$ by letting $r_\exists(w) = (L(w){\downarrow_1}, r(w))$.

Other way around, given a structure M^+ with a labelling $L_\exists : D_M^+ \to \Sigma$ and a successful run $r_\exists : D_M^+ \to \{0,1\} \times Q$ of $\mathcal{A}_\exists$ we construct a successful run $r : D_M^+ \to Q$ of $\mathcal{A}$ and the labelling $L : D_M^+ \to \{0,1\} \times \Sigma$ by letting $r(w) = r_\exists(w){\downarrow_2}$ and $L(w) = (r(w){\downarrow_1}, L_\exists(w))$. $\square$

Lemma 20 The family of automata is closed under complementation.

Proof
Let us take an automaton:

$$\mathcal{A} = \langle Q, \Sigma, \iota \in I(|Q|), \delta : Q \times \Sigma \to \mathrm{BF}(|Q|), \Omega : Q \to \mathbb{N} \rangle$$

The automaton accepts a structure M^+ with a labelling $L : D_M^+ \to Q$ iff there exists a winning strategy for player 0 in the game $\mathcal{G}(\mathcal{A}, M, L)$. By Theorem 1 if $\mathcal{A}$ does not accept (M^+, L) then there is a winning strategy for player 1 in $\mathcal{G}(\mathcal{A}, M, L)$. Our goal is to construct an automaton $\overline{\mathcal{A}}$, s.t., for every $M \in \mathit{Struct}$ and $L : D_M^+ \to \Sigma$, there is a winning strategy for player 1 in $\mathcal{G}(\mathcal{A}, M, L)$ iff there is a winning strategy for player 0 in $\mathcal{G}(\overline{\mathcal{A}}, M, L)$.

First, let us look at different ways a memoryless strategy for player 1 can be presented. As it is defined, such a strategy is a function $\sigma_1 : V_1 \to V_0$. Recall that $V_1 = (D_M \to Q) \times (D_M^+ \cup \{\varepsilon\})$ and $V_0 = Q \times D_M^+$. Looking at the rules of the game we can see that whenever $\sigma_1(m, w) = (q, w')$ then w' is of the form wd for some $d \in D_M^+$ and $q = m(d)$. Hence σ_1 can be presented as a function $\sigma_1 : (D_M \to Q) \times (D_M^+ \cup \{\varepsilon\}) \to D_M$. We would like to present σ_1 as a function from D_M^+ to some finite set. Of course σ_1 can be presented as a function from $(D_M^+ \cup \{\varepsilon\})$ to $(D_M \to Q) \to D_M$, but unfortunately D_M may be an infinite set.

We will consider functions of type $D_M^+ \to (Q \to \mathcal{P}(Q))$. Before stating the correspondence with strategies for player 1 we need two definitions.

Definition 21 Let $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$. We will say that ρ is a *covering* of D_M^+ with respect to $\mathcal{A}$ and L if the following two conditions are satisfied:

- for every initial position $(m, \varepsilon) \in V_1$ (i.e. $m \vDash \iota$) there is a direction d with $m(d) \in \rho(d)(q)$ for some q;
- for every $(m, w) \in V_1$ with $w \neq \varepsilon$ and every q, s.t., $m \vDash \delta(q, L(w))$ there is a direction d with $m(d) \in \rho(wd)(q)$.

Definition 22 Let $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$ and let $P = (d_0, d_1, \dots)$ be an infinite sequence of elements of D_M. A *trace in ρ on P* is a sequence of states $(q_0, q_1, \dots)$, s.t., $q_{i+1} \in \rho(d_0 \dots d_i)(q_i)$ for all $i \in \mathbb{N}$. A *trace in ρ* is a trace in ρ on some P.

Claim 22.1 There is a winning strategy for player 1 in $\mathcal{G}(\mathcal{A}, M, L)$ iff there is a function $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$ covering D_M^+ with respect to $\mathcal{A}$ and L, such that, every trace in ρ is winning for player 1. Recall that a sequence of states $(q_0, q_1, \dots)$ is winning for player 1 iff the smallest number appearing infinitely often in the sequence $(\Omega(q_1), \Omega(q_2), \dots)$ is odd.

Proof

For the left to right implication, let $\sigma_1 : (D_M \to Q) \times (D_M^+ \cup \{\varepsilon\}) \to D_M$ be a winning strategy for player 1 in the game $\mathcal{G}(\mathcal{A}, M, L)$. We define the function $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$ as follows:

$$\rho(d)(q) = \{q' : \exists m.\ M, m \vDash \iota,\ \sigma_1(m, \varepsilon) = d,\ m(d) = q'\} \tag{5}$$

$$\rho(wdd')(q) = \{q' : \exists m.\ M, d, m \vDash \delta(q, L(wd)),\ \sigma_1(m, wd) = wdd',\ m(d') = q'\} \tag{6}$$

for every $q \in Q$, $d, d' \in D_M$ and $w \in D_M^+ \cup \{\varepsilon\}$.

We will show that every trace in ρ corresponds to some play in $G(\mathcal{A}, M, L)$ when player 1 uses the strategy σ_1. As σ_1 is winning for player 1 this will show that every trace in ρ is winning for player 1.

Let $(q_0, q_1, \dots)$ be a trace in ρ on some path $(d_0, d_1, \dots)$. At the beginning make player 0 choose an initial position (ε, m) such that $\sigma_1(\varepsilon, m) = d_0$ and $m(d_0) = q_1$. This is possible as $q_1 \in \rho(d_0)(q_0)$ so $q_1 \in \{q' : \exists m.\ M, m \vDash \iota,\ \sigma_1(\varepsilon, m) = d_0,\ m(d_0) = q'\}$. From (m, ε) player 1 chooses d_0 as this is suggested by the strategy σ_1.

Assume now that we are in i-th round of the play and the current position is $(q_i, d_0 \dots d_{i-1})$. Make player 0 choose a marking m, such that, $\sigma_1(m, d_0 \dots d_{i-1}) = d_i$ and $m(d_i) = q_{i+1}$. Let us check that this is possible. Denote $w = d_0 \dots d_{i-1}$. As $q_{i+1} \in \rho(wd_i)(q_i)$ we get by (6) that $q_{i+1} \in \{q' : \exists m.\ M, d_{i-1}, m \vDash \delta(q_i, L(w)),\ \sigma_1(w, m) = wd_i,\ m(d_i) = q'\}$. Hence there exists a marking m with $M, d_{i-1}, m \vDash \delta(q_i, L(w))$ such that $\sigma_1(m, w) = d_i$ and $m(d_i) = q_{i+1}$. We make player 0 choose (m, w). Player 1 will respond with the direction d_i and the new position of the play will be (q_{i+1}, wd_i).

Repeating this argument ad infinitum we obtain a play on which the sequence of visited states is exactly $(q_0, q_1, \dots)$. As, in this play, player 1 was using his winning strategy σ_1, this sequence of states must be winning for player 1.

It remains to check that ρ is covering D_M^+ with respect to $\mathcal{A}$ and L, but this follows directly from the definition of ρ.

For the right to left implication let us take a function $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$ covering D_M^+ with respect to $\mathcal{A}$ and L. Assume that all the traces in ρ are winning for player 1. We define a strategy $\sigma_1 : (D_M \to Q) \times (D_M^+ \cup \{\varepsilon\}) \to D_M$.

For an initial position $(m, \varepsilon) \in V_1$, by the definition of covering, there is a direction d with $m(d) \in \rho(d)(q)$ for some q. We put $\sigma_1(m, \varepsilon) = d$.

Now, for $wd \in D_M^+$ and m such that $M, d, m \vDash \delta(q, L(wd))$ for some q, we define $\sigma_1(m, wd)$. By definition of a covering we have a direction d' with $m(d') \in \rho(wdd')(q)$. We put $\sigma_1(m, wd) = d'$.

It is easy to check that when player 1 uses σ_1 then every play in $\mathcal{G}(\mathcal{A}, M, L)$ corresponds to some trace in ρ. Hence σ_1 is a winning strategy for payer 1. $\square$

Hence to show that there is an automaton accepting the complement of $L(\mathcal{A})$ it is enough to construct an automaton accepting (M^+, L) iff there is a covering function $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$ with all the traces winning for player 1.

We first construct an automaton $\mathcal{A}''$ accepting (M^+, L) iff there is a covering function $\rho : D_M^+ \to (Q \to \mathcal{P}(Q))$. (Actually a run of $\mathcal{A}''$ itself will be a covering function.) Next we make a product of this automaton with an automaton on words checking that every trace is winning for player 1. Define

$$\mathcal{A}'' = \langle Q'' = Q \to \mathcal{P}(Q), \Sigma, \iota'' \in I(|Q''|), \delta : Q'' \times \Sigma \to \mathrm{BF}(|Q|''), \Omega : Q'' \to \mathbb{N}\rangle$$

where we describe the meanings of the components below. For notational convenience let $Q \to \mathcal{P}(Q) = \{f_1, \dots, f_k\}$.

It is the simplest to define the priority function. As being a covering is a local property, we can just make priorities of all the sates equal to 0. Hence $\Omega''(f) = 0$ for every $f \in Q''$.

The formula $\iota \in \mathrm{BF}(|Q|)$ is of the form $\varphi(Y_1, \dots, Y_n)$. By the condition (C5) there is a formula $\iota' \in I(2^n - 1)$ with the property:

$$\begin{aligned} \mathit{Struct} \vDash \iota'(R_{\{1\}}, \dots, R_{\{1,\dots,n\}}) \Leftrightarrow \big(\forall Z_1, \dots, Z_n.\ \iota(Z_1, \dots, Z_n) \Rightarrow \\ \bigvee_{j=1,\dots,n} \bigcup \{R_S : j \in S\} \cap Z_j \neq \emptyset\big) \end{aligned}$$

To make ι' into a formula from $I(|Q''|)$ we can use the condition (C4) and construct a formula ι'' with the property:

$$\mathit{Struct} \vDash \iota''(Z_{f_1}, \dots, Z_{f_k}) \Leftrightarrow \big(\iota(\bigcup_{f(q)=\{1\}} Z_f, \dots, \bigcup_{f(q)=\{1,\dots,n\}} Z_f) \wedge tt \wedge \dots \wedge tt\big)$$

here $q \in Q$ is an arbitrary state and the conjunction $tt \wedge \dots \wedge tt$ has $n-1$ elements so that all together we have as many conjuncts as there are states in Q.

Finally we define $\delta''(f, a)$. For notational convenience denote $\delta(q, a)$ by $\varphi_{qa}(x, Y_1, \dots, Y_n)$. By the condition (C5) in $\mathrm{BF}(2^n - 1)$ there is a formula

$\varphi'_{qa}(x, R_1, \dots, R_{\{1,\dots,n\}})$ with the property:

$$Struct \vDash \varphi'_{qa}(x, R_{\{1\}}, \dots, R_{\{1,\dots,n\}}) \Leftrightarrow \big(\forall Z_1, \dots, Z_n.\ \varphi_{qa}(x, Z_1, \dots, Z_n) \Rightarrow \bigvee_{i=1,\dots,n} \bigcup \{R_S : i \in S\} \cap Z_i \neq \emptyset\big)$$

Next we use the condition (C4) and construct the formula $\varphi''_a \in \mathrm{BF}(|Q''|)$ with the property:

$$Struct \vDash \varphi''_a(x, Z_{f_1}, \dots, Z_{f_k}) \Leftrightarrow \big(\bigwedge_{q \in Q} \varphi'_{qa}(x, \bigcup_{g(q)=\{1\}} Z_g, \dots, \bigcup_{g(q)=\{1,\dots,n\}} Z_g)\big)$$

We let $\delta''(f, a) = \varphi''_a$. (Observe that δ'' does not depend on f.)

It can be checked that $\mathcal{A}$ accepts (M, L) iff there is a covering function $\rho : D^+_M \to (Q \to \mathcal{P}(Q))$. Actually a run of $\mathcal{A}$ is a covering function.

Next we define an automaton for checking traces in ρ. Let $L_t \subseteq (Q \to \mathcal{P}(Q))^\omega$ be the set of those infinite sequences $g_0, g_1, \dots$ of functions of type $Q \to \mathcal{P}(Q)$ on which all the traces are winning for player 1. It is easy to construct a Buchi word automaton recognising the complement of L_t. As deterministic parity automata on infinite words are closed under complement [7] there exists a deterministic parity automaton recognising L_t. Let us denote it by

$$\mathcal{C} = \langle Q_c, \Sigma_c = Q \to \mathcal{P}(Q), q_c, \delta_c : Q_c \times \Sigma_c \to Q_c, \Omega_c : Q_c \to \mathbb{N}\rangle$$

Finally we are ready to construct the automaton $\overline{\mathcal{A}}$ accepting the complement of $L(\mathcal{A})$.

$$\overline{\mathcal{A}} = \langle \overline{Q} = Q'' \times Q_c, \Sigma, \overline{\iota}, \overline{\delta} : \overline{Q} \times \Sigma \to \mathrm{BF}(|\overline{Q}|), \overline{W} : \overline{Q} \to \mathbb{N}\rangle$$

Once again it is the easiest to describe the function $\overline{\Omega}$. We let $\overline{\Omega}(f, q) = \Omega_c(q)$.

Let us introduce an auxiliary operation. Given a state $q \in Q_c$ and a formula $\varphi(x, Z_{f_1}, \dots, Z_{f_k}) \in \mathrm{BF}(|Q''|)$, we denote by $\varphi \upharpoonright (q)$ a formula in $\mathrm{BF}(|\overline{Q}|)$ equivalent to $\varphi(x, Z_{(f_1,q)}, \dots, Z_{(f_k,q)})$. Such a formula exists by the condition (C4).

We let $\overline{\iota} = \iota'' \upharpoonright (q_c)$ where q_c is the initial state of $\mathcal{C}$. We let $\overline{\delta}((f, q), a) = \delta''(f, a) \upharpoonright (\delta(q, f))$.

One can check that $\overline{\mathcal{A}}$ accepts (M^+, L) iff $\mathcal{A}''$ has an accepting run on (M^+, L) and every path of this run is accepted by $\mathcal{C}$. Hence $\overline{\mathcal{A}}$ accepts (M^+, L) iff there is a covering function $\rho : D^+_M \to (Q \to \mathcal{P}(Q))$ with all the traces winning for player 1. $\square$

6 Applications

6.1 Muchnik's theorem

Let $Sig = \{r_1, \dots\}$ be a signature consisting of relational symbols only and $Sig^+ = Sig \cup \{son, cl\}$ be the extended signature.

Here we will consider the case when both $\mathrm{BF}(n)$ and $I(n)$ are the set of all monadic second order formulas over Sig with free variables $x, Q_1, \dots, Q_n$ (or just $Q_1, \dots, Q_n$ in case of $I(n)$). It should be obvious that the conditions (C1)–(C5) are satisfied. It is quite easy to observe that the inclusion relation and all the relations from the extended signature are definable by automata with these families of basic and initial formulas. Hence from Theorem 17 we have:

Theorem 23
Automata as in (4)*, with $BF(n)$ and $I(n)$ being the sets of all MSOL formulas, characterise the expressive power of MSOL on tree-like structures.*

We want to prove Muchnik's theorem:

Theorem 24
For every MSOL sentence φ over the signature Sig^+ one can effectively find a sentence $\widehat{\varphi}$ over the signature Sig, s.t., for every structure M over Sig:

$$M \vDash \widehat{\varphi} \quad \textit{iff} \quad M^+ \vDash \varphi$$

Given Theorem 23, and the fact that all constructions there are effective, for the proof of this theorem it remains to show:

Lemma 25 For every automaton $\mathcal{A}$ as in (4) over one letter alphabet there is a sentence $\widehat{\varphi}$ over Sig such that for every structure M:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad M^+ \text{ is acepted by } \mathcal{A}$$

Proof
Let $\mathcal{A} = \langle Q = \{q_1, \dots, q_n\}, \iota \in I(n), \delta : Q \to \mathrm{BF}(n), \Omega : Q \to \mathbb{N}\rangle$ be an automaton as in (4) but over one letter alphabet. Here $\mathrm{BF}(n)$ and $I(n)$ are MSOL formulas over the given signature with free variables among $x, Q_1, \dots, Q_n$. Without a loss of generality we can assume that n is even and that $\Omega(q_i) = i$.

Observe that in the case of unary alphabet the labelling function does not matter and the graph of the game as described in Definition 14 can be simplified:

- Positions for player 0 are $V_0 = Q \times D_M$.
- Positions for player 1 are $V_1 = (Q \to D_M)$.
- There is an edge from $m \in V_1$ to $(q, d) \in V_0$ iff $m(d) = q$.
- There is an edge from $(q, d) \in V_0$ to $m \in V_1$ iff m is a marking such that $M, d, m \vDash \varphi$ where φ is the basic formula $\delta(q)$.
- For $(q, d) \in V_0$ we let $\Omega_G(q, d) = \Omega(q)$; for $v \in V_1$ we let $\Omega_G(v) = n$.

This game can be represented as the structure

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

where V_0^G denotes the positions for player 0; E^G describes the edge relation; P_i denotes the set of positions with priority i ($i = 1, \ldots, n$).

Let us assume for a moment that we are given $G(M)$. We are going to describe the set of winning positions $\mathcal{W}_0$ for player 0 in $G(M)$ by a MSOL formula. By Lemma 8, $\mathcal{W}_0$ is definable by a fixpoint closure of a MSOL predicate (1) (see page 4). Rewriting this predicate for the special case of our graph, where all the vertices in V_1 have the priority n, we obtain:

$$\begin{aligned} F_0(Z_1, \ldots, Z_n) = \big\{x : & \big(V_0(x) \Rightarrow \exists y.\ (E(x, y) \wedge Z_n(y))\big) \\ & \wedge \big(\neg V_0(x) \Rightarrow \forall y.\ (E(x, y) \Rightarrow \mathit{Next}(y, Z_1, \ldots, Z_n)\big)\big\} \end{aligned}$$

By Lemma 8 the set $\mathcal{W}_0$ of winning vertices for player 0 in $G(M)$ is given by:

$$\mathcal{W}_0 = \|\ \mathsf{LFP}\, Z_1 \ldots \mathsf{GFP}\, Z_n.F_0(Z_1, \ldots, Z_n)\ \|^{G(M)}$$

As we are interested in the set $\mathcal{W}_0 \cap V_0$, we can simplify the formula even further and take instead of F_0 the predicate:

$$\begin{aligned} F_0'(Z_1, \ldots, Z_n) = \big\{x : V_0(x) \wedge \exists y.\big(& E(x, y) \\ & \wedge (\forall z. E(y, z) \Rightarrow \mathit{Next}(z, Z_1, \ldots, Z_n))\big)\big\} \end{aligned}$$

It can be checked that for every meaning of $Z_1, \ldots, Z_{n-1}$ the meanings of predicates $\mathsf{GFP}\, Z_n.F_0$ and $\mathsf{GFP}\, Z_n.F_0'$ restricted to the set V_0 are the same. Hence the formula

$$\mathsf{LFP}\, Z_1 \ldots \mathsf{GFP}\, Z_n.F_0'(Z_1, \ldots, Z_n)$$

defines in $G(M)$ the set $\mathcal{W}_0 \cap V_0$, i.e., the set of those positions in V_0 from which there is a winning strategy for player 0.

Now suppose that instead of $G(M)$ we are given the structure $Q \times M = \langle Q \times D_M, Eq^{QM}, P_1^{QM}, \ldots, P_n^{QM}, r_1^{QM}, \ldots \rangle$ with

- $Eq^{QM}((q_1,d_1),(q_2,d_2))$ iff $d_1 = d_2$,
- $P_i^{QM}(q,d)$ iff $q = q_i$,
- $r_i^{QM}((q_1,d_1),\dots,(q_k,d_k))$ iff $r_i^M(d_1,\dots,d_k)$.

As the carrier of $Q \times M$ is just V_0 we can try to describe $\mathcal{W}_0 \cap V_0$ by a MSOL formula. This is possible because transitions of our automaton are expressible by MSOL formulas. Consider:

$$\begin{aligned} F_0''(Z_1,\dots,Z_n) = \big\{x : & \exists Y_1,\dots,Y_n.\ (\bigwedge_{i=1,\dots,n} Y_i \subseteq P_i) \\ & \wedge (\forall z.\ \exists! z'.\ Eq(z,z') \wedge z' \in \bigcup_{i=1,\dots,n} Y_i) \\ & \wedge \big(\bigwedge_{i=1,\dots,n} P_i(x) \Rightarrow \delta(q_i)(x,Y_1,\dots,Y_n)\big) \\ & \wedge \bigwedge_{i=1,\dots,n} Y_i \subseteq Z_i\big\} \end{aligned}$$

The first line of the predicate above says that each Y_i is a set of elements of the form (q_i,d) for some $d \in D_M$. Next line says that for every d there is a unique state q_j with $(q_j,d) \in \bigcup_{i=1,\dots,n} Y_j$. Hence $Y_1,\dots,Y_n$ define a marking $m : D_M \to Q$ by setting $m(d) = q_j$. Next line says that this marking is allowed by the transition relation of the automaton $\mathcal{A}$. Hence the first three lines correspond to $\exists y.E(x,y)$ part in the definition of the predicate F_0. One can show that for arbitrary $S_1,\dots,S_n \subseteq Q \times D_M$ we have:

$$in(\parallel F_0''(Z_1,\dots,Z_n) \parallel^{Q\times M}_{[S_1/Z_1]\dots[S_n/Z_n]}) = \parallel F_0'(Z_1,\dots,Z_n) \parallel^{G(M)}_{[in(S_1)/Z_1]\dots[in(S_n)/Z_n]}$$

where $in : Q \times D_M \to (V_0 \cup V_1)$ is the function mapping $Q \times D_M$ to V_0. Hence the predicate:

$$R = \mathsf{LFP}\, Z_1 \dots \mathsf{GFP}\, Z_n. F_0''(Z_1,\dots,Z_n)$$

describes the set $\mathcal{W}_0 \cap V_0$ in $Q \times D_M$. So M^+ is accepted by $\mathcal{A}$ iff

$$\begin{aligned} Q \times M \models \exists Y_1,\dots,Y_n.\ (\bigwedge_{i=1,\dots,n} Y_i \subseteq P_i) \wedge (\forall z.\ \exists! z'.\ Eq(z,z') \wedge z' \in \bigcup_{i=1,\dots,n} Y_i) \\ \iota(Y_1,\dots,Y_n) \wedge \bigwedge_{i=1,\dots,n} Y_i \subseteq R \end{aligned}$$

To finish the proof it is enough to observe that $Q \times M$ is definable in M by means of first order formulas. Hence for every formula φ of MSOL one can produce a formula φ' of MSOL such that for every structure M: $Q \times M \vDash \varphi$ iff $M \vDash \varphi'$.

$\square$

6.2 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). The plan for this section is as follows. First we will formally define FPL and state the result. Next we will examine to what extent we can simplify basic formulas when dealing with structures over the empty signature. These considerations will give us automata on trees of arbitrary degree capturing the power of MSOL. Finally we will show how to translate these automata into FPL formulas.

A tree can be represented as a structure $\langle T, son \rangle$ satisfying the axioms: there is unique root; from every node there is unique path from the root to the node; every element has a son. Let *Trees* stand for the class of trees.

First order logic extended in with unary fixpoint operator (FPL) is roughly MSOL, as we have defined it in Section 2, without the second order quantification. We formally define the syntax of FPL below.

Let Var_1, Var_2 be the sets of first and second order variables respectively. The sets of *predicates* and *formulas* of FPL are defined by mutual recursion as follows:

- Every $X \in Var_2$ is a predicate. If φ is a formula and $x \in Var_1$ then $\{x : \varphi\}$ is a predicate. If F is predicate with X occurring only positively in F then $\mathsf{LFP}\, X.F$ and $\mathsf{GFP}\, X.F$ are predicates.

- If $x, y \in Var_1$ then $x = y$ and $son(x, y)$ are formulas. If F is a predicate and $x \in Var_1$ then $F(x)$ is a formula. If α, β are formulas and $x \in Var_1$ then $\neg\alpha$, $\alpha \vee \beta$, $\exists x.\ \alpha$ are formulas.

The semantics of all the constructs is the same as in the case of MSOL (see Section 2). Please observe that $\mathsf{LFP}\, X.F$ is a predicate only if all occurrences of X in F are positive. From the definition it follows that FPL can be translated to MSOL. The goal of this section is to show the converse.

Theorem 26
For every MSOL formula φ there is a FPL formula ψ s.t., Trees $\vDash \varphi \Leftrightarrow \psi$.

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 essentially a forest consisting of $|M|$ full trees with nodes of degree $|M|$; we use $|M|$ to denote the cardinality of M. We would like to find a simple form of automaton equivalent to MSOL on Set^+. We could of course take as basic formulas all formulas of MSOL but then we would have problems with coding such automata in FPL.

Let $\mathrm{BF}(n)$ be the set of disjunctions of sentences of the form:

$$\begin{aligned}\exists y_1,\dots,y_k.\ \mathrm{diff}(y_1,\dots,y_k) \wedge Q_{i_1}(y_1) \wedge \dots \wedge Q_{i_k}(y_k)\\ \wedge \forall z.\ \mathrm{diff}(z,y_1,\dots,y_k) \Rightarrow (Q_{j_1}(z) \vee \dots \vee Q_{j_l}(z))\end{aligned} \quad (7)$$

where $k, l \in \mathbb{N}$, Q's are second order variables from $\{Q_1,\dots,Q_n\}$ and diff is a formula saying that the meanings of all the variables are different. In $\mathrm{BF}(n)$ there are no free first order variables. In the original definition of basic formulas we allowed a free first order variable as it is needed to define the clone relation. Here we will consider formulas using only *son* relation.

We claim that when we take the set of basic formulas as above and take the initial formulas also of this form (i.e., $I(n) = \mathrm{BF}(n)$) then automata as in (4) are as expressive over Set^+ as MSOL formulas not using the clone relation. It is easy to see that a formula of the form $son(X,Y)$ is definable by an automaton of this form. Hence it remains to show closure under sum, complementation and projection. By Theorem 17 it is enough to show that the set of basic formulas satisfies conditions (C1)–(C5). It is easy to check this for all the conditions except (C5).

Lemma 27 Let $\varphi(Q_1,\dots,Q_n) \in \mathrm{BF}(n)$ be a basic formula there is a basic formula $\overline{\varphi}(R_{\{1\}},\dots,R_{\{1,\dots,n\}})$ satisfying the condition (C5).

Proof
For a basic formula $\varphi(Q_1,\dots,Q_n) \in \mathrm{BF}(n)$ one can construct a basic formula $\overline{\varphi}(R_{\{1\}},\dots,R_{\{1,\dots,n\}}) \in \mathrm{BF}(2^n-1)$ such that for every structure $M \in Set$, valuation $Val : Var_2 \to \mathcal{P}(D_M^+)$ and valuation $\overline{Val}$, s.t., $\bigcup\{Val(R_S) : i \in S\} = D_M \setminus Val(Q_i)$ for all $i = 1,\dots,n$, we have:

$$M, Val \vDash \varphi \quad \text{iff} \quad M, \overline{Val} \nvDash \overline{\varphi} \quad (8)$$

We claim that $\overline{\varphi}$ is the formula required in the lemma. Let $U_{\{1\}},\dots,U_{\{1,\dots,n\}}$ be such that $M \vDash \overline{\varphi}(U_{\{1\}},\dots,U_{\{1,\dots,n\}})$. By (8):

$$M \nvDash \varphi((D^M \setminus S_1),\dots,(D^M \setminus S_n)) \quad (9)$$

Let $U'_1, \ldots, U'_n \subseteq D_M$ be such that $M \models \varphi(U'_1, \ldots, U'_n)$ and suppose $U'_i \cap \bigcup_{i \in S} U_i = \emptyset$ for all $i \in \{1, \ldots, n\}$. This means that $U'_i \subseteq D^M \setminus \bigcup_{i \in S} U_i$ and from the monotonicity of φ we get a contradiction with (9).

The other implication is easy. $\square$

Now, from Theorem 17 we obtain:

Theorem 28
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:

$$\langle Q, \Sigma, \iota \in I(n), \delta : Q \times \Sigma \to BF(n), \Omega : Q \to \mathbb{N} \rangle \tag{10}$$

where $\Sigma = \mathcal{P}(\{1, \ldots, k\})$ and $I(n) = BF(n)$ is the set of disjunctions of formulas of the form (7) with free variables among $Q_1 \ldots, Q_n$.

Next step is to define automata on trees of arbitrary degree. These automata have the form:

$$\langle Q, \Sigma, q_1 \in Q, \delta : Q \times \Sigma \to \mathrm{BF}(n), \Omega : Q \to \mathbb{N} \rangle \tag{11}$$

with $\mathrm{BF}(n)$ as in (7). The form of these automata is almost the same as in the previous theorem with the exception that initial formula is replaced by initial state. The definition of acceptance for these automata is very similar to Definition 14 for the general automata. The only essential difference is that now the game starts not from the initial marking but from the position consisting of the root of the given tree and the initial state.

To show that automata as in (11) capture the expressive power of MSOL we will use Theorem 28. If $M \in Set$ then M^+ consists of $|M|$ full trees of degree $|M|$. A subset $S \subseteq D^+_M$ *defines a tree in* M^+ if: (i) there is a unique element that is a son of no node; (ii) every element in S has a son in S; (iii) if $u \in S$ then the father of u is also in S. Every tree with nodes of degree bounded by $|M|$ is defined by some $S \subseteq M^+$.

Definition 29 We say that a tree T with a labelling $L : T \to \Sigma$ is *represented* in M^+ by some $S \in D^+_M$ and $L' : D^+_M \to \{0, 1\} \times \Sigma$ if S defines in M^+ a tree isomorphic to T and for every $w \in D^+_M$: $L'(w) = (0, L(w))$ if $w \notin S$ and $L'(w) = (1, L(w))$ otherwise.

Using some simple constructions on automata we obtain.

Lemma 30 Let $\mathcal{A}$ be an automaton as in (10). There is an automaton $\mathcal{A}_t$ as in (11), such that, for every tree T with a labelling $L : T \to \Sigma$, structure M^+, and labelling L' representing T in M^+ we have:

$$(T, L) \text{ is accepted by } \mathcal{A}_t \quad \text{iff} \quad (M^+, L') \text{ is accepted by } \mathcal{A}$$

Next we show how to transfer formulas over trees to formulas over Set^+.

Lemma 31 For every MSOL formula φ over the signature $\{son\}$ there is a MSOL formula $\hat{\varphi}$ over the same signature with one more free variable Z, such that, for every tree T and valuation $V : T \to \mathcal{P}(T)$:

$$T, V \vDash \varphi \qquad \text{iff} \qquad M^+, V'[S/Z] \vDash \hat{\varphi}$$

where S is a subset of D_M^+ isomorphic to T and $V' = in \circ V$ for $in : T \to D_M^+$ the embedding mapping T to S.

Proof
$\hat{\varphi}$ is obtained from φ by relativising all the quantifiers to S. □

Finally we translate automata on trees into FPL formulas. The proof of the following lemma is similar to the proof of Lemma 25.

Lemma 32 For every automaton $\mathcal{A}$ as in (11) over an alphabet $\Sigma = \{0, 1\}^k$ there is an FPL formula ψ with free variables $\{Z_1, \ldots, Z_k\}$, such that, for every tree T and a labelling $L : T \to \{0, 1\}^k$:

$$(T, L) \text{ is accepted by } \mathcal{A} \quad \text{iff} \quad T \vDash \psi(U_1, \ldots, U_k)$$

where U_i is the set of nodes that are labelled with letters having i-th component 1.

Putting these three lemmas together we have means of translating MSOL formulas to FPL formulas, hence proving Theorem 26. Given a MSOL formula φ over trees we can translate it into the formula $\hat{\varphi}$ over $Struct^+$ using Lemma 31. Next, we translate this formula into an automaton $\mathcal{A}$ using Theorem 28. Then, we translate $\mathcal{A}$ into an automaton $\mathcal{A}_t$ over trees as in Lemma 30. To complete our task it remains to use Lemma 32 and translate $\mathcal{A}_t$ into an FPL formula.

6.3 MSOL and bisimulation

The goal of this section is to show that every MSOL sentence defining bisimulation closed set of transition systems is equivalent to a μ-calculus sentence. The proof of the main lemma is omitted as it would require introducing vectorial fixpoints and going deeper into formulas describing winning sets.

Let $Pred = \{p, p', \dots\}$ be a set of unary predicate symbols and let $\{son\}$ be a binary predicate symbol. A *transition system with a source*, simply called *transition system* in the sequel, is a tuple:

$$M = \langle S^M, sr^M, son^M, \{p^M\}_{p \in Prop} \rangle$$

where: S_M is a nonempty set of *states*; $sr_M \in S$ is a *source*; and each p^M is a subset of S^M. For simplicity we have chosen to have no labels on edges. One can code them using predicate symbols or introduce them explicitly without any important change in the argument.

Transition system M is called a *transition tree* (or simply a tree) if for every state $s \in M$ there exists a unique path from the root to s, or more formally, there exists a unique sequence $s_0, \dots, s_n$ such that $s_0 = sr^M$, $s_n = s$ and for every $i = 1, \dots, n$ we have $son(s_i, s_{i+1})$.

Transition systems M and N are called *bisimilar* when there exists a relation $R \subseteq S^M \times S^N$, called a *bisimulation relation*, such that $(sr^M, sr^N) \in R$ and for every $(s,t) \in R$ and $p \in Prop$:

- $s \in p^M$ iff $t \in p^M$,
- whenever $son^M(s, s')$ for some s', then there exists t' such that $son^N(t, t')$ and $(s', t') \in R$,
- whenever $son^N(t, t')$ for some t', then there exists s' such that $son^M(s, s')$ and $(s', t') \in R$.

Definition 33 (ω-expansion) Given a transition system M, an *ω-indexed path* of M is a sequence u of the form:

$$u = s_0(a_1, s_1)(a_2, s_2) \cdots (a_n, s_n)$$

where $s_0, \dots, s_n$ is a path in M and $a_i \in \mathbb{N}$, for $i = 1, \dots, n$ are arbitrary.

The *ω-expansion* $\widehat{M}$ of the system M is defined by :

1. $S^{\widehat{M}}$ is the set of ω-indexed paths of M,
2. $sr^{\widehat{M}} = sr^M$,
3. for every every u and $v \in S^{\widehat{M}}$: $son^{\widehat{M}}(u, v)$ iff v is an ω-indexed path of the form $u(a, s)$, for some a and s,
4. for every $p \in Prop$:

$$p^{\widehat{M}} = \{u(a, s) : s \in p^M, \quad u, a \text{ arbitrary}\} \cup \{sr^{\widehat{M}} : sr^M \in p^M\}$$

Let us briefly point out how the concept of ω-expansion arises from a general consideration about bisimulation relation.

Definition 34 Given two transition systems M and N, we say that M is an *expansion* of N, denoted $M \succeq N$, when there exists a partial function $h : S_M \to S_N$ such that:

1. $h(sr^M) = sr^N$,

2. for every $s \in S^M$ and $p \in Prop$: $h(\{t : son^M(s,t)\}) = \{t : son^N(h(s),t)\}$ and $s \in p^M \Leftrightarrow h(s) \in p^N$.

In [1], with distinct notations and names, Castellani shows that M_1 and M_2 are bisimilar iff there exists N such that $N \preceq M_1$ and $N \preceq M_2$. Intuitively N is a quotient of M_1 and M_2 under bisimulation relation, henceforth a minimal representative. Next fact states that ω-expansions are, in the countable case, maximal representatives of behaviours.

Fact 35 Considering only transition systems with at most countably many states: for every transition system M we have $M \preceq \widehat{M}$, and, for every transition system N, if M and N are bisimilar then $\widehat{M}$ and $\widehat{N}$ are isomorphic.

Next we define the μ-calculus. Let Var be a set of variables. The set of the μ-calculus formulas is the smallest set containing $Prop \cup Var$ which is closed under negation, disjunction and the following two formation rules:

- if α is a formula then $\langle\ \rangle\alpha$ is a formula,
- if $\alpha(X)$ is a formula and X occurs only positively (i.e., under even number of negations) in $\alpha(X)$ then $\mu X.\alpha(X)$ is a formula.

The meaning of a formula α in a transition system M and a valuation $Val : Var \to \mathcal{P}(S^M)$ is a set of states, $\parallel \alpha \parallel^M_{Val}$, where it is true. It is defined by induction on the length of the formula:

$$\begin{array}{ll}
\parallel p \parallel^M_{Val} & = p^M \\
\parallel \neg\alpha \parallel^M_{Val} & = S^M - \parallel \alpha \parallel^M_{Val} \\
\parallel \alpha \vee \beta \parallel^M_{Val} & = \parallel \alpha \parallel^M_{Val} \cup \parallel \beta \parallel^M_{Val} \\
\parallel \langle r\rangle\alpha \parallel^M_{Val} & = \{s : \exists t.(s,t) \in r^M \wedge t \in \parallel \alpha \parallel^M_{Val}\} \\
\parallel \mu X.\alpha(X) \parallel^M_{Val} & = \bigcap\{T \subseteq S^M : \parallel \alpha(X) \parallel_{Val[T/X]} \subseteq T\}
\end{array}$$

For a sentence φ we write $M, s \models \varphi$ when $s \in \parallel \varphi \parallel^M_{Val}$ (the choice of a valuation Val is irrelevant as φ is a sentence). A sentence φ of the μ-calculus defines a class of transition systems $\{M : M, sr^M \models \varphi\}$. The class of transition systems is μ-*definable* if there exists a μ-calculus sentence defining this class. A class $\mathcal{C}$ of transition systems is *bisimulation closed* if whenever $M \in \mathcal{C}$ and M' is bisimilar to M then $M' \in \mathcal{C}$. A sentence is *bisimulation invariant* if the class of transition systems it defines is bisimulation closed.

Fact 36 Every μ-definable class is bisimulation closed.

Remark: Let us comment on the fact that we consider only definability by sentences. Call a class $\mathcal{C}$, μ-f-definable (μ-formula-definable) if there is a formula φ of the μ-calculus such that:

$$\mathcal{C} = \{M : sr^M \in \parallel \varphi \parallel^M_{Val} \text{ for arbitrary } Val : Var \to S^M\}$$

There are μ-f-definable classes that are not closed under bisimulation. Consider for example the class defined by the formula $\neg(\langle\ \rangle X \wedge \langle\ \rangle \neg X)$. This formula defines a class of structures M where there is at most one $s \in S^M$, such that, $son(sr^M, s)$. This class is clearly not bisimulation closed. The μ-f-definability corresponds to definability of frames in modal logic. It is easy to see that the notion of μ-f-definability is not closed under complement. Hence this notion of definability is not interesting from expressive completeness point of view.

We are going to compare μ-calculus definable classes to MSOL-definable classes. A sentence φ of MSOL defines a class of transition systems: $\{M : M \vDash \varphi\}$. A class of transition systems is *MSOL definable* if there exists an MSOL sentence defining this class.

Remark: There exist formulas of MSOL that are not bisimulation invariant. Take for example a formula stating that there is exactly one transition from the source. Observe that the problem of checking whether an MSOL formula is bisimulation invariant is not arithmetical because the validity problem is not arithmetical.

Next observation allows us to restrict to ω-expansions.

Fact 37 A MSOL bisimulation invariant sentence is equivalent to a μ-calculus sentence iff both sentences are satisfied in the same ω-expansions.

As ω-expansions are trees, we can use what we know about MSOL on trees. In particular for every MSOL sentence there is an equivalent automaton as in 11. Using similar arguments as in the proof of Lemma 25 we can show.

Fact 38 For every automaton $\mathcal{A}$ as in (11) over the alphabet $\{1, \ldots, k\}$ there is a μ-calculus sentence such that whenever $M = \{S^M, \varepsilon, son, p_1, \ldots, p_k\}$ is an ω-expansion of a transition system:

$$T \in L(\mathcal{A}) \quad \text{iff} \quad M \vDash \alpha$$

where $T : S^M \to \mathcal{P}(\{1, \ldots, k\})$ is defined by $T(s) = \{i : M \vDash P_i\}$.

As a corollary we get:

Theorem 39
Every MSOL sentence defining bisimulation closed class of structures is equivalent to a μ-calculus sentence.

7 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 the notion of basic formulas. We have given some conditions on basic formulas which guarantee that automata are closed under disjunction, negation and existential quantification. This parametrisation is useful as we were able to get different results using different sets of basic formulas. In all these cases basic formulas were some special MSOL formulas. One can also consider families of basic formulas bigger than MSOL formulas, as for example MSOL with counting modulo predicates. In this case our closure theorem gives a characterisation of MSOL with counting. It is an interesting question whether one can formulate and prove some counterpart of Theorem 26 in this case. Conditions (C1)–(C5) also deserve some discussion. These conditions were chosen because they make the proof go through but maybe there is some good reason for weakening them?

References

[1] I. Castellani. Bisimulations and abstraction homomorphisms. *Journal of Computer and System Sciences*, 34:210–235, 1987.

[2] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS 91*, 1991.

[3] D. Janin and I. Walukiewicz. Automata for the μ-calculus and related results. In *MFCS '95*, volume 969 of *LNCS*, pages 552–562, 1995.

[4] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR'96*, volume 1119 of *LNCS*, pages 263–277, 1996.

[5] N. Klarund. Progress measures, immediate determinacy and a subset construction for tree automata. In *LICS '92*, pages 382–393, 1992.

[6] R. McNaughton. Infinite games played on finite graphs. *Ann. Pure and Applied Logic*, 65:149–184, 1993.

[7] A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In A. Skowron, editor, *Fifth Symposium on Computation Theory*, volume 208 of *LNCS*, pages 157–168, 1984.

[8] A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdansk, 1991.

[9] A. Semenov. Decidability of monadic theories. In *MFCS'84*, volume 176 of *LNCS*, pages 162–175. Springer-Verlag, 1984.

[10] R. S. Streett and E. A. Emerson. An automata theoretic procedure for the propositional mu-calculus. *Information and Computation*, 81:249–264, 1989.

[11] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer-Verlag, 1997.

[12] I. Walukiewicz. Monadic second order logic on tree-like structures. In *STACS '96*, volume 1046 of *LNCS*, pages 401–414, 1996.

[13] W. Zielonka. Infinite games on fintely coloured graphs with applications to automata on infinte trees. To appear in TCS, 1997.