Theoretical Computer Science 418 (2012) 48–70

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Regular sets over extended tree structures

Séverine Fratani *

Laboratoire d'Informatique Fondamentale de Marseille (LIF), UMR6166 CNRS – Université de la Méditerranée – Université de Provence, 13453 Marseille, France

ARTICLE INFO

Article history:
Received 6 July 2005
Received in revised form 21 October 2011
Accepted 23 October 2011
Communicated by Z. Esik

Keywords:
Labeled tree structures
MSO definable sets
Automata with oracle
Iterated pushdown structures

ABSTRACT

We investigate notions of decidability and definability for the Monadic Second-Order Logic over labeled tree structures, and its relations with finite automata using oracles to test input prefixes.

A general framework is defined allowing to transfer some MSO-properties from a graph-structure to a labeled tree structure. Transferred properties are the decidability of sentences and the existence of a definable model for every satisfiable formula. A class of finite automata with prefix-oracles is also defined, recognizing exactly languages defined by MSO-formulas in any labeled tree-structure.

Applying these results, the well-known equivalence between languages recognized by finite automata, sets of vertices MSO definable in a tree-structure and sets of pushdown contexts generated by pushdown-automata is extended to k-iterated pushdown automata.

0. Introduction

Initiated by the work of Büchi on words, the study of links between automata and logic has enabled to identify structures having a decidable Monadic Second-Order theory (see for example [1,2] for a survey). In particular, Rabin proved in [3] decidability of the MSO-theory of infinite tree structures in which numerous properties are definable and theories are interpretable. These works have also led to a logic characterization of regular languages: languages recognized by finite automata are exactly sets defined by MSO-formulas in a tree structure.

The goal of this paper is to extend these works to labeled tree structures: we exhibit labeled tree structures having a decidable MSO-theory, for which every satisfiable MSO-formula admits a MSO-definable model, and for which we can provide an automata-characterization of the definable sets.

To achieve this goal, we introduce new theoretical objects. We define a class of word/tree automata with prefix-oracles (i.e., sets of words over the input alphabet) used to test the already processed prefixes of inputs. Forests recognized by prefix-oracles automata possess useful properties, in particular, Rabin's correspondence between regular forests and models of MSO-formulas over infinite trees can be extended to these languages: forests recognized by automata with oracles $O_1, \ldots, O_m$ are forests MSO-definable in a tree structure extended by unary relations $O_1, \ldots, O_m$.

We establish MSO properties transfer theorems, from a graph structure, toward a tree structure. This approach is common for transfer reducing decidability (for example, the reduction of decidability of the MSO-theory from a structure to its *tree-like structure*, (see [4] or [5]), or from a graph to its unfolding (see [6])). We give a reduction which allows to obtain new decidability results which are not covered by the ones cited above. In addition, we give transfer theorems that apply to *MSO definable* sets in tree structures and to classes of automata *recognizing* them. In particular, we are interested in the *selection property*. This property ensures, for a structure $\mathcal{S}$, that any satisfiable formula admits at least one model

* Tel.: +33 5 40 00 69 37; fax: +33 5 40 00 66 69.
E-mail address: severine.fratani@lif.univ-mrs.fr.

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.10.020

MSO-definable in $\mathcal{S}$ (see Definition 32). Some important structures satisfy the selection property [7–9]. Rabinovich has recently intensively studied this property [10–13] and proved, in particular, that for every monadic relations $N_1, \ldots, N_m \subseteq \mathbb{N}$, the structure $\langle \mathbb{N}, +1, N_1, \ldots, N_m \rangle$ has the selection property [10].

Here, properties are transferred to a labeled tree structure from its *image structure* by any morphism. If $\mu : D \to D'$ is a surjective morphism and $\mathbb{S}$ is a relational structure over D, the *image structure* $\mu(\mathbb{S})$ of $\mathbb{S}$ has D' as domain and its relations are the images by μ of the relations of $\mathbb{S}$.

Let t be a labeled tree, and $\underline{t}$ be the structure associated with t. For any monoid morphism μ, and under some simple hypothesis on the labeling of t, we obtain the following main results:

- *Transfer of decidability:* (Theorem 53) if $\mu(\underline{t})$ has a decidable MSO-theory, then $\underline{t}$ has a decidable MSO-theory,
- *Transfer of the selection property:* (Theorem 55) under some condition on μ, if $\mu(\underline{t})$ satisfies the selection property, then $\underline{t}$ satisfies it too.
- *Theorem of structure:* (Theorem 56) under the same condition as above on μ, if $\mu(\underline{t})$ satisfies the selection property, then any set is MSO-definable in $\underline{t}$ iff it is recognized by a finite automaton using only oracles of the form $\mu^{-1}(D)$ where D is MSO-definable in $\mu(\underline{t})$. (Then each oracle tests a property MSO-definable in $\mu(\underline{t})$, on the image by μ of input word prefixes).

Applying these results, we obtain tree structures having a decidable MSO theory and classes of languages having two equivalent characterizations: one as languages recognized by automata with oracles, and the other one as sets MSO-definable in some labeled tree structures. We thus extend the two characterizations of regular languages mentioned above.

To complete this extension of regular languages, we have to take into account a third characterization: regular languages are exactly sets of pushdown contexts generated by a pushdown system of transitions [14]. This is done in the last part of the paper in which we consider the stack languages generated by "iterated pushdown automaton", which are automata whose memory is roughly a stack of stack ... of stack (see for examples [15–19]). We define a notion of "regular" sets of k-pushdowns (i.e., stacks with k level of embedded pushdowns) which generalize the notion of regular set of words.

This paper is organized as follows. Section 1 is devoted to basic definitions on words, logic, automata, as well as word automata with oracles. In Section 2, we extend the use of oracles to tree automata. Rabin's correspondence between regular forests and models of MSO-formulas over trees is adapted to these languages. In Section 3, we develop a game-theoretical approach to prove the three transfer theorems. We give also a simple application of these transfer theorems. Finally, in Section 4, we extent the notion of regular sets to sets of iterated pushdown stores.

1. Preliminaries

1.1. Basic definitions

Some notations and conventions. We denote by the $k \uparrow n$ the map defined by: $0 \uparrow n = n$ and $(k+1) \uparrow n = 2^{k \uparrow n}$.

Let S be a set. If S is finite, we denote by $|S|$ the cardinal of S. If μ is a map from S, then $\mu(S) = \{\mu(\sigma) \mid \sigma \in S\}$. If $\vec{V} = (V_1, \ldots, V_n)$ is a vector of subsets of S then $\mu(\vec{V}) = (\mu(V_1), \ldots, \mu(V_n))$.

The characteristic function of $\vec{V}$ in S is a map $\chi_S^{\vec{V}} : S \to \{0,1\}^n$ defined for all $\sigma \in S$, by $\chi_S^{\vec{V}}(\sigma) = (b_1, \ldots, b_n)$ where $b_i = 1$ iff $\sigma \in V_i$.

Words and languages. If A is a finite set, A^* denotes the set of words (finite sequences) over A, and ε the empty word. For $u, v \in A^*$, the length of u is denoted $|u|$ and we write $v \preccurlyeq u$ if v is a prefix of u, i.e. if there exists $w \in A^*$ such that $u = vw$. A language $P \subseteq A^*$ is prefix closed if: $\forall u \in P, \forall v \in A^*$, if $v \preccurlyeq u$ then $v \in P$.

Projections. For any integers $0 < i \leq j \leq n$, for any vector of elements $(a_1, \ldots, a_n)$, we define the projections $\pi_i(a_1, \ldots, a_n) = a_i$ and $\pi_{i,j}(a_1, \ldots, a_n) = (a_i, \ldots, a_j)$. For any alphabets B and A with $B \subseteq A$, the projection $\pi_B : A^* \to B^*$ is a morphism defined by $\pi_B(a) = a$ if $a \in B$ and $\pi_B(a) = \varepsilon$ else.

Trees and forests. Given finite alphabets Σ and A and a prefix closed language $P \subseteq A^*$, a P-tree(Σ) (tree of domain P labeled by Σ) is a total function $t : P \to \Sigma$. The set of all P-tree(Σ) is denoted P-Tree(Σ). In order to deal with unlabeled trees in an uniform way, we introduce the special symbol $\top$. The unique unlabeled P-tree is then the constant map $t : P \to \{\top\}$. We will often consider trees in P-Tree($\{0,1\}^n$), for $n \geq 0$ (with the convention that $\{0,1\}^0 = \{\top\}$), and we will denote this class P-Tree$_n$. Remark that a tree $t \in P$-Tree$_n$ can always be seen as the characteristic function $\chi_P^{\vec{S}}$ of the vector $\vec{S} = (S_1, \ldots, S_n)$, where $S_i = \{u \in P \mid \pi_i(t(u) = 1\}$.

We will use two kinds of operations on trees and tree-languages:

- Restriction: let $t \in A^*$-Tree(Σ), $t_{|P}$ is the P-tree(Σ) obtained by restricting the domain of t to P. If $F \subseteq A^*$-Tree(Σ), then $F_{|P} = \{t_{|P},\ t \in F\}$.
- Product: let t_1 be a P-tree(Σ_1) and t_2 a P-tree(Σ_2), the product of t_1 and t_2 is the tree $t_1 \otimes t_2 \in P$-Tree($\Sigma_1 \times \Sigma_2$) fulfilling $\forall u \in P, t_1 \otimes t_2(u) = (t_1(u), t_2(u))$. This definition can be extended to tree languages:
  if $F_1,\ F_2 \subseteq P$-Tree(Σ), then $F_1 \otimes F_2 = \{t_1 \otimes t_2 |\ t_1 \in F_1,\ t_2 \in F_2\}$.

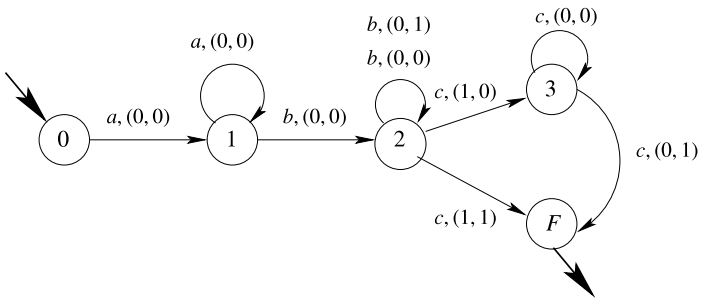

Fig. 1. A 2-automaton with $O_1 = \{a^n b^n\}_{n\geq 1}$, $O_2 = \{a^m b^n c^{n-1}\}_{n\geq 1,\, m\geq 0}$.

1.2. Finite automata with prefix-oracle

Finite automata with prefix-oracle (or p-oracle) extend the class of finite automata by allowing some membership tests on input words prefixes. Given an input alphabet A, an automaton $\mathcal{A}$ with p-oracles over A, is a finite automaton associated to a vector $\vec{O} = (O_1, \ldots, O_m)$ of languages in A^* and whose transitions contain a boolean vector of size m called **test**. During the computation by $\mathcal{A}$ of an input word, the part u of the input that is already processed is kept in memory and a transition with test $\vec{o}$ can be applied if $\vec{o}$ is equal to the characteristic vector of u inside $\vec{O}$ (i.e., if $\vec{o} = \chi^{\vec{O}}_{A^*}(u)$).

Remark that this approach has already been devised in [20] to characterize some proper subclasses of regular languages by using regular prefix-oracles, and to study their definability in First-Order Logic over extended word structures. However, the definition of automata with prefix-oracles does not explicitly appear in this paper since regular prefix-oracles can be simulated by the synchronized product of finite automata.

Definition 1 (*Finite Automaton with p-Oracles*)**.** Given $m \geq 1$, an *automaton with m p-oracles* (or m-automaton) is a tuple $\mathcal{A} = (Q, A, \vec{O}, \Delta, q_0, F)$ where Q is a finite set of states, A is the input alphabet, $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, $\Delta \subseteq Q \times A \times \{0,1\}^m \times Q$ is the set of transitions, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states.
A configuration of $\mathcal{A}$ is a pair $(q, u_\uparrow v)$ where $uv \in A^*$ and $_\uparrow$ is a symbol which does not belong to A. The binary relation on configurations is $\to_{\mathcal{A}}$ and consists in all pairs $(p, u_\uparrow av) \to_{\mathcal{A}} (q, ua_\uparrow v)$ such that $(p, a, \chi^{\vec{O}}_{A^*}(u), q) \in \Delta$. We denote by $\to^*_{\mathcal{A}}$ the reflexive, transitive closure of $\to_{\mathcal{A}}$. The language recognized by $\mathcal{A}$ is $\mathrm{L}(\mathcal{A}) = \{u \in A^* \mid \exists q_F \in F,\ (q_0, {}_\uparrow u) \to^*_{\mathcal{A}} (q_F, u_\uparrow)\}$.

We will use the following notations: $\mathsf{FA}^{\vec{O}}(A)$ is the family of automata over A with p-oracle $\vec{O}$ and $\mathsf{REG}^{\vec{O}}(A)$ is the class of $\vec{O}$-regular languages (i.e., recognized by automata in $\mathsf{FA}^{\vec{O}}(A)$). Remark that an automaton with oracle $\emptyset$ is simply a finite automaton. We write then $\mathsf{FA}(A)$ rather than $\mathsf{FA}^{\emptyset}(A)$ and $\mathsf{REG}(A)$ instead of $\mathsf{REG}^{\emptyset}(A)$.

Definition 2. An m-automaton $\mathcal{A} = (Q, A, \vec{O}, \Delta, q_0, F)$ is said to be *deterministic* if $\forall p \in Q,\ a \in A,\ \vec{o} \in \{0,1\}^m$, there is at most one $q \in Q$ such that $(p, a, \vec{o}, q) \in \Delta$. It is said to be *complete* if $\forall p \in Q,\ a \in A,\ \vec{o} \in \{0,1\}^m$, there exists $q \in Q$ such that $(p, a, \vec{o}, q) \in \Delta$.

Example 3. The automaton depicted Fig. 1 is deterministic and recognize the language $\{a^n b^n c^n\}_{n\geq 1}$. Let us describes two computation of this automaton:

1. $(q_0, {}_\uparrow abc) \to^2 (q_2, ab_\uparrow c)$ now, $ab \in O_1$ and $ab \in O_2$, that is, $\chi^{\vec{O}}_{\{a,b,c\}^*}(ab) = (1,1)$, thus we get $(q_2, ab_\uparrow c) \to (q_F,\ abc_\uparrow)$.
2. Let $n > 1$, we have $(q_0, {}_\uparrow a^n b^n c^n) \to^{n+1} (q_2,\ a^n b_\uparrow b^{n-1} c^n)$. Now, the transition $(q_2, c, (1,1), q_F)$ will never be applied since too many a were read. Then we must remain in state q_2 and read b until have processed a word in O_1, that is , until have processed the word $a^n b^n$.

   Then $(q_0, {}_\uparrow a^n b^n c^n) \to^{2n+1} (q_3,\ a^n b^n c_\uparrow c^{n-1})$. Now we must remain in q_3 and add c until have processed $a^n b^n c^{n-1}$. Finally, we get $(q_0, {}_\uparrow a^n b^n c^n) \to^* (q_F,\ a^n b^n c^n_\uparrow)$.

We associate to each m-automaton $\mathcal{A} \in \mathsf{FA}^{\vec{O}}(A)$, a finite automaton $\widetilde{\mathcal{A}} \in \mathsf{FA}(A \times \{0,1\}^m)$ called **source** of $\mathcal{A}$ and constructed by moving the test of each transition into the input letter of the transition: each transition $(p, a, \vec{o}, q)$ is transformed in $(p, (a, \vec{o}), q)$. The language $\mathrm{L}(\mathcal{A})$ can then be obtained from the language $\mathrm{L}(\widetilde{\mathcal{A}})$ and the "characteristic language of $\vec{O}$ in A" which is the language A^* annotated by the characteristic vectors of all prefixes of words.

Definition 4. For every $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, the *characteristic language* of $\vec{O}$ is defined by:
$\mathrm{L}^{\vec{O}}_\chi = \{(a_1, \vec{o}_1) \ldots (a_n, \vec{o}_n) \in (A \times \{0,1\}^m)^* \mid \forall i \in [1, n],\ \vec{o}_i = \chi^{\vec{O}}_{A^*}(a_1 \ldots a_{i-1})\}$.

Observation 5. For every $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$: $\mathrm{L}(\mathcal{A}^{\vec{O}}) = \pi_1(\mathrm{L}(\widetilde{\mathcal{A}}) \cap \mathrm{L}^{\vec{O}}_\chi)$.

Using Kleene's theorem [21], and Observation 5, we obtain easily:

Theorem 6. *Let A an alphabet, and $\vec{O}$ a vector of subsets of A^*,*

1. $\mathsf{REG}^{\vec{O}}(A)$ *is the class of languages recognized by deterministic automata in* $\mathsf{FA}^{\vec{O}}(A)$,
2. $\mathsf{REG}^{\vec{O}}(A)$ *is closed under boolean operations.*

1.3. Monadic Second-Order Logic

Let $Sig = \{r_1, \ldots, r_n\}$ be a signature containing only relational symbols, where r_i is of arity $\rho_i \in \mathbb{N}$. A (relational) **structure** $\mathcal{S}$ over the signature Sig consists of a domain $D_{\mathcal{S}}$ and relations $r_1, \ldots, r_n$ on $D_{\mathcal{S}}$ where r_i is of arity of ρ_i.

Let Sig be a signature and $Var = \{x, y, z, \ldots, X, Y, Z \ldots\}$ be a set of variables, where $x, y, \ldots$ denote first-order variables and $X, Y, \ldots$ second-order variables. The set MSO(Sig) of MSO-formulas over Sig is the smallest set such that:

- $x \in X$ and $Y \subseteq X$ are MSO-formulas for every $x, Y, X \in Var$,
- $r(x_1, \ldots x_\rho)$ is an MSO-formula for every $r \in Sig$, of arity ρ and every first order variables $x_1, \ldots x_\rho \in Var$,
- if ϕ, ψ are MSO-formulas then $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\forall x.\phi$, $\forall X.\phi$, $\exists x.\phi$ and $\exists X.\phi$ are MSO-formulas, for $x, X \in Var$

Let $\mathcal{S} = \langle D_{\mathcal{S}}, r_1, \ldots, r_n\rangle$ be a structure over the signature Sig, a valuation of Var over $D_{\mathcal{S}}$ is a function $val : Var \to D_{\mathcal{S}} \cup \mathcal{P}(D_{\mathcal{S}})$ such that for every $x, X \in Var$, $val(x) \in D_{\mathcal{S}}$ and $val(X) \subseteq D_{\mathcal{S}}$.
The satisfaction $\mathcal{S}, val \models \phi$ of an MSO-formula ϕ in the structure $\mathcal{S}$ for valuation val is then defined by induction on the structure of the formula, in the usual way.
An MSO-formula $\phi(\bar{x}, \bar{X})$ (where $\bar{x} = (x_1, \ldots, x_\rho)$ and $\bar{X} = (X_1, \ldots, X_\tau)$ denotes free first and second-order variables of ϕ) over Sig is said to be **satisfiable in** $\mathcal{S}$ if there exists a valuation val such that $\mathcal{S}, val \models \phi(\bar{x}, \bar{X})$.
We will often abbreviate $\mathcal{S}, [\bar{x} \mapsto \bar{a}, \bar{X} \mapsto \bar{A}] \models \phi(\bar{x}, \bar{X})$ by $\mathcal{S} \models \phi(\bar{a}, \bar{A})$.

Definition 7. A structure $\mathcal{S}$ admits a decidable MSO-theory if for every MSO-sentence ϕ (i.e., a MSO-formula without free variables) one can effectively decide whether $\mathcal{S} \models \phi$.

A vector $\vec{D} = (D_1, \ldots, D_m)$ of subsets of $D_{\mathcal{S}}$ is said to be **MSO-definable** in $\mathcal{S}$ iff there exists $\phi(X_1, \ldots, X_m)$ in MSO(Sig) such that:

- $\mathcal{S} \models \phi(D_1, \ldots, D_m)$ and
- $\forall \vec{S} = (S_1, \ldots, S_m), S_i \subseteq D_{\mathcal{S}}$, if $\mathcal{S} \models \phi(S_1, \ldots, S_m)$ then $\vec{S} = \vec{D}$.

Remark that $\vec{D}$ is MSO-definable in $\mathcal{S}$ iff each D_i is MSO-definable in $\mathcal{S}$.

Definition 8 (*Interpretations*)**.** Let $\mathcal{S}$ (resp. $\mathcal{S}'$) be a structure defined over the signature $Sig = \{r_1, \ldots, r_n\}$ (resp. $Sig' = \{r'_1, \ldots, r'_m\}$). An MSO-*interpretation* of the structure $\mathcal{S}$ into the structure $\mathcal{S}'$ is an injective map $f : D_{\mathcal{S}} \to D_{\mathcal{S}'}$ such that,

1. $f(D_{\mathcal{S}})$ is MSO-definable in $\mathcal{S}'$
2. $\forall i \in [1, n]$, there exists $\phi'_i(\bar{x}) \in \mathrm{MSO}(Sig')$, (where $\bar{x} = x_1, \ldots, x_{\rho_i}$) fulfilling that, for every valuation val of Var in $D_{\mathcal{S}}$

$$(\mathcal{S}, val) \models r_i(\bar{x}) \Leftrightarrow (\mathcal{S}', f \circ val) \models \phi'_i(\bar{x}).$$

Theorem 9 (*[22]*)**.** *Suppose there exists a computable* MSO-**interpretation** *of the structure $\mathcal{S}$ into the structure $\mathcal{S}'$. If $\mathcal{S}'$ has a decidable* MSO-*theory, then $\mathcal{S}$ has a decidable* MSO-*theory too.*

Definition 10. If there exists a computable MSO-interpretation of $\mathcal{S}$ into $\mathcal{S}'$, and there exists a computable MSO-interpretation of $\mathcal{S}'$ into $\mathcal{S}$, then we say that $\mathcal{S}$ and $\mathcal{S}'$ are *MSO-equivalent*.

2. Monadic Second-Order Logic and regular tree languages

2.1. Tree automata

We define here tree automata with p-oracle extending finite tree automata by allowing membership tests on nodes of input trees. For a given oracle $\vec{O}$, application of any transition to a node u of a tree depends on the characteristic vector of u in $\vec{O}$.

Definition 11 (*Tree Automata with Oracles*)**.** Let $m \geq 1$, a tree automaton with m oracles (or m-tree-automaton) is a structure $\mathcal{A} = (Q, \Sigma, A, \vec{O}, \Delta, q_0, c)$ where Q is finite set of states, Σ is a finite alphabets, $A = \{a_1, \ldots, a_n\}$ is a finite alphabet, $\vec{O}$ is a vector of m languages in A^*, $q_0 \in Q$ is an initial state, $c : Q \to [0, n_c]$, $n_c \geq 0$ is the coloring function which assigns an index value out of a finite index set to each state of the automaton and $\Delta \subseteq Q \times \Sigma \times \{0, 1\}^m \times Q^n$ is the transition relation.

Given $t \in A^*\text{-Tree}(\Sigma)$, a *run* of $\mathcal{A}$ over t is a tree $r \in A^*\text{-Tree}(Q)$ fulfilling:

$$r(\varepsilon) = q_0 \quad \text{and} \quad \forall u \in A^*, (r(u), t(u), \chi^{\vec{O}}_{A^*}(u), r(ua_1), \ldots, r(ua_n)) \in \Delta.$$

A run r is successful if for every infinite path $\pi = q_1 \cdots q_n \cdots$ in r, the smallest $i \in [0, n_c]$ appearing infinitely often in the sequence $c(q_1), \ldots, c(q_n), \ldots$ is even. The tree language recognized by $\mathcal{A}$ is denoted $\mathrm{F}(\mathcal{A})$ and refers to the set of trees for which there exists a successful run.

The class of A^*-tree(Σ) automata with oracle $\vec{O}$ is denoted $\mathsf{TFA}^{\vec{O}}(A, \Sigma)$. When $A = \{0, 1\}$ we will simply write $\mathsf{TFA}^{\vec{O}}(\Sigma)$. The class of tree languages recognized by automata in $\mathsf{TFA}^{\vec{O}}(A, \Sigma)$ is $\mathsf{TREG}^{\vec{O}}(A, \Sigma)$ (or $\mathsf{TREG}^{\vec{O}}(\Sigma)$ when $A = \{0, 1\}$), these languages are called $\vec{O}$-regular. Remark that a tree automaton with oracle $\emptyset$ is simply a parity tree automaton (see [23]. We write then $\mathsf{TFA}(A, \Sigma)$ rather than $\mathsf{TFA}^{\emptyset}(A, \Sigma)$ and $\mathsf{TREG}(A, \Sigma)$ rather than $\mathsf{TREG}^{\emptyset}(A, \Sigma)$.

Example 12. Let $A = \{a, b\}$ and $O = \{u \in A^* \mid |u|_b \text{ is prime}\}$ be the set of words containing an prime number of occurrences of b and $\mathcal{A} = (\{q_0\}, \{\alpha, \beta\}A, O, \Delta, c : q_0 \mapsto 0)$, where $\Delta = \{\delta_o = (q_0, \alpha, 1, q_0), \delta_e = (q_0, \beta, 0, q_0, q_0), \delta'_e = (q_0, \alpha, 0, q_0, q_0)\}$. Clearly, $\mathrm{F}(\mathcal{A})$ is the set of all A^*-trees t such that if $|u|_b$ is prime, then $t(u) = \alpha$.

Definition 13 (*Characteristic Forest of* $\vec{O}$)**.** Given $\vec{O} = (O_1, \ldots, O_m)$, with $O_i \subseteq A^*$, the characteristic forest of $\vec{O}$ over Σ is $\mathrm{F}^{\vec{O}}_{\chi}(\Sigma) = A^*\text{-Tree}(\Sigma) \otimes \{\chi^{\vec{O}}_{A^*}\}$.

Example 14. Consider O as defined in Example 12, then $\mathrm{F}^{\vec{O}}_{\chi}(\Sigma)$ is the set of trees t such that for all $u \in A*$, $t(u) = (\alpha, b)$, $\alpha \in \Sigma$ and $b = \chi^{\vec{O}}_{A^*}(u)$ (i.e., $b = 1$ iff $|u|_b$ is prime).

Let us map each m-tree-automaton $\mathcal{A} = (Q, \Sigma, A, \vec{O}, \Delta, q_0, c) \in \mathsf{TFA}^{\vec{O}}(A, \Sigma)$ to the tree automaton $\widetilde{\mathcal{A}} = (Q, \Sigma \times \{0, 1\}^m, A, \widetilde{\Delta}, q_0, c) \in \mathsf{TFA}(A, \Sigma \times \{0, 1\}^m)$ where $\widetilde{\Delta}$ consists of every transition $(q, (\alpha, \vec{o}), p_1, \ldots, p_n)$ such that $(q, \alpha, \vec{o}, p_1, \ldots, p_n) \in \Delta$. It can easily be checked that $\mathrm{F}(\mathcal{A}) = \pi_1(\mathrm{F}(\widetilde{\mathcal{A}}) \cap \mathrm{F}^{\vec{O}}_{\chi}(\Sigma))$.

Observation 15. For every $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$,

$$\mathsf{TREG}^{\vec{O}}(A, \Sigma) = \{\pi_1(F \cap \mathrm{F}^{\vec{O}}_{\chi}(\Sigma)) \mid F \in \mathsf{TREG}(A, \Sigma \times \{0, 1\}^m)\}.$$

Remark 16. It can be easily seen that $\{\chi^{\vec{O}}_{A^*}\}$ and $\mathrm{F}^{\vec{O}}_{\chi}(\Sigma)$ are $\vec{O}$-regular tree languages.

It is well known (see for example [3,23]) that $\mathsf{TREG}(A, \Sigma)$ is closed under union, intersection, complementation and product. Then, we obtain from Observation 15:

Theorem 17. *The class* $\mathsf{TREG}^{\vec{O}}(A, \Sigma)$ *is closed under boolean operations. Furthermore,* $\mathsf{TREG}^{\vec{O}}(A, \Sigma_1 \times \Sigma_2)$ *is the class of all* $F_1 \otimes F_2$*, such that* $F_1 \in \mathsf{TREG}^{\vec{O}}(A, \Sigma_1)$ *and* $F_2 \in \mathsf{TREG}^{\vec{O}}(A, \Sigma_2)$.

Given $P \subseteq A^*$ a prefix closed language, we define now automata "tagging" P and for which the success of a run depends only on nodes in P. They will by used to recognize P-trees.

Definition 18 (*P-cut Automaton*)**.** An automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A, \Sigma)$ is called *P-cut* if there exists a special state $q_\perp \in Q$ such that $c(q_\perp) = 0$ and $\forall t \in A^*\text{-Tree}(\Sigma)$, $r \in A^*\text{-Tree}(Q)$ run of $\mathcal{A}$ over t, for every $u \in A^*$:

$$u \notin P \text{ iff } r(u) = q_\perp.$$

In this case, for every run r, nodes external to P are colored by 0, then the success of r depends only on infinite paths inside P (hence if P is finite, any run is always successful).

In the rest of the paper, $\mathsf{TREG}^{\vec{O}}_P(A, \Sigma)$ refers to the class of forests $F_{|P}$, for $F \in \mathsf{TREG}^{\vec{O}}(A, \Sigma)$.

2.2. Tree languages as models of formulas

We adapt here the interpreted formalism of the MSO-logic of two successors (S2S) introduced in [3] to establish a correspondence between $\vec{O}$-regular forests and models of MSO-formulas over a labeled tree structure. For easier exposition, we shall restrict to binary trees. In addition trees will be labeled by $\{0, 1\}^n$, $n \geq 0$ (we will write Tree_n instead of $\{0, 1\}^*\text{-}\mathsf{Tree}_n$). All definitions and results can be naturally extended to the case where A is arbitrary. In this subsection, $\vec{O}$ is always a fixed vector $(O_1, \ldots, O_m)$, with $m \geq 1$ and P is a prefix closed subset of $\{0, 1\}^*$.

We recall first the interpreted formalism of the MSO-logic of two successors by sticking to notations used in [23, Chapter 12].

Definition 19 (*Tree Structure*)**.** Let $t = \chi^{\vec{O}}_P \in P\text{-}\mathsf{Tree}_m$, we associate t to the structure

$$\underline{t} = \langle P, \varepsilon, \mathrm{succ}_0, \mathrm{succ}_1, O_1, \ldots, O_m \rangle,$$

where $\epsilon = \{\epsilon\}$ and $\forall i \in \{0, 1\}, \mathrm{succ}_i = \{(u, ui),\ u \in P,\ ui \in P\}$.

Definition 20. An S2S-formula is an MSO-formula defined over the signature $(\mathrm{succ}_0, \mathrm{succ}_1)$, where succ_i is a 2-ary relation symbol.
If $\phi(X_1, \ldots, X_m)$ is an S2S-formula and $t = \chi^{\vec{O}}_{\{0,1\}^*} \in \mathsf{Tree}_m$, write $\underline{t} \vdash \phi(\bar{X})$ if $\langle \{0,1\}^*, \varepsilon, \mathrm{succ}_0, \mathrm{succ}_1 \rangle \models \phi(\vec{O})$.
Let $T(\phi) = \{t \in \mathsf{Tree}_m \mid \underline{t} \vdash \phi(X_1, \ldots, X_m)\}$. A tree language $F \in \mathsf{Tree}_m$ is called definable in S2S if $F = T(\phi)$ for some S2S-formula ϕ.

Theorem 21 ([3]). *The union of classes* $\mathsf{TREG}(\{0,1\}^n)$, *for* $n \geq 0$, *corresponds exactly to the class of tree languages definable in* S2S.

We now interpret S2S-formulas by fixing some free variables and interpreting formulas over restricted trees.

Definition 22. Let $\phi(X_1, \ldots, X_n)$ be an S2S-formula with $n \geq m$, we define the forest $T^{\vec{O}}_P(\phi)$ (or $T^{\vec{O}}$ if $P = \{0,1\}^*$) by:

$$T^{\vec{O}}_P(\phi) = \{t \in P\text{-}\mathsf{Tree}_{n-m} \mid \underline{t \otimes \chi^{\vec{O}}_P} \vdash \phi(X_1, \ldots, X_n)\},$$

with the convention that if $t \in P\text{-}\mathsf{Tree}_0$ then $t \otimes \chi^{\vec{O}}_P = \chi^{\vec{O}}_P$.

If $F = T^{\vec{O}}_P(\phi)$ for some S2S-formula ϕ, F is called definable in $\mathrm{S2S}^{\vec{O}}_P$ (or in $\mathrm{S2S}^{\vec{O}}$ if $P = \{0,1\}^*$).

Remark that if $n \neq m$, then $T^{\vec{O}}(\phi) = \pi_{1,n-m}(T(\phi) \cap F^{\vec{O}}_{\chi}(\{0,1\}^{n-m}))$. Then, using Observation 15, Theorem 21 can easily be extended to the $\mathrm{S2S}^{\vec{O}}$ formalism.

Theorem 23. *The union of classes* $\mathsf{TREG}^{\vec{O}}(\{0,1\}^n)$, *for* $n \geq 0$, *corresponds exactly to the class of tree languages definable in* $\mathrm{S2S}^{\vec{O}}$.

Remark 24. Given ϕ an S2S-formula, the size (the number of states) of $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}$ such that $\mathrm{F}(\mathcal{A}) = T^{\vec{O}}(\phi)$ is at most the size of $\widetilde{\mathcal{A}} \in \mathsf{TFA}$ such that $\mathrm{F}(\widetilde{\mathcal{A}}) = T(\phi)$. Then if ϕ has q quantifier alternations and its length is n, the size of $\mathcal{A}$ is $n \uparrow q$, i.e., a tower $2^{2^{\cdot^{\cdot^{\cdot^{\mathcal{O}(n)}}}}}$ of height $q+1$ (see [24, Section 12.3]).

We obtain as corollary of Theorem 23:

Corollary 25. *The emptiness problem is decidable for forests in* $\mathsf{TREG}^{\vec{O}}(\{0,1\}^n)$, *for all* $n \geq 1$ *iff* $\underline{\chi^{\vec{O}}_{\{0,1\}^*}}$ *has a decidable MSO-theory.*

Proof. $\underline{\chi^{\vec{O}}_{\{0,1\}^*}}$ has a decidable MSO-theory

iff one can decide whether $\underline{\chi^{\vec{O}}_{\{0,1\}^*}} \models \phi$ for any S2S-formula ϕ,
iff one can decide whether $\underline{\chi^{\vec{O}}_{\{0,1\}^*}} \vdash \phi'(X_1, \ldots, X_m)$ for any S2S-formula $\phi'(X_1, \ldots, X_m)$,
iff one can decide whether $\underline{\chi^{\vec{O}}_{\{0,1\}^*}} \vdash \exists Y_1, \ldots Y_n,\ \psi(Y_1, \ldots Y_n, X_1, \ldots, X_m)$, for any $n \geq 0$ and any S2S-formula $\psi(Y_1, \ldots, Y_n, X_1, \ldots, X_m)$,
iff one can decide whether there exists $t \in \mathsf{Tree}_n$ such that $\underline{t \otimes \chi^{\vec{O}}_{\{0,1\}^*}} \vdash \psi(Y_1, \ldots Y_n, X_1, \ldots, X_m)$, for any $n \geq 0$ and any S2S-formula $\psi(Y_1, \ldots, Y_n, X_1, \ldots, X_m)$,
iff one can decide whether $T^{\vec{O}}(\psi) = \emptyset$, for any $n \geq 0$ and any S2S-formula $\psi(Y_1, \ldots, Y_n, X_1, \ldots, X_m)$,
iff the emptiness problem is decidable for forests in $\mathsf{TREG}^{\vec{O}}(\{0,1\}^n)$, $n \geq 0$ (from Theorem 23). □

We generalize now Theorem 23 to tree languages of domain P.

Theorem 26. *If* P *is* MSO*-definable in* $\underline{\chi^{\vec{O}}_{\{0,1\}^*}}$, *with* $O_i \subseteq P$, *then union of classes* $\mathsf{TREG}^{\vec{O}}_P(\{0,1\}^n)$, *for* $n \geq 0$, *corresponds exactly to the class of* $\mathrm{S2S}^{\vec{O}}_P$*-definable tree languages.*

We start by proving the following lemma

Lemma 27. *If* P *is MSO-definable in* $\underline{\chi^{\vec{O}}_{\{0,1\}^*}}$ *then every tree language definable in* $\mathrm{S2S}^{\vec{O}}_P$ *belongs to* $\mathsf{TREG}^{\vec{O}}_P(\{0,1\}^n)$ *for some* $n \geq 1$.

Proof. Let $\phi(X_1, \ldots, X_n)$ be an S2S-formula, by relativizing ϕ to P, we construct an S2S-formula $\phi_P(X_1, \ldots, X_n)$ such that $\forall S_1, \ldots, S_n \subseteq \{0,1\}^*$,

$$\underline{\chi^{\vec{O}}_{\{0,1\}^*}} \models \phi_P(S_1, \ldots, S_n) \quad \text{iff} \quad \underline{\chi^{\vec{O}}_P} \models \phi(S_1, \ldots, S_n) \text{ and } S_1, \ldots, S_n \subseteq P.$$

Let $F = T^{\vec{O}}_P(\phi)$ and $F' = T^{\vec{O}}(\phi_P)$, then $F = F'_{|P}$. From Theorem 23 applied to ϕ_P, F' is $\vec{O}$-regular and thus $F \in \mathsf{TREG}^{\vec{O}}_P(\{0,1\}^n)$. □

The restriction of the domain makes the proof of the other direction more difficult, since it cannot be directly deduces from the general case. Hence, we restrict ourselves to P-cut automata (see Definition 18), for which we do not have to consider vertices outside P.

Lemma 28. *For every P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(\{0,1\}^n)$, the forest $\mathrm{F}(\mathcal{A})_{|P}$ is S2S$^{\vec{O}}_P$-definable.*

Proof. In the same way as for automata without oracles (see [24, Lemma 12.20]), one can effortlessly construct an S2S-formula which S2S$^{\vec{O}}_P$-define $\mathrm{F}(\mathcal{A})$. In addition, if $\mathcal{A}$ is P-cut, one can easily show that the same formula S2S$^{\vec{O}}_P$-define $F(\mathcal{A})_{|P}$. □

To achieve the proof of Theorem 26, it remains to show the following lemma.

Lemma 29. *Suppose that $P \in \mathsf{REG}^{\vec{O}}(\{0,1\})$ and $\forall i \in [1,m]$, $O_i \subseteq P$. For every $F \in \mathsf{TREG}^{\vec{O}}(\{0,1\}^k)$, there effectively exists a P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(\{0,1\}^k)$ such that $F_{|P} = \mathrm{F}(\mathcal{A})_{|P}$.*

Proof. We considers the sets $P \cdot i^{-1} = \{u \in \{0,1\}^* \mid ui \in P\}$, for $i = 0, 1$ and the vector $\vec{P} = (P \cdot 0^{-1}, P \cdot 1^{-1})$. Since $P \in \mathsf{REG}^{\vec{O}}(\{0,1\})$ one can easily find an automaton in $\mathsf{TFA}^{\vec{O}}(\{0,1\}^2)$ recognizing $\{\chi^{\vec{P}}_{\{0,1\}^*}\}$ which then belongs to $\mathsf{TREG}^{\vec{O}}(\{0,1\}^2)$. Consider the forest $F' = F \otimes \{\chi^{\vec{P}}_{\{0,1\}^*}\}$, from Theorem 17, there exists $\mathcal{A}_1 \in \mathsf{TFA}^{\vec{O}}(\{0,1\}^{k+2})$ such that $\mathrm{F}(\mathcal{A}_1) = F'$. Then $\mathcal{A}_1$ allows to describe F, and also the *borders* of P. From $\mathcal{A}_1$, we construct now a new automaton $\mathcal{A}$ able to check from the borders of P, that there exists a labeling of the subtrees external to P such that the complete tree belongs to F.

For every $q \in Q_1$ (the set of states of $\mathcal{A}_1$), we construct $\mathcal{A}_q \in \mathsf{TFA}(\{0,1\}^{k+2})$ whose transitions are all (p, α, q) such that $(p, \alpha, (0, \ldots, 0), q)$ is a transition of $\mathcal{A}_1$ and whose initial state is q. Since $\forall i \in [1,m]$, $O_i \subseteq P$, these transitions are the only ones that can be applied outside of P. The emptiness problem being decidable for regular forests ([3,23][Chapter 9]), we can effectively construct the set $Acc = \{q \in Q_1 \mid \mathrm{L}(\mathcal{A}_q) \neq \emptyset\}$.
Acc describes states from which, outside of P, one can find an accepting subtree. Then, $t \in F(\mathcal{A}_1)$ iff there exists a run r over t, such that

- every infinite path $r(u_1)r(u_2)\cdots r(u_n)\cdots$, with $u_i \in P$ is successful, and
- for all $u \in P$, for all $i = 0, 1$, if $t(u) = (\alpha, b_0, b_1)$ and $b_i = 0$, then $r(ui) \in Acc$.

Now, we construct an automaton $\mathcal{A}_2$ which is P-cut and such that $\mathrm{F}(\mathcal{A}_1)_{|P} = \mathrm{F}(\mathcal{A}_2)_{|P}$. We obtain this automaton by adding $q_\perp$ to the set of states (with $c(q_\perp) = 0$) and modifying the set of transitions of $\mathcal{A}_1$ in the following way: a transition $(p, (\alpha, b_0, b_1), \vec{o}, q_0, q_1)$ belongs to Q_2 iff

- $b_0 = b_1 = 1$ and $(p, (\alpha, b_0, b_1), \vec{o}, q_0, q_1)$ belongs to $Q_{\mathcal{A}_1}$, or
- There exists a set $I \subseteq \{0,1\}$ such that $\forall i \in I, b_i = 0$ and $q_i = q_\perp$ and there exists a transition $(p, (\alpha, b_0, b_1), \vec{o}, p_0, p_1) \in Q_1$ such that for all i, $p_i = q_i$ if $i \notin I$ and $p_i \in Acc$ if $i \in I$.

From this new automaton, it is then easy to construct a P-cut automaton $\mathcal{A}$ recognizing the language $\pi_{1,k}(\mathrm{F}(\mathcal{A}_2))$. We have then $\mathrm{F}(\mathcal{A})_{|P} = F_{|P}$. □

Proof of Theorem 26. From Lemmas 28 and 29, if $P \in \mathsf{REG}^{\vec{O}}(\{0,1\})$ and $\forall i \in [1,m]$, $O_i \subseteq P$, then every tree language in $\mathsf{TREG}^{\vec{O}}_P(\{0,1\}^n)$ is S2S$^{\vec{O}}_P$-definable. Combined with Lemma 27, this proves the Theorem 26 □

Complexity analysis. Suppose P is recognized by a word-automaton of size τ_P and F by a tree automaton of size τ. Then the size of $\mathcal{A}_1$ is $\tau \cdot \tau_P$, so is the P-cut automaton $\mathcal{A}$.
The construction of $\mathcal{A}$ requires to construct the set Acc. From [24][Cor 8.22], for a parity tree automaton of size s, this can be made in time $\mathcal{O}(|\Sigma| \cdot s^s)$. Then $\mathcal{A}$ can be constructed in time $\mathcal{O}(|\Sigma| \cdot (\tau \cdot \tau_P)^{\tau \cdot \tau_P})$.

Corollary 30. *The emptiness problem is decidable for forests in $\mathsf{TREG}^{\vec{O}}_P(\{0,1\}^k)$, for all $k \geq 0$ iff $\underline{\chi^{\vec{O}}_P}$ has a decidable MSO-theory.*

2.3. Regular trees and selection property

Regular trees are a natural extension of finite trees: they correspond to unfolding of finite graphs, i.e., of graphs of finite automata. They are useful in several areas of computer science (see [6] for a survey on basic theory and applications in semantics). We generalize here the notion of regular trees by defining **$\vec{O}$-regular trees** which correspond to the unfolding of a deterministic word automaton with p-oracle $\vec{O}$, with respect to conditions on transitions imposed by the tests. We then study links between existence of such a tree in a forest recognized with oracle-automata, and the satisfiability of the selection property for a labeled tree structures. Eventually, we close this subsection by defining **input-free** tree automata (with p-oracles), which are, as the name suggests, tree automata operating without any input trees. We show that the study of the emptiness problem and the existence of an accepting $\vec{O}$-regular tree for a tree automaton with oracles can be restricted to the study of these problems for input-free tree automata with oracles.

$\vec{O}$-Regular trees. A tree $t \in A^*\text{-Tree}(\Sigma)$ is said to be **$\vec{O}$-regular** iff there exists a deterministic word automaton $\mathcal{A} \in \mathsf{FA}^{\vec{O}}(A)$ and a function $out : Q \to \Sigma$ *generating* t, i.e., such that $\forall u \in A^*, q \in Q$,

$$(q_0, \uparrow u) \to_{\mathcal{A}} (q, u_\uparrow) \quad \text{iff} \quad out(q) = t(u).$$

Remark 31. The following remarks will be useful:

1. If $t \in A^*\text{-Tree}(\Sigma)$ is $\vec{O}$-regular, then for every $\alpha \in \Sigma$, the set of nodes of t labeled by α (i.e., the set $L_\alpha = \{u |\ t(u) = \alpha\}$) is $\vec{O}$-regular.
2. For every $\vec{O}$, the characteristic tree $\chi^{\vec{O}}_{A^*}$ is $\vec{O}$-regular.

We extend this definition to P-trees: any $t \in P\text{-Tree}(\Sigma)$ is $\vec{O}$-regular when there exists $t' \in \mathsf{Tree}(\Sigma)$, $\vec{O}$-regular such that $t = t'_{|P}$.

The selection property. We study links between regular trees and the selection property (SP) (see [10,11]). A structure has the selection property, if for each satisfiable MSO-formula, there exists a model of the formula which is definable.

Definition 32. A structure $\mathcal{S}$ satisfies the *selection* property (SP) if for every formula $\phi(X) \in \mathrm{MSO}(Sig)$ satisfiable in $\mathcal{S}$, there exists $S \subseteq D_{\mathcal{S}}$ such that

1. $\mathcal{S} \models \phi(S)$ and
2. S is MSO-definable in $\mathcal{S}$.

Proposition 33. *If P is* MSO*-definable in $\underline{\chi^{\vec{O}}_{A^*}}$, the following properties are equivalent:*

1. *$\underline{\chi^{\vec{O}}_P}$ fulfills* SP,
2. *for all $n \geq 1$, for every non-empty forest $F \in \mathsf{TREG}^{\vec{O}}_P(\{0,1\}^n)$, there exists $\vec{D}$* MSO*-definable in $\underline{\chi^{\vec{O}}_P}$ such that F contains a $\vec{D}$-regular tree.*

Proof. Suppose P is MSO-definable in $\underline{\chi^{\vec{O}}_{A^*}}$, a rewriting of the selection property using Theorem 26 implies the equivalence of the two following properties:

(1) $\underline{\chi^{\vec{O}}_P}$ fulfills SP,

(2′) for every nonempty $F \subseteq \mathsf{TREG}^{\vec{O}}_P(\{0,1\}^n)$, there exists $\vec{S} = (S_1, \ldots, S_n)$ MSO-definable in $\underline{\chi^{\vec{O}}_P}$ such that the tree $\chi^{\vec{S}}_P$ belongs to F.

(2′) $\Rightarrow$ (2) Suppose (2′), according to Remark 31(2), the tree $\chi^{\vec{S}}_P$ is $\vec{S}$-regular.

(2) $\Rightarrow$ (2′) Suppose that $F \subseteq \mathsf{TREG}^{\vec{O}}_P$ is $\vec{O}$-regular and contains a $\vec{D}$-regular tree t, for $\vec{D}$ MSO-definable in $\underline{\chi^{\vec{O}}_P}$. From definition of $\vec{D}$-regular tree, the language $\{t\}$ is $\vec{D}$-regular. Suppose that $t = \chi^{\vec{S}}_P$, with $\vec{S} = (S_1, \ldots, S_n)$, Theorem 26 ensures that $\vec{S}$ is MSO-definable in $\underline{\chi^{\vec{D}}_P}$. Since $\vec{D}$ is MSO-definable in $\underline{\chi^{\vec{O}}_P}$, $\vec{S}$ is too. □

Any non-empty regular forest contains a regular tree ([3], [23][Thm 9.3]), the following result is then a straightforward corollary of Proposition 33.

Theorem 34. *For every finite alphabet A, the structure $\langle A^*, \varepsilon, (\mathrm{succ}_a)_{a \in A}\rangle$ fulfills the selection property.*

Input-free tree automata. To deal with emptiness problems or existence of regular trees, one can without lose of generality work with **input-free tree automata** i.e., tree automata whose input alphabet is $\{\top\}$. The input letter can be omitted in the transitions of a such an automaton, thus $\Delta \subseteq Q \times \{0,1\}^m \times Q^{|A|}$. In the sequel, we write $\mathsf{TFA}^{\vec{O}}(A)$ rather than $\mathsf{TFA}^{\vec{O}}(A, \{\top\})$.

Any tree automaton with m oracles $\mathcal{A} = (Q, \Sigma, A, \vec{O}, \Delta, q_0, c)$ can be transformed in $\mathcal{B} = (Q \times \Sigma, A, \vec{O}, \Delta', Q_0, c')$ input-free where for every $\alpha_1, \ldots, \alpha_n \in \Sigma$, $((q,\alpha), \vec{o}, (p_1,\alpha_1), \ldots, (p_{|A|}, \alpha_{|A|})) \in \Delta'$ iff $(q, \alpha, \vec{o}, p_1, \ldots, p_{|A|}) \in \Delta$. Q_0 contains every (q_0, α), $\alpha \in \Sigma$ and $c'(q,\alpha) = c(q)$, $\forall \alpha \in \Sigma$. (It remains to reduce Q_0 to only one state, this construction being classical, we do not describe it.) Obviously, successful runs of $\mathcal{B}$ are exactly pairs $r' = r \otimes t$, where r is a successful run of $\mathcal{A}$ over t. We obtain then the following result which will permit to restrict the next proofs to input-free automata:

Proposition 35. *For every $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A, \Sigma)$, one can construct an input-free automaton $\mathcal{B} \in \mathsf{TFA}^{\vec{O}}(A)$ satisfying:*

1. *the forest* $\mathrm{F}(\mathcal{A})$ *is non-empty iff there exists a successful run on $\mathcal{B}$,*
2. *for all vector $\vec{R}$ of subsets of A^*, the forest* $\mathrm{F}(\mathcal{A})$ *contains a $\vec{R}$-regular tree iff there exists a successful $\vec{R}$-regular run on $\mathcal{B}$,*
3. *for every set $P \subseteq A^*$ prefix closed, if $\mathcal{A}$ is P-cut, then $\mathcal{B}$ is P-cut.*

Proposition 36. *For every $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$ input-free and deterministic, if there exists a run of $\mathcal{A}$ then this run is unique and $\vec{O}$-regular.*

Proof. Let us suppose that $\mathcal{A} = (Q, A, \vec{O}, \Delta, q_0, c)$, with $A = \{a_1, \ldots, a_n\}$, and consider the word-automaton $\mathcal{A}_r = (Q, A, \vec{O}, \Delta_r, q_0)$ where Δ_r consists of all transitions $(q, a_i, \vec{o}, p_i)$ such that $(q, \vec{o}, p_1, \ldots, p_n) \in \Delta$. Clearly, $\mathcal{A}_r$ is deterministic and if there exists a successful run of $\mathcal{A}$, it corresponds to the tree generated by $\mathcal{A}_r$ associated to the function $out : q \mapsto q$, $\forall q \in Q$. □

3. Logic for restricted oracles

In this section, we study relation between a tree structure and its image structure:

Definition 37 (*Image Structure*)**.** Let $P \subseteq A^*$, $t = \chi_P^{\vec{O}}$ and $\mu : P \to S$ be a map. We denote by $\mu(\underline{t})$ the relational structure

$$\mu(\underline{t}) = \langle \mu(P), \mu(\varepsilon), (E_a)_{a\in A}, \mu(O_1), \ldots, \mu(O_n)\rangle,$$

where $E_a = \{(\mu(u), \mu(ua)) \mid u, ua \in P\}$.

Remark that if $\mu : (P, \cdot) \to (M, \star)$ is a morphism, then

$$E_a = \{(\sigma, \sigma') \mid \sigma, \sigma' \in \mu(P),\ \sigma' = \sigma \star \mu(a)\}.$$

We now restrict ourselves to tree automata with oracles of the form $\vec{O} = (\mu_{|P}{}^{-1}(R_1), \ldots, \mu_{|P}{}^{-1}(R_m))$, where μ is a morphism from A^* to any monoid M, P is a prefix closed subset of A^* and $R_i \subseteq M$.

We use a game-theoretical approach to express problems over $\vec{O}$-regular forests by means of MSO-formulas over the graph structure $\mu(\underline{\chi_P^{\vec{O}}})$. This allows to obtain some transfer results:

1. if P is MSO-definable in $\underline{\chi_{A^*}^{\vec{O}}}$, then the MSO-decidability can be transferred from $\mu(\underline{\chi_P^{\vec{O}}})$ to $\underline{\chi_P^{\vec{O}}}$.
2. we define a condition on $\mu_{|P}$ making possible the transfer of the selection property from $\mu(\underline{\chi_P^{\vec{O}}})$ to $\underline{\chi_P^{\vec{O}}}$.

The condition on $\mu_{|P}$ also ensures that if $\mu(\underline{\chi_P^{\vec{O}}})$ satisfies SP, then the class of sets which are MSO-definable in $\underline{\chi_P^{\vec{O}}}$ corresponds exactly to the class of $\mu_{|P}{}^{-1}(\vec{D})$-regular languages intersected with P, for any $\vec{D}$ MSO-definable in $\mu(\underline{\chi_P^{\vec{O}}})$.

3.1. Games for prefix-oracle automata

Parity game. A two-player game (player 0 and player 1) is a colored directed graph whose set of vertices V is partitioned in player 0's vertices (V_0) and player 1 ones (V_1), associated to a winning condition. Parity games are special games which have been much studied (see for example [25,24,26]).

Definition 38. A **parity game** is a tuple $G = (V_0, V_1, E, v_0, c)$ where $V = V_0 \uplus V_1$ is the set of positions, $E \subseteq V \times V$ is the sets of possible moves, $v_0 \in V$ is the start position and $c : V \to [0, max]$ is a map associating to each vertex a priority by means of an integer which belongs to a bounded interval. A **play** in G is a (finite or infinite) path in the graph (V, E) starting at v_0. If the play is finite and ends in any vertex $v \in V_\epsilon$, $\epsilon \in \{0, 1\}$ (i.e., player ϵ cannot play anymore), then player ϵ is declared loser (and therefore the other player wins the play). Otherwise, the winner is determined by n_0, value of the minimal priority appearing infinitely often in the play. In other words, if the play is $v_0 v_1 \cdots v_n \cdots$, then n_0 is the smallest integer having an infinity of occurrences in the word $c(v_0)c(v_1)\cdots c(v_n)\cdots$. If n_0 is even, player 0 is declared the winner of the play, otherwise player 1 wins the play.

A **strategy** for player ϵ is a map $s : (V^*V_\epsilon) \to V$ connecting any prefix of play $\rho = v_0 v_1 \ldots v_n$ to a vertex v_{n+1} such that $(v_n, v_{n+1}) \in E$. A strategy is **memoryless** if for any $\rho = v_0 v_1 \ldots v_n$ the value of $s(\rho)$ depends only on the current vertex v_n. In this case, the strategy is represented as an application from V_ϵ to V. A play $\rho = v_0 v_1 \ldots v_n \ldots$ is said **conform** with s if for any $i \geq 0$ if $v_i \in V_\epsilon$, then $v_{i+1} = s(v_0 \ldots v_i)$. A strategy s for player ϵ is a **winning strategy** if every play conform with s is won by player ϵ. We say that player ϵ **wins the game** if there exists a **winning** strategy for ϵ.

The following result will be useful in the sequel.

Theorem 39 (*[25,24]*)**.** *Given any parity game G:*

1. *one and only one player wins the game,*
2. *for $\epsilon \in \{0, 1\}$, if player ϵ wins the game, then player ϵ has a winning memoryless strategy.*

Games with p-oracle and regular trees. We use now parity games to express some problems related to $\vec{O}$-regular tree languages in the context fixed as follows:

- A is a finite alphabet, supposed to be reduced to two elements: $A = \{a_0, a_1\}$ (all results established in this subsection remain true if the size of A is arbitrary),
- μ is a surjective morphism from A^* onto a monoid $(M, \star)$,

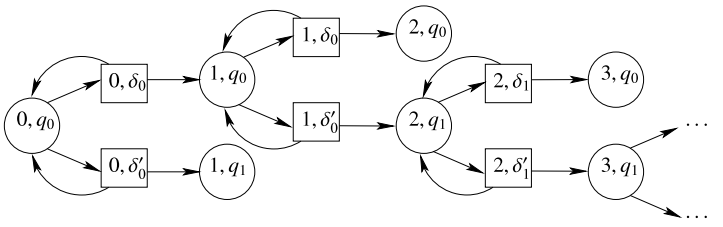

Fig. 2. The game $G_{\mathcal{A}}$.

- P is a prefix closed subset of A^*,
- $\vec{O} = (\mu_{|P}{}^{-1}(R_1), \ldots, \mu_{|P}{}^{-1}(R_m))$, with $m \geq 1$ and $R_i \subseteq M$.

We prove that the emptiness problem for $\vec{O}$-regular tree languages reduces to the determination of the winner of a parity game. From this result, we show (Proposition 51(1)) that the emptiness problem for $\vec{O}$-regular tree languages reduces to the satisfiability of an MSO-formula in $\mu(\underline{\chi_P^{\vec{O}}})$ (see Definition 37). We prove, in addition, that every non-empty $\vec{O}$-regular tree language contains a $\mu_{|P}(\vec{D})$-regular tree, where $\vec{D}$ is MSO-definable in $\mu(\underline{\chi_P^{\vec{O}}})$ (Proposition 51(2)).

We first restrict ourselves to the study of input-free P-cut automata. The advantage of using P-cut automata is that to determine the existence of a run of a successful one, we only need to consider nodes in P. In addition, in a run of a P-cut automaton, nodes which do not belong to P are indicated by the label $q_\perp$. Then, from such an automaton, one can easily find a game which do not leave P.

Definition 40 (*Game with p-Oracles*)**.** Given an input-free P-cut automaton $\mathcal{A} = (Q, A, \vec{O}, \Delta, q_0, c)$, we construct the parity game $G_{\mathcal{A}} = (V_0, V_1, E, v_0, c')$ where
$V_0 = \mu(P) \times Q$ and $V_1 = \mu(P) \times \Delta$, $v_0 = (\mu(\varepsilon), q_0)$
$E = E_0 \cup E_1$ where $E_0 \subseteq V_0 \times V_1$, $E_1 \subseteq V_1 \times V_0$ and
$E_0 = \{((\sigma, p), (\sigma, \delta)) \mid \delta = (p, \chi_M^{\mu(\vec{O})}(\sigma), p_0, p_1) \in \Delta\}$,
$E_1 = \{((\sigma, \delta), (\sigma \star \mu(a_i), p_i)) \mid p_i \neq q_\perp,\ \delta = (p, \vec{o}, p_0, p_1),\ i \in \{0, 1\}\}$ and
c' is defined by $c'(\sigma, p) = c(p)$ and $c'(\sigma, (p, \vec{o}, p_0, p_1)) = c(p)$.

Each player moves alternately in the game. In position $(\mu(u), p)$, player 0 chooses a transition $\delta = (p, \vec{o}, p_0, p_1)$ from those fulfilling $\vec{o} = \chi_{A^*}^{\vec{O}}(u)$. He moves then to $(\mu(u), \delta)$. Now its the player 1's turn to play, he chooses a direction a_i to follow $(i \in \{0, 1\})$ and moves to $(\mu(ua_i), p_i)$. Hence, μ being a morphism, for every prefix of play ending in $(\sigma, x) \in V, \sigma = \mu(u)$ where u consists of the sequence of directions chosen by player 1.

Example 41. Given two sets A, B, with $B \subseteq A$, we define the map $\ell_B : A^* \to \mathbb{N}$ associating to every word in A^* its number of occurrences of letters in B. The map l_B is a surjective morphism when $\mathbb{N}$ is endowed with the "+" operator.

Suppose now that $A = \{a_0, a_1\}$ and $B = \{a_1\}$. Let $O = \ell_B{}^{-1}(\mathbb{P})$ where $\mathbb{P}$ is the set of prime numbers, be the set of words containing a prime number of occurrences of a_1 and $\mathcal{A} = (\{q_0\}, A, O, \Delta, c : q_0 \mapsto 0)$, where $\Delta = \{\delta_0 = (q_0, 0, q_0, q_0), \delta_0' = (q_0, 0, q_0, q_1), \delta_1 = (q_1, 1, q_1, q_0), \delta_1' = (q_1, 1, q_1, q_1)\}$. Clearly, $\mathcal{A}$ admits a unique run r, and r is such for all $u \in A^*$, $\ell_B(u)$ is prime iff $r(u) = q_1$.

The associated game is $G_{\mathcal{A}} = (V_0, V_1, E, (0, q_0), c' : v \in V_1 \cup V_0 \mapsto 0)$ is presented Fig. 2. We have the 0-vertices $V_0 = \mathbb{N} \times \{q_0, q_1\}$ (circles) and the 1-vertices $V_1 = \mathbb{N} \times \{\delta_0, \delta_0', \delta_1\}$ (squares).

Remark that there are dead ends in our example. Hence, if Player 0 moves from $(0, q_0)$ to $(0, \delta_0')$, Player 1 can win the play by moving to $(1, q_1)$.

Lemma 42. *For any input-free P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$, $\mathcal{A}$ has a successful run iff player* 0 *has a winning strategy in $G_{\mathcal{A}}$.*

Proof. Let r be a run on $\mathcal{A}$, and s_r the strategy defined by: $\forall v_0 \ldots v_n$ prefix of a play with $v_n \in V_0$ and such that the sequence of directions chosen by player 1 is $u \in A^*$,

$$s_r(v_0 \ldots v_n) = (\mu(u), (r(u), \chi_{A^*}^{\vec{O}}(u), r(ua_0), r(ua_1))).$$

Clearly, s_r is winning iff r is successful.

Conversely, given any winning strategy s for player 0 we construct the tree r_s by applying to each vertex u, the transition given by $s(\rho_u)$ where ρ_u is the prefix of play conform with s whose sequence of selected directions is u:

- $r_s(\varepsilon) = q_0$ and $\rho_\varepsilon = (\mu(\varepsilon), q_0)$,
- $\forall u \in P$, if $s(\rho_u) = (\mu(u), (p, \vec{o}, p_0, p_1))$, then $\forall i \in \{0, 1\}$, $r_s(ua_i) = p_i$ and $\rho_{ua_i} = \rho_u \cdot s(\rho_u) \cdot (\mu(u) \star \mu(a_i), p_i)$,
- $\forall u \notin P$, $r_s(u) = q_\perp$.

The tree thus constructed is obviously a run on $\mathcal{A}$ and is successful iff s is a winning strategy. □

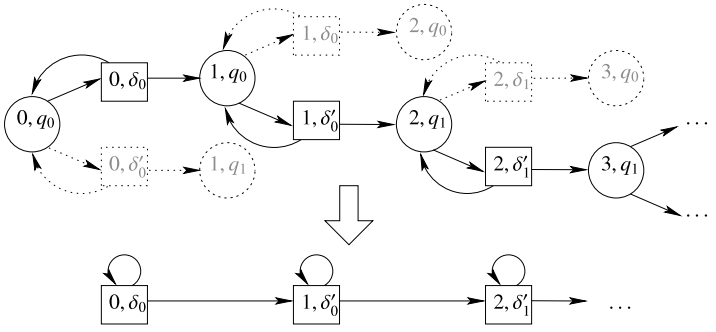

Fig. 3. The game $G^{\mathsf{s}}_{\mathcal{A}}$.

Thus, by applying Theorem 39:

Lemma 43. *For any input-free P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$, $\mathcal{A}$ has a successful run iff player* 0 *has a memoryless winning strategy in $G_{\mathcal{A}}$.*

Given a memoryless strategy s for player 0, to decide whether s is winning, we simply need to consider the game reduced to plays conform with s. In this game, there is always only one outgoing edge of a node of player 0. One can then find an equivalent game in which 0 does not play. Let $G^{\mathsf{s}}_{\mathcal{A}} = (\emptyset, V_0 \cup V_1, E^{\mathsf{s}}, c')$, where $E^{\mathsf{s}} = E_1 \cup \{(v, v') \mid v \in V_0,\ \mathsf{s}(v) = v'\}$. Without loss of generality, we can suppose that $\mathcal{A}$ is *complete*, i.e., for every $(q, \vec{o})$, there exists a transition $(q, \vec{o}, q_1, q_2)$. In this case, any finite play in $G_{\mathcal{A}}$ ends in a player 1's position, and is then winning for player 0.

Example 44. Consider the game $G_{\mathcal{A}}$ defined in Example 41 and the strategy s defined for $i \in \mathbb{N}, \epsilon \in \{0, 1\}$ by $\mathsf{s}(i, q_\epsilon) = (i, \delta'_\epsilon)$ if $i + 1$ is prime, else $\mathsf{s}(i, q_\epsilon) = (i, \delta_\epsilon)$.

The transformation of $G_{\mathcal{A}}$ in $G^{\mathsf{s}}_{\mathcal{A}}$ is depicted Fig. 3

In this game, there are no more dead ends, thus every play is winning for Player 0.

Lemma 45. *For every complete input-free P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$, any memoryless strategy for player* 0, s, *is winning in $G_{\mathcal{A}}$ iff the reduced game $G^{\mathsf{s}}_{\mathcal{A}}$ is winning for player* 0.

Proof. An infinite sequence of vertices $v_0 \cdots v_n \cdots$ is a play in $G^{\mathsf{s}}_{\mathcal{A}}$ iff it is a play in $G_{\mathcal{A}}$ conforms with s. Since vertices of $G_{\mathcal{A}}$ and $G^{\mathsf{s}}_{\mathcal{A}}$ have same priority, an infinite sequence of vertices is a winning play in $G_{\mathcal{A}}$ conform with s iff it is a winning play in $G^{\mathsf{s}}_{\mathcal{A}}$. □

Any parity game $G = (V_0, V_1, E, v_0, c)$ with $c : V \to [0, max]$ is naturally associated to a relational structure $\mathbf{G}$ of domain V defined over the signature $\mathcal{G} = (V_0, V_1, E, v_0, c_0, \ldots, c_{max})$, where for all $i \in [0, max]$, the arity of c_i is 1.

Lemma 46. *For every complete input-free P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$, one can find an MSO($\mathcal{G}_{\mathcal{A}}$)-sentence* WIN *such that for every* s *memoryless strategy for player* 0 *in $G_{\mathcal{A}}$,*

$$\mathbf{G}^{\mathsf{s}}_{\mathcal{A}} \models \textsc{win} \quad \textit{iff} \quad \mathsf{s} \textit{ is winning.}$$

Proof. We construct a formula P_0 such that $\mathbf{G}^{\mathsf{s}}_{\mathcal{A}} \models P_0$ iff there exists a play in $G^{\mathsf{s}}_{\mathcal{A}}$ lost by player 0, i.e., iff there exists an infinite path $v_0 \cdots v_n \cdots$ such that the smallest integer appearing infinitely often in $c'(v_0) \cdots c'(v_n) \cdots$ is odd.

$P_0 := \exists X, X_0, \ldots, X_{max}$,

1. X is a path containing v_0
2. $\forall n, X_n = \{x \text{ appearing infinitely often in } X \mid c_n(x)\}$
3. the smallest n such that $X_n \neq \emptyset$ is odd

Using [24, Section 12.2], "being a path" is MSO-expressible in $G^{\mathsf{s}}_{\mathcal{A}}$. From Theorem 39, player 0 loses the game iff player 1 wins the game, hence $\textsc{win} := \neg P_0$. □

We now translate MSO-formulas over $\mathbf{G}^{\mathsf{s}}_{\mathcal{A}}$ into MSO-formulas over $\mu(\underline{\chi^{\vec{O}}_P})$, using the following facts:

1. since the domain V of $\mathbf{G}^{\mathsf{s}}_{\mathcal{A}}$ is $\mu(P) \times Q \cup \mu(P) \times \Delta$, and both Q and Δ are finite, a subset of V can be encoded by a vector of subsets of $\mu(P)$ of size $|\Delta| + |Q|$,
2. a memoryless strategy for 0 maps vertices $(\sigma, \pi_1(\delta))$ on vertices (σ, δ) were $\delta \in \Delta$. It can then be expressed by a map from $\mu(P)$ into Δ. Since Δ is finite, it can be encoded by a vector of subsets of $\mu(P)$ (of size $|\Delta|$).

Formally, we fix the following notations:

- $\Delta = \{\delta_1, \ldots, \delta_d\}$, where $\delta_i \neq \delta_j$ for all $i \neq j$,
- $Q = \{s_1, \ldots, s_\tau\}$,

- for every $D \subseteq V$, $g(D) = (g_1(D), \ldots, g_d(D), h_1(D), \ldots, h_\tau(D))$, where $\forall i \in [1, \ldots d], j \in [1, \tau]$,
$$g_i(D) = \{\sigma \mid (\sigma, \delta_i) \in D\},\ h_j(D) = \{\sigma \mid (\sigma, s_j) \in D\},$$
- we associate to any player 0's memoryless strategy s the vector $\vec{S_s} = (S_{s,1}, \ldots, S_{s,d})$, where $\forall i \in [1, d]$,
$$S_{s,i} = \{\sigma \in \mu(P), \quad s(\sigma, \pi_1(\delta_i)) = (\sigma, \delta_i)\},$$
and denote $\vec{\mathcal{S}}_s$ the vector $\mu_{|P}^{-1}(\vec{S_s})$.

Remark that $\vec{S_s}$ gives a complete characterization of s.

Lemma 47. *Given an input-free P-cut automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$, s a memoryless strategy for player 0 in $G_{\mathcal{A}}$, and $\phi(X_1, \ldots, X_n) \in MSO(\mathcal{G}_{\mathcal{A}})$, one can effectively construct an MSO-formula ϕ^g such that $\forall D_1, \ldots, D_n \subseteq \mu(P) \times Q \cup \mu(P) \times \Delta$,*

$$\mathbf{G}_{\mathcal{A}}^{s} \models \phi(D_1, \ldots, D_n) \quad \textit{iff} \quad \mu(\underline{\chi_P^{\vec{\mathcal{S}}_s}}) \models \phi^g(g(D_1), \ldots, g(D_n)).$$

Proof. Let us construct ϕ^g when ϕ is an atomic formula:

- $\forall i \in [1, d], j \in [1, \tau], \sigma, \sigma' \in \mu(P)$,
  - $E^s(\{(\sigma, \delta_i)\}, \{(\sigma', p_j)\})$ iff $\exists \epsilon \in \{0, 1\}$ s.t. $\sigma' = \sigma \star \mu(a_\epsilon)$ and $\pi_{2+\epsilon}(\delta_i) = p_j$
  - $E^s((\sigma, p_j), (\sigma', \delta_i))$ iff $\sigma' = \sigma, \pi_1(\delta_i) = p_j$ and $\sigma \in S_{s,i}$

  Then $(E^s)^g(X_1, \ldots, X_{d+\tau}, Y_1, \ldots, Y_{d+\tau})$ can be expressed in the following way: $\exists i \in [1, d],\ \exists j \in [1, \tau]$ such that
  - either $X_i = \{x\}$, $Y_{d+j} = \{y\}$ and the other ones are empty and $\exists \epsilon \in \{0, 1\}$ s.t. $y = x \star \mu(a_\epsilon)$ and $\pi_{2+\epsilon}(\delta_i) = p_j$
  - or $Y_i = \{y\}, X_{d+j} = \{x\}$ and the other ones are empty and $y = x, \pi_1(\delta_i) = p_j$ and $x \in S_{s,i}$
- $c_{s,n}^g(X_1, \ldots, X_{d+\tau})$ corresponds to the XOR of the two following properties:
  - $\exists i \in [1, d]$ such that $X_i = \{x\}$ and the other ones are empty and $c(\delta_i) = n$,
  - $\exists j \in [1, \tau]$ such that $X_{d+j} = \{x\}$ and the other ones are empty and $c(p_j) = n$.
- if $\phi(X, Y) := X \subseteq Y$,
  then $\phi^g(X_1, \ldots, X_{d+\tau}, Y_1, \ldots, Y_{d+\tau}) := \forall i \in [1, d + \tau], X_i \subseteq Y_i$.

If ϕ is not atomic, ϕ^g is given by an obvious induction on the structure of ϕ: $(\neg\phi)^s = \neg(\phi^s)$, $(\phi_1 \wedge \phi_2)^s = {\phi_1}^s \wedge {\phi_2}^s$ and $(\exists x \phi)^s = \exists x \phi^s$. □

Combining Lemmas 46 and 47, we obtain:

Lemma 48. *Let $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$ complete, input-free and P-cut, one can effectively construct an MSO-sentence* SG*, such that for every memoryless strategy* s *for player* 0 *in $G_{\mathcal{A}}$,*

$$\mu(\underline{\chi_P^{\vec{\mathcal{S}}_s}}) \models \textsc{sg} \quad \textit{iff} \quad \text{s is winning.}$$

Given $\vec{D} = (D_1, \ldots, D_d)$ any vector of subsets of $\mu(P)$, it is easy to determine if $\vec{D}$ encodes some memoryless strategy s (i.e., $\vec{D} = \vec{S_s}$).

Indeed, if we suppose that $\mathcal{A}$ is complete, it consists in checking that for every pair (σ, p) with $\sigma \in \mu(P)$, there exists one and only one $i \in [1, d]$ such that $\sigma \in D_i$, and the first component of δ_i is p, and $\chi_{\mu(P)}^{\mu(\vec{O})}(\sigma) = \pi_2(\delta_i)$. This property can easily be expressed in MSO, hence, we deduce from Lemma 48:

Lemma 49. *For every complete P-cut input-free automaton $\mathcal{A} \in \mathsf{TFA}^{\vec{O}}(A)$, there exists $d \geq 0$ and an* MSO*-formula* $\textsc{reg}_{\mathcal{A}}(X_1, \ldots, X_d)$ *such that $\forall \vec{S} = (S_1, \ldots, S_d), S_i \subseteq \mu(P)$, the following properties are equivalent:*

1. $\mu(\underline{\chi_P^{\vec{O}}}) \models \textsc{reg}_{\mathcal{A}}(S_1, \ldots, S_d)$
2. *$\vec{S}$ encodes a winning memoryless strategy for player 0 in $G_{\mathcal{A}}$.*

As seen in the proof of Lemma 42, each winning strategy for player 0 corresponds to an accepting run in $\mathcal{A}$. By using the encoding of a memoryless strategy as an oracle vector, we can construct a deterministic tree automaton accepting this run. It suffices to keep the transitions of the initial automaton, and using the oracles, restrict them to transitions indicated by the strategy. Given $\mathcal{A}$ and any memoryless strategy s, let $\mathcal{A}_s = (Q, A, \vec{\mathcal{S}}_s, \Delta_s, q_0)$ be the **deterministic** input-free tree automaton, where Δ_s is constructed in the following way:

- $\forall \vec{o}, (q_\perp, \vec{o}, q_\perp, q_\perp) \in \Delta_s$
- if $\delta_i = (q, \vec{o}, p_0, p_1) \in \Delta$, then $(q, \vec{b}, p_0, p_1) \in \Delta_s, \forall \vec{b} \in \{0, 1\}^d$ such that $b_i = 1$ and $\forall j \neq i, b_j = 0$ if the first component of δ_j is q.

This automaton follows the transitions of $\mathcal{A}$ indicated by the strategy s. The following result is then straightforward:

Lemma 50. *For every winning memoryless strategy* s *for player* 0*, $\mathcal{A}_s$ is deterministic, P-cut and its unique run is a successful run of $\mathcal{A}$.*

Since $\mathcal{A}_s$ is deterministic, Proposition 36 implies that its unique run is a $\vec{S}_s$-regular tree. Applying Lemma 35, we extend Lemmas 49 and 50 to the case of automata with inputs. In addition, from Lemma 29, if P is MSO-definable in $\underline{\chi^{\vec{O}}_{A^*}}$, for every $\mathcal{B} \in \mathsf{TFA}^{\vec{O}}(A)$, there exists a P-cut automaton $\mathcal{A}$ in $\mathsf{TFA}^{\vec{O}}(A)$ such that $\mathrm{F}(\mathcal{A})_{|P} = \mathrm{F}(\mathcal{B})_{|P}$. Hence, when P is MSO-definable in $\underline{\chi^{\vec{O}}_{A^*}}$, Lemmas 49 and 50 can be extended to every automaton in $\mathsf{TFA}^{\vec{O}}(A)$. The following proposition summarizes these results.

Proposition 51. *For every forest $F \in \mathsf{TREG}^{\vec{O}}_P(A, \Sigma)$, where P is MSO-definable in $\underline{\chi^{\vec{O}}_{A^*}}$, there exists $d \geq 0$ and a formula $\mathrm{REG}_F(X_1, \ldots, X_d)$, such that*

1. *$F \neq \emptyset$ iff $\mu(\underline{\chi^{\vec{O}}_P}) \models \exists X_1, \ldots, X_d \cdot \mathrm{REG}_F(X_1, \ldots, X_d)$*
2. *for every $\vec{S} = (S_1, \ldots, S_d)$, $S_i \subseteq \mu(P)$, if $\mu(\underline{\chi^{\vec{O}}_P}) \models \mathrm{REG}_F(\vec{S})$, then F contains a ${\mu_{|P}}^{-1}(\vec{S})$-regular tree.*

Complexity analysis of Lemma 49: Let $\mathcal{A}$ be an input-free P-cut automaton in $\mathsf{TFA}^{\vec{O}}(A)$. Using [24, Section 12.2], the formula WIN constructed in Lemma 46 contains 4 quantifier alternations (and begin with $\exists$) and its length is linear (in the number of states). The formula SG obtained in Lemma 48 has then 4 quantifier alternations and has length $\mathcal{O}(\tau)$, where τ is the number of states of $\mathcal{A}$. Finally, the transformation of SG in $\mathrm{REG}_{\mathcal{A}}$ (Lemma 49) adds existential quantifiers at the beginning of the formula, hence $\mathrm{REG}_{\mathcal{A}}$ has 4 quantifier alternations and its length is again $\mathcal{O}(\tau)$.

Complexity analysis of Proposition 51: Let $F \in \mathsf{TREG}^{\vec{O}}_P(A, \Sigma)$. Suppose that τ is the number of states of an automaton recognizing F and τ' is the number of states of the word-automaton recognizing P. Using Lemma 29 we construct in time $\mathcal{O}(|\Sigma| \cdot (\tau \cdot \tau')^{\tau \cdot \tau'})$ a P-cut automaton $\mathcal{A}$ having $\tau \cdot \tau'$ states and such that $\mathrm{F}(\mathcal{A})_P = F_{|P}$. This automaton can be transformed in an input-free automaton having $|\Sigma|^{|A|} \cdot \tau \cdot \tau'$ states.
Finally, using the complexity analysis of Lemma 49, the formula REG_F defined in Proposition 51 contains 4 quantifier alternations and its length is $\mathcal{O}(|\Sigma|^{|A|} \cdot \tau \cdot \tau')$. This formula can be constructed in time $\mathcal{O}(|\Sigma| \cdot (\tau \cdot \tau')^{\tau \cdot \tau'})$.

3.2. Transfer theorems

We use Proposition 51 to transfer some properties of the structure $\mu(\chi^{\vec{O}}_P)$ toward the structure $\underline{\chi^{\vec{O}}_P}$. The following definition fixes hypothesis for which these results hold.

Definition 52 (*Transfer Hypothesis (TH)*)**.** We write $\mathrm{TH}(\mu_{|P}, \vec{O})$ whenever:

$\mu : A^* \to M$ is a surjective semi-group morphism, P is a prefix closed language in A^*, $P \in \mathsf{REG}^{\vec{O}}(A)$ and there exists a vector $\vec{R}$ of subsets of $\mu(P)$ such that $\vec{O} = {\mu_{|P}}^{-1}(\vec{R})$.

Theorem 53 (*Transfer of Decidability*)**.** *Let μ be a morphism from A^* to any semi-group, $P \subseteq A^*$ be a prefix closed language, and $\vec{O}$ a vector of subsets of P, such that* $\mathrm{TH}(\mu_{|P}, \vec{O})$.
If the MSO*-theory of $\mu(\underline{\chi^{\vec{O}}_P})$ is decidable, then the* MSO*-theory of $\underline{\chi^{\vec{O}}_P}$ is decidable.*

Proof. Suppose that the MSO-theory of $\mu(\underline{\chi^{\vec{O}}_P})$ is decidable. For all $F \in \mathsf{TREG}^{\vec{O}}_P(A, \Sigma)$, one can decide whether $\mu(\underline{\chi^{\vec{O}}_P}) \models \exists \vec{X}, \mathrm{REG}_F(\vec{X})$ where REG_F is the formula established in Proposition 51, i.e., whether F is empty. Hence, from Corollary 25, the MSO-theory of $\underline{\chi^{\vec{O}}_P}$ is decidable. □

Complexity analysis
Let $\mathcal{C}_D(n, \tau)$ be the time needed to decide the truth value, in $\underline{\chi^{\vec{O}}_P}$, of an S2S-sentence of length n and having τ quantifier alternations. Suppose that P is recognized by an automaton having τ_P states. Let ϕ be an S2S-sentence of length n and having τ quantifier alternations. From Remark 24, we can compute a tree automaton $\mathcal{A} \in \mathsf{TREG}^{\vec{O}}$ such that $T(\phi) = T(\mathcal{A})$, and having $n \uparrow \tau$ states. Then, the formula $\mathrm{REG}_{T(\mathcal{A})}$ has 3 quantifier alternations and length $|\Sigma|^{|A|}(n \uparrow \tau)\tau_P$ and is constructed in time $\mathcal{O}(|\Sigma|((n \uparrow \tau)\tau_P)^{(n \uparrow \tau)\tau_P})$. Finally, we decide if ϕ is true in time $\mathcal{C}_D(3, |\Sigma|^{|A|}(n \uparrow \tau)\tau_P) + \mathcal{O}(|\Sigma|((n \uparrow \tau)\tau_P)^{(n \uparrow q)\tau_P})$ (or $\mathcal{C}_D(3, (n \uparrow \tau))$ if $P = A^*$).

We define now a condition on $\mu_{|P}$ allowing to transfer the selection property (see Definition 32).

Definition 54 (*MSO-invertibility*)**.** Given any surjective morphism μ from A^* into some semi-group, and $P \subseteq A^*$ prefix closed, the restricted map $\mu_{|P}$ is said to be MSO**-invertible** if for every $\vec{O}$, vector of subsets of P, and every $D \subseteq \mu(P)$,

if D is MSO-definable in $\mu(\underline{\chi^{\vec{O}}_P})$ then ${\mu_{|P}}^{-1}(D)$ is MSO-definable in $\underline{\chi^{\vec{O}}_P}$.

Theorem 55 (*Transfer of Selection Property*)**.** *If* $\mathrm{TH}(\mu_{|P}, \vec{O})$ *and $\mu_{|P}$ is* MSO*-invertible, then $\mu(\underline{\chi^{\vec{O}}_P})$ satisfies* SP *implies $\underline{\chi^{\vec{O}}_P}$ satisfies* SP.

Proof. Let F be a non-empty $\vec{O}$-regular forest in P-Tree$_n$, from Proposition 51 and since $\mu(\underline{\chi_P^{\vec{O}}})$ fulfills SP, there exists $\vec{S}$ such that F contains a ${\mu_{|P}}^{-1}(\vec{S})$-regular tree and $\vec{S}$ is MSO-definable in $\mu(\underline{\chi_P^{\vec{O}}})$. Let $\vec{D} = {\mu_{|P}}^{-1}(\vec{S})$, since $\mu_{|P}$ is MSO-invertible, $\vec{D}$ is MSO-definable in $\underline{\chi_P^{\vec{O}}}$ and F contains a $\vec{D}$-regular tree. Hence, from Proposition 33, $\underline{\chi_P^{\vec{O}}}$ fulfills SP. □

Complexity analysis
For an S2S-formula of length n and having τ quantifier alternations, we denote by $\mathcal{C}_M(n, \tau)$ the time needed to construct a formula that defines a model in $\underline{\chi_P^{\vec{O}}}$, and by $(n_\mathcal{S}, \tau_\mathcal{S})$ the size of a formula that defines inverse models. Suppose P is recognized by an automaton having τ_P states. Given an S2S-formula ϕ of length n and having τ quantifier alternations, one can construct a formula that defines a model of ϕ in time $\mathcal{C}_M(3, (n \uparrow \tau)) + \mathcal{O}((2(n \uparrow \tau) \cdot \tau_P)^{(n\uparrow\tau)\cdot\tau_P})$ (or $\mathcal{C}_M(3, (n \uparrow \tau))$ if $P = A^*$).

Theorem 56 (*Structure Theorem*). *If $\mu_{|P}$ is* MSO*-invertible,* TH$(\mu_{|P}, \vec{O})$ *and $\mu(\underline{\chi_P^{\vec{O}}})$ satisfies* SP*, then for every $L \subseteq P$, the following properties are equivalent:*

- *L is* MSO*-definable in $\underline{\chi_P^{\vec{O}}}$*
- *there exists $\vec{D}$* MSO*-definable in $\mu(\underline{\chi_P^{\vec{O}}})$ such that L is ${\mu_{|P}}^{-1}(\vec{D})$-regular.*

Proof. Let us suppose that L is MSO-defined in $\underline{\chi_P^{\vec{O}}}$ by a formula $\phi(X)$. Then from Theorem 26, there exists a $\vec{O}$-regular forest $F = \{\chi_P^L\}$ such that $F = T_P^{\vec{O}}(\phi)$. From Proposition 51 and since $\mu(\underline{\chi_P^{\vec{O}}})$ fulfills SP, there exists $\vec{D}$ such that χ_P^L is ${\mu_{|P}}^{-1}(\vec{D})$-regular and $\vec{D}$ is MSO-definable in $\mu(\underline{\chi_P^{\vec{O}}})$. From Remark 31.1, the language L is ${\mu_{|P}}^{-1}(\vec{D})$-regular.

Conversely, given $\vec{D}$, MSO-definable in $\mu(\underline{\chi_P^{\vec{O}}})$, then since ${\mu_{|P}}^{-1}$ is MSO-invertible, ${\mu_{|P}}^{-1}(\vec{D})$ is MSO-definable in $\underline{\chi_P^{\vec{O}}}$. Given L a ${\mu_{|P}}^{-1}(\vec{D})$-regular language, by using the automata-characterization of L, it is then easy to find a MSO-formula defining L in $\underline{\chi_P^{\vec{O}}}$. □

Complexity analysis
For an S2S-formula of length n and having τ quantifier alternations, we denote by $\mathcal{C}_M(n, \tau)$ the time needed to construct a formula that defines a model in $\underline{\chi_P^{\vec{O}}}$, and by $(n_\mathcal{S}, \tau_\mathcal{S})$ the size of a formula that defines inverse models. Suppose that P is recognized by an automaton having τ_P states and that L is defined by an S2S-formula ϕ of length n and having τ quantifier changes, one can construct a word oracle-automaton recognizing L having $F(3, F(n, \tau))$ states. It can be constructed in time $\mathcal{C}_M(3, n \uparrow \tau) + \mathcal{O}((2(n \uparrow \tau) \cdot \tau_P)^{(n\uparrow\tau)\cdot\tau_P})$ (or $\mathcal{C}_M(3, n \uparrow \tau)$ if $P = A^*$).

3.3. On the MSO-invertibility

Given any surjective morphism μ from A^* into some semi-group $(M, \star)$, and $P \subseteq A^*$ prefix closed, it is often difficult to prove that $\mu_{|P}$ is MSO-invertible. In this paper (Propositions 59 and 77), we will apply the following method:

1. Find a covering $\mathcal{C}$ of P such that
   (a) for all $C \in \mathcal{C}$, $\mu_{|C} : C \to M$ is bijective
   (b) the property "be an element of $\mathcal{C}$" can be expressed by an MSO-formula over $\langle P, \varepsilon, (\text{succ}_a)_{a\in A}\rangle$.
2. Find, for every MSO-formula ϕ over $\mu(\underline{\chi_P^{\vec{O}}})$ (with $\vec{O} = {\mu_{|P}}^{-1}(\vec{R})$ and $\vec{R}$ arbitrary) with n free variables, an MSO-formula ϕ' over $\underline{\chi_P^{\vec{O}}}$ with $n + 1$ free variables, satisfying for every $C \in \mathcal{C}$, for every $S_1, \ldots, S_n \subseteq P$:

$$\underline{\chi_P^{\vec{O}}} \models \phi'(S_1 \cap C, \ldots, S_n \cap C, C) \quad \text{iff} \quad \mu(\underline{\chi_P^{\vec{O}}}) \models \phi(\mu(S_1 \cap C, \ldots, S_n \cap C)). \tag{1}$$

It suffices to find ϕ' for ϕ atomic: for inclusions, the construction is trivial: $(X_1 \subseteq X_2)' := X_1 \subseteq X_2$) and $(X \subseteq R_i)' := X \subseteq O_i$. Furthermore, for boolean combinations we have obviously $(\phi \vee \psi)' := \phi' \vee \psi'$, $(\phi \wedge \psi)' := \phi' \wedge \psi'$ et $(\neg\phi)' := \neg\phi'$ and for quantified formulas, $(\exists X, \psi(X, X_1, \ldots, X_n))' := \exists X, \psi'(X \cap Y, X_1, \ldots, X_n, Y)$ and $(\forall X, \psi(X, X_1, \ldots, X_n))' := \exists X, \psi'(X \cap Y, X_1, \ldots, X_n, Y)$.

Let us prove the case of the existential quantification, the universal case is similar:
Fix $S_1, \ldots, S_n \subseteq P$ and $C \in \mathcal{C}$.

$$\begin{aligned}
\underline{\chi_P^{\vec{O}}} &\models \exists X, \psi'(X \cap C, S_1 \cap C, \ldots, S_n \cap C, C) \quad \text{iff} \\
\exists D \subseteq P,\ \underline{\chi_P^{\vec{O}}} &\models \psi'(D \cap C, S_1 \cap C, \ldots, S_n \cap C, C) \quad \text{iff (by (i.h.))} \\
\exists D \subseteq P,\ \mu(\underline{\chi_P^{\vec{O}}}) &\models \psi(\mu(D \cap C), \mu(S_1 \cap C), \ldots, \mu(S_n \cap C)) \quad \text{iff (since } \mu_{|C} \text{ is bijective)} \\
\exists D' \subseteq M,\ \mu(\underline{\chi_P^{\vec{O}}}) &\models \psi(D', \mu(S_1 \cap C), \ldots, \mu(S_n \cap C)) \quad \text{iff} \\
\mu(\underline{\chi_P^{\vec{O}}}) &\models \exists X,\ \psi(X, \mu(S_1 \cap C), \ldots, \mu(S_n \cap C)).
\end{aligned}$$

If these two steps are achieved, we can conclude in the following way: since $\mathcal{C}$ is a covering of P and for all $C \in \mathcal{C}$, $\mu_{|C}$ is bijective: $\forall D \subseteq P, D' \subseteq M, D = {\mu_{|P}}^{-1}(D')$ iff $\forall C \in \mathcal{C},\ \mu(C \cap D) = D'$.

Then, for every formula $\phi(X_1, \ldots, X_n)$ over $\mu(\underline{\chi_P^{\vec{O}}})$, the formula

$$\phi_\mu(X_1, \ldots, X_n) := \forall C \in \mathcal{C},\ \phi'(C \cap X_1, \ldots, C \cap X_n, C)$$

fulfills $\underline{\chi_P^{\vec{O}}} \models \phi_\mu(\vec{D})$ iff $\exists \vec{D}'$ such that $\mathcal{S} \models \phi(\vec{D}')$ and ${\mu_{|P}}^{-1}(\vec{D}) = D'$.

3.4. A first application

Various authors have exhibited classes of relations $R \subseteq \mathbb{N}$ for which the structure $\langle \mathbb{N}, +1, R\rangle$ has a decidable MSO-theory (see for recent examples [27,15,10]). We show how to use our transfer theorems from such a structure. Given two alphabets A and B, with $B \subseteq A$, we consider the map $l_B : A^* \to \mathbb{N}$ associating to every word in A^* its number of occurrences of letters in B.

Proposition 57. *For every $\vec{N} = (N_1, \ldots, N_m)$, $N_i \subseteq \mathbb{N}$ such that $\langle \mathbb{N}, +1, \vec{N}\rangle$ has a decidable* MSO*-theory, the structure $\langle A^*, (\mathrm{succ}_a)_{a\in A}, {l_B}^{-1}(\vec{N})\rangle$ has a decidable* MSO*-theory.*

Proof. The map l_B is a surjective morphism when $\mathbb{N}$ is endowed with the "+" operator. Let $\vec{O} = {l_B}^{-1}(\vec{N})$, since $l_B(\underline{\chi_{A^*}^{\vec{O}}}) = \langle \mathbb{N}, +1, \vec{N}\rangle$, Proposition 57 is a direct consequence of Theorem 53. □

This result has already been proved in [28][Proposition 2] for the case $A = B$, as a direct application of the results about unfolding of graphs obtained in [29,30]. However, to our knowledge, this method does not allow the transfer of the selection property, nor to deal with the decidability for the case $B \neq A$. Conversely, Theorem 53 does not seems to cover all results that can be obtained by unfolding, since some graph unfoldings are not included in a semi-group.

Theorem 58 (*[10]*). *For all vector $\vec{N}$ of subsets of $\mathbb{N}$, the structure $\langle \mathbb{N}, 0, +1, \vec{N}\rangle$ satisfies SP.*

Proposition 59. *The map $l_B : A^* \to \mathbb{N}$ is MSO-invertible.*

Proof. We follow the schema described in Section 3.3:

1. We associated to each infinite word $u = u_0b_0u_1b_1 \ldots u_nb_nu_{n+1} \ldots \in A^\omega$ with $u_i \in (A - B)^*$ and $b_i \in B$ for all i, the set $C_u = \{u_0, u_0b_0u_1, u_0b_0u_1b_1u_2, \ldots\}$. Consider the family $\mathcal{C}$ of all C_u, for $u \in A^\omega$: $\mathcal{C}$ is a covering of A^* and for all $C \in \mathcal{C}$, the restriction of l_B to C is a bijective map from C to $\mathbb{N}$. In addition, the property "be an element of $\mathcal{C}$" can easily be expressed by an MSO-formula over $\langle A^*, \varepsilon, (\mathrm{succ}_a)_{a\in A}\rangle$.
2. For each atomic formula ϕ, we can find a formula ϕ' fulfilling the equivalence 1 (we replace relations with free first order variables by "equivalent" relations having free second order variables. For example, $\varepsilon(x)$ is replaced by $\varepsilon(X) == \exists x.\ X = \{x\} \wedge \varepsilon(x)$):
   - $0'(X, Y) := X \subseteq (A - B)^*$,
   - $(+1)'(X_1, X_2, Y) := \exists u, v,\ X_1 = \{u\} \wedge X_2 = \{v\} \wedge v \in uB(A - B)^*$. □

Combining Theorems 55, 56 and 58 and Proposition 59 proves the following result.

Corollary 60. *Given a vector $\vec{N}$ of subsets of $\mathbb{N}$,*

1. *the structure $\langle A^*, (\mathrm{succ}_a)_{a\in A}, {l_B}^{-1}(\vec{N})\rangle$ satisfies SP,*
2. *a set L is MSO-definable in $\langle A^*, (\mathrm{succ}_a)_{a\in A}, {l_B}^{-1}(\vec{N})\rangle$ iff there exists a vector $\vec{D}$ of sets MSO-definable in $\langle \mathbb{N}, 0, +1, \vec{N}\rangle$ such that $L \in \mathsf{REG}^{{l_B}^{-1}(\vec{D})}$.*

4. Words, iterated-pushdown stores and tree-structures

In this section, we apply results obtained previously to define the notion of regularity for sets of k-pushdown store that naturally extends what is known for classical pushdown stores (level 1). At level 1, a set of pushdown store is regular iff it is generated by a pushdown automaton (i.e., it is the set of stacks appearing in the reachable final configurations of a pushdown automaton) iff ii is a regular word language.

This section is split as follows: first we give some preliminary definitions on iterated pushdown stores; then we show that iterated pushdown stores can be seen as words in a free group; then, using our transfer theorems, we prove some MSO-properties on the structure associated with the free group; finally, we translate these result in terms of set of iterated pushdown store.

4.1. Iterated pushdown stores

4.1.1. The storage structure

Originally defined by Greibach in [31], iterated pushdown stores are storage structures built iteratively. Let $(A_k)_{k\geq 1}$ be disjoint and finite alphabets. For all $k \geq 1$, we denote by $A_{1,k}$ the finite sequence $A_1, \ldots, A_k$ and adopt the convention that $\bot$ is a special symbol and that $\{\bot, [,]\} \cap A_i$ is empty for all $i \geq 1$.

Definition 61. We define inductively the set k-pds$(A_{1,k})$ (or k-pds when store alphabets are understood) of k-iterated pushdown-stores over $A_{1,k}$:
1-pds$(A_{1,1}) = A_1{}^*\{\bot\}$,
$(k+1)$-pds$(A_{1,k+1}) = (A_{k+1}[k\text{-pds}(A_{1,k})])^* \bot [k\text{-pds}(A_{1,k})]$,
it-pds$((A_k)_{k\geq 1}) = \bigcup_{k\geq 1} k$-pds$(A_{1,k})$.

We denote by $\bot_k$ the "empty" k-pds containing only symbols $\bot$: $\bot_1{=}\bot$ and $\bot_{k+1}{=}\bot\,[\bot_k]$.

From the definition, every ω in $(k+1)$-pds$(A_{1,k+1})$, $k \geq 0$, has a unique decomposition as $\omega = a[\omega_1]\omega'$ with $\omega_1 \in k$-pds$(A_{1,k})$, $\omega' \in (k+1)$-pds$(A_{1,k+1}) \cup \{\varepsilon\}$ and $a \in A_{k+1} \cup \{\bot\}$. Furthermore, $a = \bot$ iff $\omega' = \varepsilon$.

Example 62. Let $A_1 = \{a_1, b_1\}$, $A_2 = \{a_2, b_2\}$, $A_3 = \{a_3\}$ be storage alphabets, and $\omega_{ex} = a_3[\underbrace{b_2[b_1a_1\ \bot]a_2[a_1\ \bot]\ \bot_2}_{\omega_1}]$ $\underbrace{\bot\ [a_2[a_1\ \bot]a_2[\bot]\ \bot_2]}_{\omega'} \in 3\text{-pds}(A_{1,3})$.

The decomposition of ω_{ex} is $\omega_{ex} = a_3[\omega_1]\omega'$.

The two following maps will be useful.

Projection: for all $k \geq 1$, the map $\mathrm{p}_{k,i}$ associating a k-pds to its top i-pds (for $1 \leq i \leq k$) is defined inductively: $\forall \omega = a[\omega_1]\omega' \in k\text{-pds}(A_{1,k})$,

$$\mathrm{p}_{k,k}(\omega) = \omega \quad \text{and} \quad \mathrm{p}_{k,i}(\omega) = \mathrm{p}_{k-1,i}(\omega_1) \quad \text{if } 1 \leq i \leq k-1.$$

The double subscript notation will be used to handle inverse projections, the rest of the time, we will note p_i instead of $\mathrm{p}_{k,i}$.

Top symbols: for all $k \geq 1$, the map top associating any k-pds, to the word formed by its k top-symbols is defined inductively: $\forall \omega = a[\omega_1]\omega' \in k\text{-pds}(A_{1,k})$ by

$$\mathrm{top}(\omega) = a, \quad \text{if } k = 1, \text{ else } \mathrm{top}(\omega) = a \cdot \mathrm{top}(\omega_1).$$

Remark that $\mathrm{top}(\omega) \in (A_k \cup \{\bot\}) \cdots (A_2 \cup \{\bot\})(A_1 \cup \{\bot\})$. For $i \in [1, k]$, and $\omega \in k$-pds, we denote by $\mathrm{top}_i(\omega)$ the $(k-i+1)$-th letter of $\mathrm{top}(\omega)$, i.e., the top symbol of level i.

Example 63. Let ω_{ex} be the 3-pds given in Example 62: $\mathrm{p}_2(\omega_{ex}) = b_2[b_1a_1\ \bot]a_2[a_1\ \bot]\ \bot_2$, $\mathrm{p}_1(\omega_{ex}) = b_1a_1\ \bot$, and $\mathrm{top}(\omega_{ex}) = a_3b_2b_1$, $\mathrm{top}(\mathrm{p}_2(\omega_{ex})) = b_2b_1$, $\mathrm{top}(\mathrm{p}_1(\omega_{ex})) = b_1$.

An **instruction** on it-pds is a function from it-pds to it-pds which does not modify the level of the store (i.e., if instr is an instruction then for any $k \geq 1$ and any $\omega \in k$-pds, $\mathrm{instr}(\omega) \in k$-pds). An **instruction of level** i is an instruction instr which does not modify the levels greater than i of any it-pds and satisfies:
if $\omega = a[\omega_1]\omega' \in k$-pds, $k > i$, then $\mathrm{instr}(\omega) = a[\mathrm{instr}(\omega_1)]\omega'$,
if $\omega \in k$-pds, $k < i$, then $\mathrm{instr}(w) = w$.
Therefore, to define an instruction of level i, there is only need to define it for any $\omega \in i$-pds.

Three instructions of level k are generally applicable to it-pushdowns.

Definition 64. Let $1 \leq i \leq k$, "classical" instructions of level i over $A_{1,k}$ are defined for every $\omega = b[\omega_1]\omega' \in i$-pds$(A_{1,i})$ by:

$\mathrm{pop}_i(\omega) = \omega'$ if $b \neq \bot$, else $\mathrm{pop}_i(\omega)$ is undefined,
$\mathrm{push}_{i,a}(\omega) = a[\omega_1]\omega$,
$\mathrm{change}_{i,a}(\omega) = a[\omega_1]\omega'$, if $b \neq \bot$ else $\mathrm{change}_{i,a}(\omega)$ is undefined.

For $k \geq 1$, $\mathcal{I}_k(A_{1,k}) = \{\mathrm{pop}_i \mid i \in [1,k]\} \cup \{\mathrm{push}_{i,a}, \mathrm{change}_{i,a} \mid a \in A_i, i \in [1,k]\}$. is the set of instructions over $A_{1,k}$.

Thus, given $\omega \in k$-pds and $i \leq k$, $\mathrm{pop}_i(\omega)$ erases $\mathrm{p}_i(\omega)$ on the top of the store, $\mathrm{push}_{i,a_i}(\omega)$ adds $a_i[\mathrm{p}_{i-1}(\omega)]$ on the top of the top i-pds and $\mathrm{change}_{i,a_i}(\omega)$ replaces $\mathrm{top}_i(\omega)$ by a_i.

Example 65. Let $\omega = b_3[b_2[b_1\ \bot]\ \bot_2]\ \bot_3$ be a 3-pds,
$\mathrm{pop}_3(\omega) = \bot_3$, $\mathrm{pop}_2(\omega) = b_3[\bot_2]\ \bot_3$, $\mathrm{pop}_1(\omega) = b_3[b_2[\bot]\ \bot_2]\ \bot_3$,
$\mathrm{push}_{2,a_2}(\omega) = b_3[a_2[b_1\ \bot]b_2[b_1\ \bot]\ \bot_2]\ \bot_3$, $\mathrm{push}_{1,a_1}(\omega) = b_3[b_2[a_1b_1\ \bot]\ \bot_2]\ \bot_3$,
$\mathrm{change}_{3,a_3}(\omega) = a_3[b_2[b_1\ \bot]\ \bot_2]\ \bot_3$, $\mathrm{change}_{1,a_1}(\omega) = b_3[b_2[a_1\ \bot]\ \bot_2]\ \bot_3$.

We also define the inverse instruction of $\text{push}_{i,a}$ which will be used to encode the k-pushdowns as words.

Definition 66. For any $i \geq 1$ and $a \in A_i$, the instruction of level i $\overline{\text{push}}_{i,a}$ is defined for any $\omega \in i\text{-pds}(A_{1,i})$ by

$\overline{\text{push}}_{i,a}(\omega) = \omega'$ if there exists $\omega' \in i\text{-pds}$ such that $\omega = \text{push}_{i,a}(\omega')$
$\overline{\text{push}}_{i,a}(\omega)$ is undefined otherwise.

In other words, $\forall \omega \in k\text{-pds}$,

$$\overline{\text{push}}_{k,a}(\omega) = \omega' \quad \text{iff} \quad \omega = a[\omega_1]b[\omega_1]\omega'' \text{ and } \omega' = b[\omega_1]\omega''.$$

Remark that $\forall \omega \in k\text{-pds}$, $\overline{\text{push}}_{k,a}(\text{push}_{k,a}(\omega)) = \omega$ and if $\overline{\text{push}}_{k,a}(\omega)$ is defined then $\text{push}_{k,a}(\overline{\text{push}}_{k,a}(\omega)) = \omega$.

4.1.2. Iterated pushdown machines

We define here *controlled iterated pushdowns systems* which extend systems with iterated storage structure intensively studied in the 70's (see [32,31,33,34]) and more recently in [35–37,18,16,17,38,39,15]. Here we define iterated pushdown machines whose transitions are conditioned by membership tests on the store.

Definition 67 (*Controlled k-pushdown Transitions System*)**.** Let $k \geq 0$, a k-TS is a structure $\mathfrak{A} = (Q, A_{1,k}, \vec{C}, \Delta, q_0, F)$ where Q is a finite set of states, $A_{1,k}$ is the sequence of pushdown alphabets, $\vec{C} = (C_1, \ldots, C_m)$ is a vector of *controllers* $C_i \subseteq k\text{-pds}(A_{1,k})$, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states and $\Delta \subseteq Q \times \text{top}(k\text{-pds}(A_{1,k})) \times \{0,1\}^m \times \mathcal{I}_k(A_{1,k}) \times Q$ is a finite set of transitions.

The family of all k-TS controlled by $\vec{C}$ is $k\text{-TS}^{\vec{C}}(A_{1,k})$. The set of *configurations* of $\mathfrak{A}$ is $Con_{\mathfrak{A}} = Q \times k\text{-pds}(A_{1,k})$. The single step relation $\rightarrow_{\mathfrak{A}} \subseteq Con_{\mathfrak{A}} \times Con_{\mathfrak{A}}$ of $\mathfrak{A}$ is defined by

$$(p, \omega) \rightarrow_{\mathfrak{A}} (q, \omega') \quad \text{iff} \quad (p, \text{top}(\omega), \chi_{\vec{C}}(\omega), \mathit{instr}, q) \in \Delta, \text{ and } \omega' = \mathit{instr}(\omega).$$

We denote by $\overset{*}{\rightarrow}_{\mathfrak{A}}$ the reflexive and transitive closure of $\rightarrow_{\mathfrak{A}}$. The *set of k-pds generated by* $\mathfrak{A}$ is $\text{P}(\mathfrak{A}) = \{\omega \in k\text{-pds}(A_{1,k}) \mid \exists q \in F,\ (q_0, \bot_k) \overset{*}{\rightarrow}_{\mathfrak{A}} (q, \omega)\}$.

4.1.3. Iterated pushdown structures

We define the structure PDS_k associated to the type k-pushdowns. It is proved in [15], that for all $k \geq 1$, the MSO-theory of this structure is decidable.

Definition 68 (*k-pds Structure*)**.** Given $k \geq 1$, $\text{PDS}_k(A_{1,k})$ is the structure whose domain is $k\text{-pds}(A_{1,k})$ and endowed with the binary relations pop_i, $\text{push}_{i,a}$ and $\text{change}_{i,a}$ for every $1 \leq i \leq k$, $a \in A_i$. Relations pop_i, $\text{push}_{i,a}$ and $\text{change}_{i,a}$ are graphs of the corresponding instructions on pushdowns.

4.2. Iterated-pushdowns viewed as words

In order to apply results obtained above to iterated pushdowns, we now represent k-pds as words in a free group.

4.2.1. Free group

Given a finite alphabet A , let us associate to each $a \in A$ the inverse symbol $\bar{a}$ (with $\bar{a} \notin A$). We denote by $\overline{A}$ the set of inverse letters of A and define $\widehat{A} = A \cup \overline{A}$. For every $u = a_1 \cdots a_n \in \widehat{A}^*$, the **inverse** word of u is $\overline{u} = b_n \cdots b_1$ where each b_i is the inverse symbol of a_i.

Let us then consider the reduction system $S = \{(a\bar{a}, \varepsilon), (\bar{a}a, \varepsilon)\}$. A word in $\widehat{A}^*$ is said to be **reduced** if it is S-reduced, i.e., it does not contain occurrences of $a\bar{a}$ or $\bar{a}a$, for $a \in A$. We denote by $\text{Irr}(A)$ the set of reduced words in $\widehat{A}^*$. As S is confluent, each word w is equivalent (mod $\leftrightarrow_S^*$) to a unique reduced word denoted $\rho(w)$.
We define the free group $(\text{Irr}(A), \bullet, \varepsilon)$, where $\forall u, v \in \text{Irr}(A)$, $u \bullet v = \rho(u \cdot v)$.

4.2.2. Encoding

We encode each $\omega \in k\text{-pds}(A_{1,k})$ by a word representing the smallest instructions sequence of $\text{push}_{i,a}$ and $\overline{\text{push}}_{i,a}$ computing ω from $\bot_k$. The set of such encodings is a prefix closed language $\mathcal{P}_k$ over the alphabet $\widehat{A_{1,k}}$.

Every $\omega \in k\text{-pds}(A_{1,k})$ can be represented by a word on $\widehat{A_{1,k}} = \overline{A_{1,k}} \cup A_{1,k}$ (with the convention that $a \in A_{1,k}$ if $a \in A_1 \cup \ldots A_k$) encoding an instructions sequence computing ω from $\bot_k$:

- every $a \in A_i$ corresponds to $\text{push}_{i,a}$
- every $\bar{a} \in \overline{A}_i$ corresponds to $\overline{\text{push}}_{i,a}$

For instance, the 2-pds $\omega = a_2[c_1 b_1\ \bot]a_2[a_1\ \bot]\ \bot\ [\bot]$ can be represented by the word $u_1 = a_2 a_1 a_2 \bar{a}_1 b_1 c_1$, or by $u_2 = a_2 a_1 b_2 \bar{b}_2 a_2 \bar{a}_1 b_1 c_1$, or by $u_3 = a_2 a_1 a_1 b_2 \bar{b}_2 \bar{a}_1 a_2 \bar{a}_1 b_1 c_1$.

There are then several representatives of the same k-pds but all have the same reduced representative in $(\mathrm{Irr}(A_{1,k}), \bullet, \varepsilon)$. Each k-pds will be encoded by its reduced representative. In the previous example, the reduced representative is u_1 (since $\rho(u_1) = \rho(u_2) = \rho(u_3) = u_1$).

Each word in $\widehat{A_{1,k}}^*$ does not define a *valid* sequence of instructions. For example, $a_1b_1a_2\bar{b}_1\bar{b}_1$ is not valid since $a_1b_1a_2\bar{b}_1$ correspond to $a_2[a_1\ \bot]\ \bot\ [b_1a_1\ \bot]$ and $\overline{\mathrm{push}}_{b_1,1}$ is then undefined.

Let us introduce the set $\mathcal{M}_k$ of words in $\widehat{A_{1,k}}^*$ encoding all valid sequences of moves, as well as the set $\mathcal{P}_k$ of reduced words of $\mathcal{M}_k$ which encodes the set of k-pds. We define simultaneously $\mathcal{P}_k(A_{1,k})$ (or simply $\mathcal{P}_k$ when the A_i's are fixed) and $\mathcal{M}_k(A_{1,k})$ (or simply $\mathcal{M}_k$) by induction on k:

- $\mathcal{P}_0 = \{\varepsilon\}$,
- $\forall k \geq 0$, $\mathcal{M}_k(A_{1,k}) = \{u \in \widehat{A_{1,k}}^* \mid \forall v \preccurlyeq u,\ \rho(v) \in \mathcal{P}_k(A_1, \ldots, A_k)\}$ and
  $\mathcal{P}_{k+1}(A_{1,k+1}) = \{u \in \rho((\widehat{A_{1,k}} \cup A_{k+1})^*) \mid \pi_{\widehat{A_{1,k}}}(u) \in \mathcal{M}_k(A_1, \ldots, A_k)\}$.

Clearly, $\mathcal{P}_1(A_1) = A_1^*$.

We now gives some definition and properties on $\mathcal{P}_k(A_{1,k})$ and its links with $\mathcal{P}_k(A_{1,k})$ which will be useful in the next subsections.

Definition 69 (*Projection*)**.** For $k \geq 0$, $\mathrm{f}_k : \mathcal{P}_{k+1} \to \mathcal{P}_k$ is defined for every $u \in \mathcal{P}_{k+1}$ by $\mathrm{f}_k(u) = \rho(\pi_{\widehat{A_{1,k}}}(u))$. We extend f_k by $\mathrm{f}_{i+1,k} : \mathcal{P}_{i+1} \to \mathcal{P}_k$ obtained by successive applications of $\mathrm{f}_i, \mathrm{f}_{i-1}, \ldots, \mathrm{f}_k$.

An obvious induction on k proves the following recursive definition of $\mathcal{P}_k$

Proposition 70. *For every $k \geq 1$, $u \in \mathcal{P}_k$ and $a \in A_i$, $1 \leq i \leq k$, $u \bullet a \in \mathcal{P}_k$ and $[u \bullet \bar{a} \in \mathcal{P}_k$ iff $\mathrm{f}_{k,i}(u) \in \mathcal{P}_i \cdot a]$.*

For every $k \geq 0$, there is a bijection, denoted φ_k, between $\mathcal{P}_k$ and k-pds:

Definition 71. The map $\varphi_k : k\text{-pds}(A_{1,k}) \to \mathcal{P}_k(A_{1,k})$ is defined by induction on $k \geq 0$:

- $\varphi_0(\bot_0) = \varepsilon$,
- $\forall k \geq 0$, $\omega_1 \in k\text{-pds}(A_{1,k})$, $\omega \in (k+1)\text{-pds}(A_{1,k+1})$ and $a \in A_{k+1}$,
  - $\varphi_{k+1}(\bot\ [\omega_1]) = \varphi_k(\omega_1)$
  - $\varphi_{k+1}(a[\omega_1]\omega) = (\varphi_{k+1}(\omega) \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega))}) \bullet \varphi_k(\omega_1)$.

Example 72. Consider the 3-pds $\omega_{ex} = a_3[b_2[b_1a_1\ \bot]a_2[a_1\ \bot]\ \bot_2]\ \bot\ [a_2[a_1\ \bot]a_2[\bot]\ \bot_2] = a_3[\omega_1]\omega$. Applying φ_3, we get $\varphi_3(\omega_{ex}) = (\varphi_3(\omega)a_3\overline{f_2(\varphi_3(\omega))}) \bullet \varphi_2(\omega_1)$.
We have, $\varphi_2(b_2[b_1a_1\ \bot]a_2[a_1\ \bot]\ \bot_2) = a_2a_1b_2b_1$ and $\varphi_2(a_2[a_1\ \bot]a_2[\bot]\ \bot_2) = a_2a_2a_1$, then $\varphi_3(\omega) = a_2a_2a_1$. We obtain then,

$$\varphi_3(\omega_{ex}) = a_2a_2a_1a_3\overline{a_2a_2a_1} \bullet (a_2a_1b_2b_1) = a_2a_2a_1a_3(\bar{a_1}\,\bar{a_2}\,\bar{a_2}) \bullet (a_2a_1b_2b_1) = a_2a_2a_1a_3\bar{a_1}\,\bar{a_2}a_1b_2b_1.$$

Proposition 73. *For every $(k+1)$-pds $\omega = a[\omega_1]\omega'$, $\varphi_k(\omega_1) = \mathrm{f}_k(\varphi_{k+1}(\omega))$.*

Proof. From definition of φ_k and f_k:

$$\mathrm{f}_k(\varphi_{k+1}(a[\omega_1]\omega')) = \mathrm{f}_k(\varphi_{k+1}(\omega') \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega'))}) \bullet \varphi_k(\omega_1) = \mathrm{f}_k(\varphi_{k+1}(\omega')) \bullet \overline{\mathrm{f}_k(\varphi_{k+1}(\omega'))} \bullet \varphi_k(\omega_1) = \varphi_k(\omega_1). \quad \square$$

Lemma 74. *For every $k \geq 0$, φ_k is bijective.*

Proof. Injectivity: Let $\omega = a[\omega_1]\omega_2$, $\omega' = a'[\omega_1']\omega_2'$ and suppose that $\varphi_{k+1}(\omega) = \varphi_{k+1}(\omega')$. From the definition of φ_{k+1}, we have then

$$\varphi_{k+1}(\omega_2) \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega_2))} \bullet \varphi_k(\omega_1) = \varphi_{k+1}(\omega_2') \cdot a' \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega_2'))} \bullet \varphi_k(\omega_1').$$

Then $a = a'$, $\varphi_{k+1}(\omega_2) = \varphi_{k+1}(\omega_2')$ and $\overline{\mathrm{f}_k(\varphi_{k+1}(\omega_2))} \bullet \varphi_k(\omega_1) = \overline{\mathrm{f}_k(\varphi_{k+1}(\omega_2'))} \bullet \varphi_k(\omega_1')$

By induction on the length of the pds, we obtain $\omega_2 = \omega_2'$, then $\overline{\mathrm{f}_k(\varphi_{k+1}(\omega_2))} \bullet \varphi_k(\omega_1) = \overline{\mathrm{f}_k(\varphi_{k+1}(\omega_2))} \bullet \varphi_k(\omega_1')$ and since $(\mathrm{Irr}(A_{1,k}), \bullet, \varepsilon)$ is a group, $\varphi_k(\omega_1) = \varphi_k(\omega_1')$. By induction on k, $\omega_1 = \omega_1'$ and then $\omega = \omega'$.
Surjectivity: We detail the case where $u = u'au_1 \in \mathcal{P}_{k+1}$. By induction on k and induction on the length of the word, we get ω_1 and ω' such that $\varphi_k(\omega_1) = \mathrm{f}_k(u)$ and $\varphi_{k+1}(\omega') = u'$. Then let $\omega = a[\omega_1]\omega'$, we have:

$$\begin{aligned}\varphi_{k+1}(\omega) &= \varphi_{k+1}(\omega') \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega'))} \bullet \varphi_k(\omega_1)\\ &= u' \cdot a \cdot \overline{\mathrm{f}_k(u')} \bullet \mathrm{f}_k(u) = u' \cdot a \cdot \overline{\mathrm{f}_k(u')} \bullet \mathrm{f}_k(u') \bullet u_1 = u' \cdot a \cdot u_1. \quad \square\end{aligned}$$

We conclude this section by emphasizing connections between the right-product in $\mathcal{P}_k$ and the application of instructions to k-pushdowns. The next lemma follows directly from the definition of φ_k.

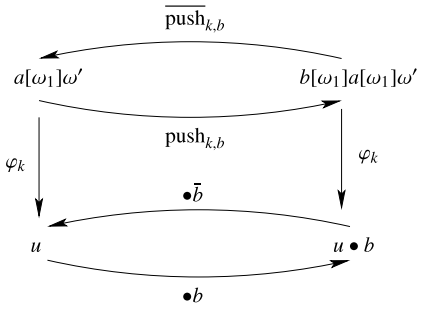

Fig. 4. Correspondence instruction/right-product.

Lemma 75. *For every $k \geq 1$, $u, v \in \mathcal{P}_k$, and $a \in A_i$, $1 \leq i \leq k$,*

$v = u \bullet a$ *iff* ${\varphi_k}^{-1}(v) = \text{push}_{i,a}({\varphi_k}^{-1}(u))$
$v = u \bullet \bar{a}$ *iff* ${\varphi_k}^{-1}(u) = \text{push}_{i,a}({\varphi_k}^{-1}(v))$ *iff* ${\varphi_k}^{-1}(v) = \overline{\text{push}}_{i,a}({\varphi_k}^{-1}(u))$.

Fig. 4 gives an illustration of the Lemma 75.

4.3. Logic on a free group

We now use transfer theorems proved in Section 3.2 to study MSO-properties of the structure $\boldsymbol{\mathcal{P}_k} = \langle \mathcal{P}_k, \varepsilon, (\bullet_a)_{a \in \widehat{A_{1,k}}} \rangle$ where $\bullet_a$ is the binary relation *right-product* by a inside the free group $(\text{Irr}(A_{1,k}), \bullet, \varepsilon)$. We show that for every $k \geq 1$, the structure $\boldsymbol{\mathcal{P}_k}$ has a decidable MSO-theory and fulfills the selection property (Theorem 80). We also define a class of automata with p-oracles recognizing exactly sets which are MSO-definable inside $\boldsymbol{\mathcal{P}_k}$ (Theorem 83).

Let $A_1, \ldots, A_k, \ldots$ be disjoint alphabets fixed for the rest of the paper and Sig_k the signature $(\varepsilon, (\bullet_a)_{a \in \widehat{A_{1,k}}})$ where ε and $\bullet_a$ are respectively unary and binary relations. The signature Sig_k^m is Sig_k augmented with m unary relations. Consider the structure $\boldsymbol{\mathcal{P}_k}$ defined on Sig_k whose domain is $\mathcal{P}_k(A_{1,k})$ and such that $\forall a \in \widehat{A_{1,k}}$, $\bullet_a = \{(u, v) |\ u, v \in \mathcal{P}_k, v = u \bullet a\}$. For every $\vec{O} = (O_1, \ldots, O_m)$ with $O_i \subseteq \mathcal{P}_k$, $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ denotes the structure $\boldsymbol{\mathcal{P}_k}$ augmented with relations $O_1, \ldots, O_m$.

In the sequel, we denote by T_k the tree structure $\langle \mathcal{P}_k, \varepsilon, (\text{succ}_a)_{a \in \widehat{A_{1,k-1}} \cup A_k} \rangle$. In addition, for all vector $\vec{O}$ of subsets of $\mathcal{P}_k$, we write $T_k^{\vec{O}}$ the structure obtained by adding to T_k the unary relations O_i.

Observation 76. For all $k \geq 1$, for all vector $\vec{O}$ of subsets of $\mathcal{P}_{k+1}$,

1. for all vector $\vec{O}$ of subsets of $\mathcal{P}_{k+1}$, $\text{f}_k(T_{k+1}^{\vec{O}}) = \boldsymbol{\mathcal{P}_k}^{\text{f}_k(\vec{O})}$,
2. $\boldsymbol{\mathcal{P}_k}$ is MSO-interpretable in T_k. Indeed, for all $u, v \in \mathcal{P}_k$, $a \in \widehat{A_{1,k}}$:
   - $\boldsymbol{\mathcal{P}_k} \models \varepsilon(u)$ iff $T_k \models \varepsilon(u)$,
   - $\boldsymbol{\mathcal{P}_k} \models \bullet_a(u, v)$ iff $T_k \models \text{succ}_a(u, v) \vee \text{succ}_{\bar{a}}(v, u)$.

Using this fact, we apply transfer theorems of Section 3, to show inductively that $\boldsymbol{\mathcal{P}_k}$ satisfies the selection property, that its MSO-theory is decidable and we give an automata-characterization of the MSO-definable sets of $\boldsymbol{\mathcal{P}_k}$.

First, we need to show that f_k, seen as the restriction to $\mathcal{P}_{k+1}$ of the morphism $\rho \circ \pi_{\widehat{A_{1,k}}} : (\widehat{A_{1,k+1}}^*, \cdot) \to (\text{Irr}(A_{1,k}), \bullet)$ is MSO-invertible. This result will be helpful in two ways: first to apply transfer theorems to $\boldsymbol{\mathcal{P}_k}$ and latter to show that structures $\boldsymbol{\mathcal{P}_k}$ and PDS_k are MSO-equivalent.

Proposition 77. *For every $k \geq 1$, f_k is MSO-invertible.*

In addition, for every $m \geq 0$, for every formula $\phi(\vec{X}) \in \text{MSO}(Sig_k^m)$, one can construct a formula $\phi^{+1}(\vec{X}) \in \text{MSO}(Sig_{k+1}^m)$ such that for all vector $\vec{R}$ of m subsets of $\mathcal{P}_k$, for all vector $\vec{S}$ of subsets of $\mathcal{P}_{k+1}$,

$$\boldsymbol{\mathcal{P}_{k+1}}^{\text{f}_k^{-1}(\vec{R})} \models \phi^{+1}(\vec{S}) \quad \textit{iff} \quad \exists \vec{D} \textit{ such that } \boldsymbol{\mathcal{P}_k}^{(\vec{R})} \models \phi(\vec{D}) \textit{ and } \vec{S} = \text{f}_k^{-1}(\vec{D}).$$

Proof. We follow the technique described in Section 3.3, by using Observation 76(2) we work with $\boldsymbol{\mathcal{P}_{k+1}}$ rather than T_{k+1}.

1. Consider the covering $\mathcal{C}_{k+1}$ of $\mathcal{P}_{k+1}$ consisting of all sets C such that:
   - either $C = \mathcal{P}_k$,
   - or $\exists u \in \mathcal{P}_{k+1}$ and $\exists a \in A_{k+1}$ such that $C = \{uaw \in \mathcal{P}_{k+1} |\ w \in \text{Irr}(A_{1,k})\}$.
     For every $C \in \mathcal{C}_{k+1}$, the restriction of f_k to C is a bijection onto $\mathcal{P}_k$.

   In addition, the property "be an element of $\mathcal{C}$" can easily be expressed by an MSO-formula over $\boldsymbol{\mathcal{P}_{k+1}}$ since $C \in \mathcal{C}_{k+1}$ iff C is a maximal set such that for every $u, v \in C$, there exists a path from u to v using only edges $\bullet_a$ where $a \in \widehat{A_{1,k}}$.

2. For each atomic formula ϕ, we can find a formula ϕ' fulfilling the equivalence (1) of Section 3.3. (we replace relations with free first order variables by "equivalent" relations with free second order variables.
   - $(\varepsilon)'(X, Y) := \exists x \mid X = \{x\} \wedge \bigwedge_{a\in A_{1,k}} \neg(\exists y,\ x \bullet \bar{a} = y)$,
   - $\forall a \in \widehat{A_{1,k}}$, $(\bullet_a(X_1, X_2, Y))' := \bullet_a(X_1, X_2)$.

Let us verify the formula for $C = \{ubw \in \mathcal{P}_{k+1} \mid w \in \mathrm{Irr}(A_k)\}$, with $b \in A_{k+1}$: for every $v = ubw$, $v' = ubw' \in C$,

$$\begin{aligned}
\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k^{-1}(\vec{R})} \models (\bullet_a(S_1 \cap C, S_2 \cap C, C))' \quad \text{iff} \quad & \boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k^{-1}(\vec{R})} \models S_1 \cap C = \{ubw\} \wedge S_1 \cap C = \{ubw'\} \wedge \bullet_a(\{ubw\}, \{ubw'\}) \\
\text{iff} \quad & S_1 \cap C = \{ubw\} \wedge S_1 \cap C = \{ubw'\} \wedge w' = w \bullet a \\
\text{iff} \quad & S_1 \cap C = \{ubw\} \wedge S_1 \cap C = \{ubw'\} \wedge \mathrm{f}_k(ub) \bullet w' = \mathrm{f}_k(ub) \bullet w \bullet a \\
\text{iff} \quad & \boldsymbol{\mathcal{P}_k}^{\vec{R}} \models \bullet_a(\mathrm{f}_k(S_1 \cap C), \mathrm{f}_k(S_2 \cap C)). \quad \square
\end{aligned}$$

Now, to apply transfer theorems, the transfer hypothesis (TH) (see Definition 52) is required: $\mathcal{P}_{k+1}$ must belong to $\mathsf{REG}^{\vec{O}}(A)$, for $\vec{O} = \mathrm{f}_k^{-1}(\vec{R})$ and $\vec{R}$ vector of subsets of $\mu(P)$. Let us define such a vector $\vec{O}$ by induction over k: for every $k \geq 1$, the vector $\vec{O}_k$ of subsets of $\mathcal{P}_k$ is defined by the following induction:

- $\vec{O}_1 = \emptyset$
- $\vec{O}_{k+1} = (\mathrm{f}_k^{-1}(\vec{O}_k), \mathrm{f}_k^{-1}(\mathcal{P}_k a_1), \ldots, \mathrm{f}_k^{-1}(\mathcal{P}_k a_n))$ where $A_{1,k} = \{a_1, \ldots, a_n\}$

In other words, $\vec{O}_k$ consists in every $\mathrm{f}_{k,i}^{-1}(\mathcal{P}_i a)$, where $1 \leq i \leq k$ and $a \in A_{1,i}$.

The following result is required to apply transfer theorems (see Definition 52).

Lemma 78. *For every $k \geq 1$, the property* $\mathrm{TH}(\mathrm{f}_k, \vec{O_{k+1}})$ *is satisfied.*

Proof. It is enough to check that for every $k \geq 0$, $\mathcal{P}_{k+1} \in \mathsf{REG}^{\vec{O}_{k+1}}$. From Proposition 70, whenever $u \in \mathcal{P}_{k+1}$ and $a \in A_{1,k+1}$, then $ua \in \mathcal{P}_{k+1}$ except if u ends by $\bar{a}$, hence it suffices to code the last letter read in the state. In addition, if $a \in A_{1,k}$, then $u\bar{a} \in \mathcal{P}_{k+1}$ iff $u \neq u'a$ and $u \in \mathrm{f}_{k,i}^{-1}(\mathcal{P}_i a)$, this last condition can be tested with the oracles. $\square$

Now we show that relations $\vec{O}_k$ do not add expressivity to the logics we consider.

Lemma 79. *For every $k \geq 1$, structures $\boldsymbol{\mathcal{P}_k}$, T_k, $T_k^{\vec{O}_k}$ and $\mathrm{f}_k(T_{k+1}^{\vec{O}_{k+1}})$ are* MSO*-equivalent.*

Proof.

1. From Observation 76, $\boldsymbol{\mathcal{P}_k}$ is MSO-interpretable in T_k
2. Obviously, T_k is MSO-interpretable in $T_k^{\vec{O}_k}$
3. $T_k^{\vec{O}_k}$ is MSO-interpretable in $\boldsymbol{\mathcal{P}_k}$: each $\mathcal{P}_k \cdot a \in \widehat{A_{1,k}}$ is MSO-definable inside $\boldsymbol{\mathcal{P}_k}$. Indeed, $\mathcal{P}_k \cdot a$ is the smallest set $S \subseteq \mathcal{P}_k$ such that for every $u \in \mathcal{P}_k$, $u \in S$ iff either $\bullet_a(u, \varepsilon)$, or there exists $b \neq \bar{a} \in \widehat{A_{1,k}}$ and $v \in \mathcal{P}_k \cdot b$ such that $u = v \bullet a$.
   - $T_k^{\vec{O}_k} \models \mathrm{succ}_a(u, v)$ iff $\boldsymbol{\mathcal{P}_k} \models \bullet_a(u, v) \wedge v \notin \mathcal{P}_k \cdot \bar{a}$
   - From Proposition 77, each element $\mathrm{f}_{k,i}^{-1}(\mathcal{P}_i \cdot a)$ of $\vec{O}_k$ is definable in $\boldsymbol{\mathcal{P}_k}$, since $\mathcal{P}_i \cdot a$ is definable in $\boldsymbol{\mathcal{P}_i}$.
4. $\mathrm{f}_k(T_{k+1}^{\vec{O}_{k+1}})$ is MSO-equivalent to the structure $\boldsymbol{\mathcal{P}_k}^{\vec{O}_k}$, which is MSO-equivalent to the structure $T_k^{\vec{O}_k}$, which is MSO-equivalent to the structure $\boldsymbol{\mathcal{P}_k}$. $\square$

Theorem 80. *For every $k \geq 1$, the structure $\boldsymbol{\mathcal{P}_k}$ has a decidable* MSO*-theory and fulfills the selection property.*

Proof. We prove this result by induction on $k \geq 1$:

Basis: From Theorem 34, $\boldsymbol{\mathcal{P}_1}$ has a decidable MSO-theory and satisfies the selection property.

Induction step: let us suppose the property true for $k \geq 1$. Since $\mathrm{TH}(\mathrm{f}_k, \vec{O_{k+1}})$, by using Theorem 53 and equivalence between structures proved Proposition 79, the MSO-theory of $\boldsymbol{\mathcal{P}_{k+1}}$ is decidable. In the same way, since from Proposition 77, the map f_k is MSO-invertible, and by using Theorem 55, $\boldsymbol{\mathcal{P}_{k+1}}$ satisfies the selection property. $\square$

Remark that the decidability result has already been proved in [17].

The same kind of reasoning can be applied to the structure $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ for any vector $\vec{O}$ of subsets of $\mathcal{P}_k$: if the MSO-theory of $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ is decidable, then the MSO-theory of $\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k^{-1}(\vec{O})}$ is decidable.

Theorem 81. *Given $\vec{R}$ a vector of subsets of A_1^*, and $\vec{O} = \mathrm{f}_{k,1}^{-1}(\vec{R})$,*

1. *if the* MSO*-theory of $\langle A_1^*, \varepsilon, (\mathrm{succ}_a)_{a\in A_1}, \vec{R}\rangle$ is decidable, then for every $k \geq 1$, the* MSO*-theory of $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ is decidable,*
2. *if $\langle A_1^*, \varepsilon, (\mathrm{succ}_a)_{a\in A_1}, \vec{R}\rangle$ fulfills* SP*, then $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ fulfills* SP*.*

We define now the class FA_k of automata which will be use to recognize languages in $\mathcal{P}_k$.

Definition 82. For all $k \geq 1$, classes FA_k and REG_k are defined inductively as follows:

- FA_1 is the class of finite automata, and REG_1 the regular languages one,
- for every $k \geq 1$, FA_{k+1} consists in all automata $\mathcal{A}$ with p-oracle $(\mathrm{f}_k^{-1}(R_1), \ldots, \mathrm{f}_k^{-1}(R_m))$ such that each R_i belongs to REG_k,
- for every $k \geq 1$, REG_{k+1} consists in all languages in $\mathcal{P}_k$ recognized by automata in FA_{k+1}.

Theorem 83. *For every language $L \subseteq \mathcal{P}_k$ with $k \geq 1$, L is* MSO*-definable in $\boldsymbol{\mathcal{P}_k}$ iff L belongs to REG_k.*

Proof. Let us prove this result by induction on $k \geq 1$.

Basis: the case $k = 1$ is Rabin's theorem,

Induction step: let us suppose the property holds for $k \geq 1$.

From Theorem 56, any language $L \subseteq \mathcal{P}_{k+1}$ is MSO-definable in $\boldsymbol{\mathcal{P}_{k+1}}$ iff there exists a vector $\vec{D}$ MSO-definable in $\boldsymbol{\mathcal{P}_k}$ and $\mathcal{A} \in \mathsf{FA}^{\vec{D}}$ such that $L = \mathrm{L}(\mathcal{A})$. By induction hypothesis, each component of $\vec{D}$ belongs to REG_k, and then, L is MSO-definable in $\boldsymbol{\mathcal{P}_{k+1}}$ iff $L \in \mathsf{REG}_{k+1}$. □

4.4. Regular sets of k-pushdowns

We now translate results obtained on $\mathcal{P}_k$ in terms of k-pushdowns. For that, we simply prove that $\boldsymbol{\mathcal{P}_k}$ and the structure PDS_k associated to the type k-pushdowns are MSO-equivalent (see Definition 10). Precisely, we show that $\varphi_k : \mathsf{PDS}_k \to \boldsymbol{\mathcal{P}_k}$ and $\varphi_k{}^{-1} : \boldsymbol{\mathcal{P}_k} \to \mathsf{PDS}_k$ are computable MSO-interpretations. Then, the two structures have the same MSO-properties and we have a class of p-oracle-automata available to characterize the class of all $\varphi_k(D)$ such that D is MSO-definable in PDS_k. Using this automata characterization, we define the class of controlled k-pushdown transition systems o generating the class of all sets MSO-definable in PDS_k.

Proposition 84. *For every $k \geq 1$, $\varphi_k{}^{-1} : \boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k) \to \mathsf{PDS}_k(A_1, \ldots, A_k)$ is an* MSO*-interpretation.*

Proof. Let us check that conditions of Definition 8 are well satisfied,

1. $\varphi_k{}^{-1}(\mathcal{P}_k) = k$-pds is MSO-definable in PDS_k
2. from the Lemma 75, it follows that for every $u, v \in \mathcal{P}_k$, $a \in A_i$, $1 \leq i \leq k$:
   $\boldsymbol{\mathcal{P}_k} \models \bullet_a(u, v)$ iff $\mathsf{PDS}_k \models \mathrm{push}_a(\varphi_k{}^{-1}(u), \varphi_k{}^{-1}(v))$ and
   $\boldsymbol{\mathcal{P}_k} \models \bullet_{\bar{a}}(u, v)$ iff $\mathsf{PDS}_k \models \mathrm{push}_a(\varphi_k{}^{-1}(v), \varphi_k{}^{-1}(u))$. □

Conversely, let us prove that $\forall k \geq 1$, $\varphi_k : \mathsf{PDS}_k \to \boldsymbol{\mathcal{P}_k}$ is an MSO-interpretation. The next lemma establishes that to show that a given instruction of level k is MSO-definable in $\boldsymbol{\mathcal{P}_{k+i}}$, $i \geq 0$, one only to need to demonstrate it is MSO-definable in $\boldsymbol{\mathcal{P}_k}$:

Lemma 85. *Given instr an instruction of level $k \geq 1$, and $\phi(x, y)$ an* MSO*-formula over Sig_k satisfying for all $u, v \in \mathcal{P}_k$:*

$$\boldsymbol{\mathcal{P}_k} \models \phi(u, v) \quad \textit{iff} \quad \varphi_k{}^{-1}(v) = \textit{instr}(\varphi_k{}^{-1}(u)),$$

then for every $i \geq 0$, there exists a formula $\phi_{+i}(x, y) \in \mathrm{MSO}(\mathrm{Sig}_{k+i})$ such that $\forall u, v \in \mathcal{P}_{k+i}$,

$$\boldsymbol{\mathcal{P}_{k+i}} \models \phi_{+i}(u, v) \quad \textit{iff} \quad \varphi_{k+i}{}^{-1}(v) = \textit{instr}(\varphi_{k+i}{}^{-1}(u)).$$

Proof. From Proposition 73 and definition of $\varphi_k{}^{-1}$, it follows that for every $v, v' \in (k+1)$-pds and $\omega_1, \omega_1' \in k$-pds, the following properties are equivalent:

1. there exists $\omega \in (k+1)\text{-pds} \cup \{\varepsilon\}$ and $a \in A_{k+1} \cup \{\bot\}$ such that
   $$\varphi_{k+1}{}^{-1}(v) = a[\omega_1]\omega \quad \text{and} \quad \varphi_{k+1}{}^{-1}(v') = a[\omega_1']\omega$$
2. $f_k(v) = \varphi_k(\omega_1)$ and $f_k(v') = \varphi_k(\omega_1')$ and there exists $u \in \mathrm{Irr}(A_k)$ such that $v' = v \bullet u$.

Then, given instr an instruction of level k and ϕ MSO-defining instr in $\boldsymbol{\mathcal{P}_k}$, we obtain (by using formulas constructed in Proposition 77) the following iterative construction of ϕ_{+i}: $\phi_{+0}(x, y) := \phi(x, y)$ and $\forall i \geq 0$, $\phi_{+(i+1)}(x, y) := \exists u, y = x \bullet u \wedge (\phi_{+i})^{+1}(x, y)$. □

Proposition 86. *For every $k \geq 1$, $\varphi_k : \mathsf{PDS}_k \to \boldsymbol{\mathcal{P}_k}$ is a* MSO*-interpretation.*

Proof. Using Lemma 85, it only remains to show that $\mathrm{push}_{k,a}$, pop_k and $\mathrm{change}_{k,a}$ are MSO-definable in $\boldsymbol{\mathcal{P}_k}$: for every $a \in A_i$, $1 \leq i \leq k$,

$\textsc{push}_{k,a}(x, y) := y = x \bullet a$,
$\textsc{pop}_{k,a}(x, y) := \exists a \in A_k, \exists w, x = y \bullet a \bullet w$,
$\textsc{change}_{k,a}(x, y) := \exists u, u', \exists b \in A_k, y = x \bullet u' \bullet \bar{b} \bullet a \bullet u \wedge x\,(=)^{+1}\,y$. □

Corollary 87. *For every $k \geq 1$, every $D \subseteq k$-pds, D is* MSO*-definable in* PDS_k *iff $\varphi_k(D)$ is* MSO*-definable in $\mathcal{P}_k$.*

We now translate the properties of k-regular languages in terms of k-pushdowns by using the MSO-equivalence of the structures $\mathcal{P}_k$ and PDS_k.
The following theorem is straightforward from Theorem 80, Propositions 84 and 86.

Theorem 88. *For every $k \geq 1$, the structure* PDS_k *has a decidable* MSO*-theory and fulfills the selection property.*

The decidability result is proved in [15] by using Muchnik's Theorem on tree-like structures (see [40] or [5]). Finally, we show that the class REG_k (Definition 82) admits several characterizations that extend the REG ones.

Theorem 89. *For every $S \subseteq k$-$\mathsf{pds}(A_1, \ldots, A_k)$, $k \geq 1$, the following properties are equivalent:*

1. *S is generated by a k-pds system of transitions whose controllers are* MSO*-definable in* $\mathsf{PDS}_k(A_1, \ldots, A_k)$
2. *S is* MSO*-definable in* $\mathsf{PDS}_k(A_1, \ldots, A_k)$
3. *$\varphi_k(S)$ is* MSO*-definable in $\mathcal{P}_k(A_1, \ldots, A_k)$*
4. *$\varphi_k(S)$ is recognized by an automaton in* $\mathsf{FA}_k(A_1, \ldots, A_k)$.

Proof. Equivalence between 2 and 3 stems from the equivalence between the two structures. Equivalence between 3 and 4 is established in Theorem 83.
Given a k-pds system of transitions $\mathcal{S}$ controlled by a vector $\vec{C}$ of sets which are MSO-definable in PDS_k. It is possible to write a formula defining in PDS_k the set of k-pds generated by $\mathcal{S}$. So 1 implies 2.
Let us end the proof by showing that 4 implies 1. Given $k \geq 0$ and $\mathcal{A} = (Q, \widehat{A_{1,k}}, \vec{R}, \Delta, q_0, F) \in \mathsf{FA}_k$, we construct $\mathfrak{A} \in k$-$\mathsf{TS}^{\vec{C}}$ with $\vec{C} = {\varphi_k}^{-1}(\vec{R})$ fulfilling $\varphi_k(\mathrm{L}(\mathcal{A})) = \mathrm{P}(\mathfrak{A})$, as follows.
From the equivalence between 2 and 3, $\vec{C}$ is a vector of sets which are MSO-definable in PDS_k. We can suppose w.l.o.g. that $\mathcal{A}$ is complete in $\mathcal{P}_k$, i.e., that $\forall u \in \widehat{A_{1,k}}^*$, u is computed by $\mathcal{A}$ iff $u \in \mathcal{P}_k$.
Let us set $\mathfrak{A} = (Q, (A_1, \ldots, A_k), \Delta', \vec{C}, q_0, \bot, F)$ where Δ' is constructed in the following way:

- $\forall (p, a, \vec{o}, q) \in \Delta, a \in A_i, 1 \leq i \leq k+1$, then $\forall w \in \mathrm{top}(k\text{-}\mathsf{pds}(A_1, \ldots, A_k))$
  $(p, w, \vec{o}, \mathrm{push}_{i,a}, q) \in \Delta'$
- $\forall (p, \bar{a}, \vec{o}, q) \in \Delta, a \in A_i, 1 \leq i \leq k$, then $\forall w \in \mathrm{top}(k\text{-}\mathsf{pds}(A_1, \ldots, A_k))$
  $(p, w, \vec{o}, \mathrm{pop}_i, q) \in \Delta'$.

It can be easily checked that $\varphi_k(\mathrm{L}(\mathcal{A})) = \mathrm{P}(\mathfrak{A})$. □

Remark 90.

1. It can be proved that languages recognized by k-pds automata controlled by MSO-definable sets are languages recognized by k-pds automata without controllers.
2. The equivalence between (1) and (4) is proved in [14] for $k = 1$.

5. Final comment

The work presented here is a part of the author's Ph.D. presented in 2005 at Bordeaux University on the theme of Iterated Pushdown automata [39]. It was shown that there are several applications of Theorem 89. In particular, using Theorem 89, one can define a large class of tuples $(P_1, \ldots, P_m)$ of unary predicates for which the MSO-theory of $\langle \mathbb{N}, +1, P_1, \ldots, P_m \rangle$ is decidable.

Recent other work deals with the notion of regular sets of "higher-order pushdowns" (hop) which are restricted *it*-pushdowns. In [41], a set S of k-hops is called *regular* if the set of words in $(\mathcal{A}_k \cup \{[,]\})^*$ representing S is accepted by a finite automaton. It is shown that for any higher-order process with a single state, the set of all predecessors of a given *regular* set of configurations is *regular*.

In [38], Carayol introduces a notion of regular sets of higher-order pushdowns. He studies the class Reg_k corresponding to the sets of k-hop accessible by using only instruction push and $\overline{\text{push}}$. He gives a normalized representation of this class using regular expressions over a monoid in $(\widehat{A_{1,k}} \cup T_k)^*$, where T_k is an infinite alphabet consisting of all symbols T_R, for $R \in Reg_{k-1}$. This normalization extended the one obtained in [42], an later in [43], for the level 1. The author proves also that the class Reg_k corresponds to the class of sets MSO-definable in PDS_k. The class Reg_k corresponds then to the image by φ_k of the class REG^k defined in Section 4. These two independent works prove that the class REGk possesses numerous properties generalizing the REG ones, and that the representation of k-pds by the words in the free group seems very suited to define regular sets of stacks.

All these results have already been used in several papers dealing with higher order pushdown automata and games [44–47]. Note also that, in [48], authors extends oracles automata defined here, to infinite words by defining *Büchi automaton with advice*.

Acknowledgements

The author would thank G. Sénizergues for having directed this work, F. Carrère and C. Morvan for their helpful corrections, and the reviewers for their thoughtful comments and suggestions, and for their patience.

References

[1] O. Matz, N. Schweikardt, Expressive power of monadic logics on words, trees, pictures, and graphs, in: J. Flum, E. Grädel, T. Wilke (Eds.), Logic and Automata, in: Texts in Logic and Games, vol. 2, Amsterdam University Press, 2008, pp. 531–552.
[2] W. Thomas, Languages, automata, and logic, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 3, Springer, New York, 1997, pp. 389–455.
[3] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, Trans. Amer. Math. Soc. 141 (1969) 1–35.
[4] A.L. Semenov, Decidability of monadic theories, in: Mathematical foundations of computer science, 1984 (Prague, 1984), in: Lecture Notes in Comput. Sci., vol. 176, Springer, Berlin, 1984, pp. 162–175.
[5] I. Walukiewicz, Monadic second-order logic on tree-like structures, Theoret. Comput. Sci. 275 (2002) 311–346.
[6] B. Courcelle, Fundamental properties of infinite trees, Theoret. Comput. Sci. 25 (1983) 95–169.
[7] J. Büchi, L. Landweber, Solving sequential conditions finite-state strategies, Trans. Amer. Math. Soc. 138 (1969) 295–311.
[8] S. Lifsches, S. Shelah, Uniformization, choice functions and well orders in the class of trees, J. Symbolic Logic 61 (1996) 1206–1227.
[9] S. Lifsches, S. Shelah, Uniformization and Skolem functions in the class of trees, J. Symbolic Logic 63 (1998) 103–127.
[10] A. Rabinovich, On decidability of monadic logic of order over the naturals extended by monadic predicates, Inf. Comput. 205 (2007) 870–889.
[11] A. Rabinovich, A. Shomrat, Selection over classes of ordinals expanded by monadic predicates, Ann. Pure Appl. Logic 161 (2010) 1006–1023.
[12] A. Rabinovich, A. Shomrat, Selection and uniformization problems in the monadic theory of ordinals: a survey, in: A. Avron, N. Dershowitz, A. Rabinovich (Eds.), Pillars of Computer Science, in: Lecture Notes in Computer Science, vol. 4800, Springer, 2008, pp. 571–588.
[13] A. Rabinovich, A. Shomrat, Selection in the monadic theory of a countable ordinal, J. Symbolic Logic 73 (2008) 783–816.
[14] S.A. Greibach, A note on pushdown store automata and regular systems, Proc. Amer. Math. Soc. 18 (1967) 263–268.
[15] S. Fratani, G. Sénizergues, Iterated pushdown automata and sequences of rational numbers, in: Ann. Pure Appl. Logic, vol. 141, Elsevier, 2005, pp. 363–411.
[16] T. Cachat, Higher order pushdown automata, the Caucal hierarchy of graphs and parity games, in: Automata, Languages and Programming, in: Lecture Notes in Comput. Sci., vol. 2719, Springer, Berlin, 2003, pp. 556–569.
[17] A. Carayol, S. Wöhrle, The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata, in: FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, in: Lecture Notes in Comput. Sci., vol. 2914, Springer, Berlin, 2003, pp. 112–123.
[18] T. Knapik, D. Niwiński, P. Urzyczyn, Higher-order pushdown trees are easy, in: Foundations of Software Science and Computation Structures (Grenoble, 2002), in: Lecture Notes in Comput. Sci., vol. 2303, Springer, Berlin, 2002, pp. 205–222.
[19] M. Hague, C.-H.L. Ong, Symbolic backwards-reachability analysis for higher-order pushdown systems, in: Proceedings of the 10th International Conference on Foundations of Software Science and Computational Structures, FOSSACS'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 213–227.
[20] I. Schiering, W. Thomas, Counter-free automata, first-order logic, and star-free expressions extended by prefix oracles, in: Developments in Language Theory, II (Magdeburg, 1995), Worl Sci. Publishing, River Edge, NJ, 1996, pp. 166–175.
[21] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, in: Addison-Wesley Series in Computer Science, Addison-Wesley Publishing Co, Reading, Mass., 1979.
[22] M. Rabin, Decidable theories, in: Handbook of Mathematical Logic, 1977, pp. 595–629.
[23] W. Thomas, Automata on infinite objects, in: Handbook of Theoretical Computer Science, vol. B, Elsevier, Amsterdam, 1990, pp. 133–191.
[24] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current Research [Outcome of a Dagstuhl Seminar, February 2001], in: Lecture Notes in Comput. Sci., vol. 2500, Springer, 2002.
[25] E.A. Emerson, C.S. Jutla, Tree automata, mu-calculus and determinacy, in: Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, USA, 1991, pp. 368–377.
[26] Y. Gurevich, L. Harrington, Trees, automata, and games, in: Proceedings of the 14th Symposium on Theory of Computing, Association for Computing Machinery, pp. 60–65.
[27] O. Carton, W. Thomas, The monadic theory of morphic infinite words and generalizations, Inform. Comput. 176 (2002) 51–65.
[28] W. Thomas, Constructing infinite graphs with a decidable MSO-theory, in: Mathematical Foundations of Computer Science 2003, in: Lecture Notes in Comput. Sci., vol. 2747, Springer, Berlin, 2003, pp. 113–124.
[29] B. Courcelle, The monadic second-order logic of graphs. IX. Machines and their behaviours, Theoret. Comput. Sci. 151 (1995) 125–162. Topology and completion in semantics (Chartres, 1993).
[30] B. Courcelle, I. Walukiewicz, Monadic second-order logic, graph coverings and unfoldings of transition systems, Ann. Pure Appl. Logic 92 (1998) 35–62.
[31] S.A. Greibach, Full AFLs and nested iterated substitution, Inf. Control 16 (1970) 7–35.
[32] A.V. Aho, Nested stack automata, J. Assoc. Comput. Mach. 16 (1969) 383–406.
[33] A.N. Maslov, The hierarchy of index languages of arbitrary level, Dokl. Akad. Nauk SSSR 217 (1974) 1013–1016.
[34] A.N. Maslov, Multilevel pushdown automata, Probl. Peredači Inf. 12 (1976) 55–62.
[35] W. Damm, A. Goerdt, An automata-theoretical characterization of the OI-hierarchy, Inform. Control 71 (1986) 1–32.
[36] J. Engelfriet, Iterated pushdown automata and complexity classes, in: Proceedings of the 14th Symposium on Theory of Computing, Association for Computing Machinery, pp. 365–373.
[37] J. Engelfriet, Iterated stack automata and complexity classes, Inform. Comput. 95 (1991) 21–75.
[38] A. Carayol, Regular sets of higher-order pushdown stacks, in: MFCS, in: Lecture Notes in Comput. Sci., vol. 3618, Springer, 2005, pp. 168–179.
[39] S. Fratani, Automates à piles de piles... de piles, Ph.D. Thesis, Université Bordeaux 1, 2005.
[40] A.L. Semenov, Decidability of monadic theories, in: Mathematical Foundations of Computer Science, 1984 (Prague, 1984), Springer, Berlin, 1984, pp. 162–175.
[41] A. Bouajjani, A. Meyer, Symbolic reachability analysis of higher-order context-free processes, in: FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, in: Lecture Notes in Comput. Sci., vol. 3328, Springer, Berlin, 2004, pp. 135–147.
[42] D. Caucal, On infinite transition graphs having a decidable monadic theory, in: Automata, Languages and Programming (Paderborn, 1996), in: Lecture Notes in Comput. Sci., vol. 1099, Springer, Berlin, 1996, pp. 194–205.
[43] D. Caucal, On infinite transition graphs having a decidable monadic theory, Theoret. Comput. Sci. 290 (2003) 79–115.
[44] A. Carayol, M. Hague, A. Meyer, C.-H.L. Ong, O. Serre, Winning regions of higher-order pushdown games, in: Logic in Computer Science, Symposium on 0, 2008, pp. 193–204.
[45] A. Carayol, M. Slaats, Positional strategies for higher-order pushdown parity games, in: Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science, MFCS '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 217–228.
[46] L. Braud, A. Carayol, Linear orders in the pushdown hierarchy, in: Proceedings of the 37th International Colloquium Conference on Automata, Languages and Programming: Part II, ICALP'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 88–99.
[47] P. Hänsch, M. Slaats, W. Thomas, Parametrized regular infinite games and higher-order pushdown strategies, in: Proceedings of the 17th International Conference on Fundamentals of Computation Theory, FCT'09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 181–192.
[48] M. Bojańczyk, D. Niwiński, A. Rabinovich, A. Radziwończyk-Syta, M. Skrzypczak, On the Borel complexity of MSO definable sets of branches, Fundam. Inf. 98 (2010) 337–349.