Walking automata in the free inverse monoid

David Janin

LaBRI, CNRS UMR 5800
Laboratoire Bordelais de Recherche en Informatique

Rapport de recherche RR-1464-12

Walking automata in the free inverse monoid

October 5, 2012

David Janin,
LaBRI, IPB, Université de Bordeaux

Abstract

This paper considers subsets of the free inverse monoid or, following Munn's representation theorem, languages of finite birooted trees.

We propose three notions of definability for these languages: definability by means of finite state walking automata with nested (invisible) pebbles, definability by means of (extension of) Kleene regular expressions, and definability by means of (adequate) premorphisms (or relational morphisms) in (adequate) finite ordered monoids.

Various correspondences, linking these three notions together, are shown to hold. Finite walking automata with a finite number of pebbles are shown to be captured by regular expressions; the number of allowed pebbles corresponds to the nesting depth of a projection operator onto languages of idempotent birooted trees. Finite walking automata with a finite or an infinite number of pebbles are also shown to be captured by finitely generated premorphisms from the (naturally ordered) free inverse monoid in their associated (finite) transition monoids (ordered by inclusion).

These results strengthen the idea that (some subcategory of) the category of ordered monoids and premorphisms is an adequate framework for the study of walking automata on trees much in the same way the category of monoids and morphisms is an adequate framework for the study of classic (one way) automata.

Moreover, since our algebraic characterization of walking automata holds even for automata using infinitely many pebbles, it also provides a presumably new algebraic framework for the study of regular languages of finite trees.

Contents

1 Introduction

Background

Whether they walk on words or trees, finite state *walking automata* have been studied for many years in theoretical computer science.

Presumably simple in nature - they are simply standard word automata extended with the capacity to read letters both ways - it however took more than thirty years to prove that, on finite trees, they are strictly less expressive than finite state (branching) tree automata [3]. Worse still, when extended with the capacity to drop and lift a finite number of pebbles (in a nested fashion) [7], walking automata still fail to capture all the regular languages of trees [4]. Indeed, one needs to allow an unbounded number of pebbles [6] to reach the classic yardstick of expressive power defined by regular languages of trees or, equivalently, languages definable by means of formulae of monadic second order (MSO) logic.

At first sight, these results reveal that walking automata are barely understandable and inadequately expressive. Must the story end with such negative results?

As observed in [7], walking automata are models of *sequential machines* that extend classic finite state automata. With a bounded number of pebbles they behave like (restricted) *hierarchical finite state systems*. With an unbounded number of pebbles they behave like (restricted) *recursive finite state systems*. And, because of these restrictions, the language emptyness problem for walking automata is still decidable.

In other words, walking automata provide a fairly general decidable model of sequential machines, which is intrinsically different from the model of *parallel* machines that is induced by (branching) tree automata. From the standpoint of computation models, achieving a deeper understanding of the behavior of walking automata is thus highly relevant.

Since walking automata are sequential machines, one may expect to extend to walking automata (some elements of) the enormous amount of mathematical tools that have been successfully developed for the study and (fine grained) analysis of finite state (one way) automata on words (see [14] for an overview). Of course, in the simpler case of two-way automata, one can built finite monoids that characterize their semantics. This however amounts to converting two-way automata into one-way automata. As observed by Birget [1] years ago and still true nowadays, there are as yet no appropriate algebraic settings that capture the behavior of two-way automata. With the remarkable exception of deterministic machines [11], adapting algebraic approaches presumably fails when applied to walking automata.

As observed in [4], this can be explained by the fact that walking automata runs are *not presumably compositional*: the set of runs of a walking automata on a given tree is not presumably a function of the set of subruns induced by decomposition of the same tree into substructures. Still, coping with this lack of compositionality is a key issue. For instance, in their proof of the infiniteness of the pebble hierarchy [4], the authors eventually achieve some (unfortunately technical) compositional description of walking automata behavior.

Yet how close to success are naive applications of classic techniques? Are these approaches bound to fail?

In this paper, we show that although simple compositionality fails, given a walking automaton, input structures can still be mapped to sets of runs (i.e. pairs of states) in such a way that the resulting mapping still enjoys many interesting algebraic properties. In particular, this mapping turns out to be *finitely generated*, and this property is one of the corner stones of algebraic approaches.

How do we achieve such a result? We propose to study walking automata semantics through an embedding of (languages of) trees into (languages of) birooted trees.

Birooted trees came to the forefront in the 70s with Munn's theorem[13]. They provide a concrete description of elements of the free inverse monoid.

For computer scientists, the free inverse monoid can be seen as the *quotient* of the set of *walks* in tree-shaped structures (with labeled directed edges) by the *equivalence* which indicates when two walks, issuing from the same initial vertex, traverse the same (implicit) substructure and end in the same final vertex. Munn's birooted trees can thus be seen as explicit models of these implicit underlying substructures.

Since the above equivalence turns out to be a congruence with respect to the sequential composition of walks, the resulting algebraic structure is a monoid. As every walk can actually be reversed, the monoid in question turns out to be an inverse monoid. Since every walk can be modeled by the sequence of (labeled directed) edges traversed either forwards or backwards, the monoid of birooted trees is even finitely generated by (forward or backward) single edge birooted trees.

The free inverse monoid is thus peculiarly well suited for the study of walking automata. Indeed, partial runs (or walks) of tree walking automata actually define bi-rooted trees.

Moreover, the induced embedding of all (edge labeled deterministic) finite trees (with unlabeled vertices) into the monoid of birooted trees also provides a surprisingly simple algebraic framework for generating those finite trees. This observation also constitutes another prime motivation for studying the languages of birooted

trees since, in the long term, such studies may also impact our understanding of regular tree languages themselves beyond the theory of walking automata.

Outline

In this paper, we thus propose a computer-science flavored study of subsets of the free-inverse monoid which, following Munn's theorem [13], are seen as languages of birooted trees.

This study continues and extends the study initiated in [10] for languages of overlapping tiles that are linear unidirectional birooted trees. It has already been shown [5] that two-way automata on words can relevantly be studied on overlapping tiles. We aim at proving here that tree walking automata semantics can relevantly be studied on birooted trees.

Generalizing the results presented in [5], we first provide a one to one correspondence between finite state (invisible) pebble automata (with complete run semantics) and (extensions of) Kleene regular expressions. Automata with no pebbles are shown to be captured by standard Kleene expressions (with forward or backward letters) and, more generally, the number of allowed pebbles is shown to correspond to the nesting depth (in extended Kleene expressions) of the projection operator of birooted tree languages to the subset of their idempotent elements.

We also demonstrate that recognizability by means of McAlister and Reilly's premorphisms [12] can be used to capture the partial run semantics of walking automata on birooted trees. Premorphisms are monotonic mappings φ between ordered monoids such that $\varphi(xy) \leq \varphi(x) \cdot \varphi(y)$. Their use in theoretical computer science has recently been advocated for languages of tiles [9],

We first show that the (partial run) semantics of every finite 0-pebble (or classic) walking automaton is captured by a *finitely generated* premorphism β^0 from the free-inverse monoid into the (finite) transition monoid $\mathcal{P}(Q \times Q)$ induced by this automaton. As premorphisms are only sub-multiplicative this result thus constitutes a first non-trivial issue.

We then show that the transition monoid $\mathcal{P}(Q \times Q)$ associated with a walking automaton can itself be extended by a self mapping (encoding the recursion mechanism defined by pebbles handling) in such a way that the (partial run) semantics of every finite k-pebble (or unbounded number of pebbles) walking automaton is also captured by a *finitely generated* premorphism β^k (or just β) from the free-inverse monoid into the same transition monoid.

We conclude our study by showing that, even if birooted trees have no vertex labels, such vertex labeling can still be simulate by additional dangling edges. The

main difficulty is to restrict to well-marked birooted trees (where regular edges are distinguished from edges defining vertex labels). But we show that this amounts to taking the Rees's quotient of the free inverse monoid by the ideal of badly-marked birooted trees and all our results still hold restricting to such a kind of quotient of the free inverse monoid.

As a corollary, we prove that every MSO definable language of finite trees is the inverse image of some finite subset of a finite ordered monoid by some (finitely generated) premorphism, i.e. regular languages of trees are quasi-recognizable in the sense of [9].

Preliminaries

Free monoid. Given a finite alphabet A, let A^* be the free monoid generated by A, 1 denoting the neutral element. The concatenation of two words (or more generally two element of a monoid) u and v is denoted by $u \cdot v$ or even just uv.

Monoid of walks. Given $\bar{A}$ a disjoint copy of A, $u \mapsto \bar{u}$ denotes the mapping from $(A + \bar{A})^*$ to itself inductively defined by $\bar{1} = 1$, for every letter $a \in A$, $\bar{a}$ is the copy of a in $\bar{A}$ and $\bar{\bar{a}} = a$ and, for every word $u \in (A + \bar{A})^*$, $\overline{au} = \bar{u} \cdot \bar{a}$. The mapping $u \mapsto \bar{u}$ is an antimorphism, i.e. for all words u and $v \in (A + \bar{A})^*$, $\overline{uv} = \bar{v} \cdot \bar{u}$ and an involution, i.e. for every word $u \in (A + \bar{A})^*$, $\bar{\bar{u}} = u$. In the sequel, elements of the free monoid $(A + \bar{A})^*$ are called *walks*.

Free group. The free group $FG(A)$ generated by A is defined as $(A+\bar{A})^*$ quotiented by the least congruence over $(A + \bar{A})^*$ such that, for every letter $a \in A$, $a\bar{a} \simeq 1$ and $\bar{a}a \simeq 1$. Every equivalence class $[u] \in FG(A)$ contains a unique element $red(u)$ (the *reduce* of u) that contains no factor of the form $a \cdot \bar{a}$ or $\bar{a} \cdot a$. Elements of $FG(A)$ are identified with reduced words, the product of two elements u and $v \in FG(A)$ being defined in $FG(A)$ by $red(u \cdot v)$.

Free inverse monoid. The free inverse monoid $FIM(A)$ generated by A is defined as $(A + \bar{A})^*$ quotiented by the Wagner congruence $\simeq_W$, i.e. the least congruence over $(A + \bar{A})^*$ such that $u\bar{u}u \simeq_W u$ and $u\bar{u}v\bar{v} \simeq_W v\bar{v}u\bar{u}$ for all $u, v \in (A + \bar{A})^*$.

Let us recall that, being an inverse monoid, for every $x \in FIM(A)$, there is a unique element x^{-1}, the *pseudo inverse* of x, such that $xx^{-1}x = x$ and $x^{-1}xx^{-1} = x^{-1}$. As a consequence, idempotents are elements of the form xx^{-1} for $x \in FIM(A)$, they are self inverse and they commute one with the other.

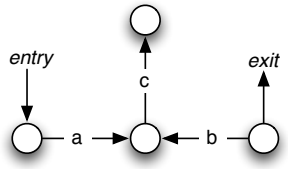
Munn's representation. An alternative definition of $FIM(A)$, due to Scheiblich and Munn [15, 13], is the following. Elements are pairs of the form $(u, P) \in FG(A) \times \mathcal{P}(FG(A))$ where P is a finite prefix-closed subset of $FG(A)$ with $1 \in P$ and $u \in P$.

The product of two pairs (u, P) and (v, Q) is defined by $(u, P) \cdot (v, Q) = (uv, P \cup red(uQ))$. Scheiblich-Munn's Theorem states that the mapping θ that maps every word $w \in (A + \bar{A})^*$ to the pair $\theta(w) = (red(w), \{red(w') \in FG(A) : w' \leq_p w\})$, with $\leq_p$ the prefix order on words, is a onto monoid morphism whose induced equivalence is Wagner's congruence, i.e. for all w_1 and $w_2 \in (A + \bar{A})^*$, $\theta(w_1) = \theta(w_2)$ if and only if $w_1 \simeq_W w_2$.

Inverses and idempotents. For every element (u, P), the unique *pseudo-inverse* $(u, P)^{-1}$ of (u, P) can directly be defined by $(u, P)^{-1} = (\bar{u}, \bar{u}P)$. In particular, one has $(u, P) \cdot (u, P)^{-1} = (1, P)$ and $(u, P)^{-1} \cdot (u, P) = (1, \bar{u}P)$. Elements of the form $(1, P) \in FIM(A)$ are the *idempotents* in $FIM(A)$. As a consequence, idempotents are self inverses.

Birooted trees. An element $(u, P) \in FIM(A)$ is *conveniently seen* as a birooted tree which *vertices* are elements of P, with two distinguished vertices: the *input root* 1 and the *output root* u , and with a *directed edge* labeled by $a \in A$ between v and $w \in P$, which is written $v \xrightarrow{a} w$ whenever $va \simeq_W w$ (equivalently $w\bar{a} \simeq_W v$). In the sequel, we identify birooted trees and elements of $FIM(A)$.

For instance, the birooted tree $x = (a\bar{b}, \{1, a, ac, a\bar{b}\})$ is drawn in the following picture,



It can be generated by the *complete walks* $ac\bar{c}\bar{b}$, or $a\bar{b}bc\bar{c}\bar{b}$, or many others. Walk $a\bar{b}$ is a partial walk on x since $\theta(a\bar{b}) = (a\bar{b}, \{1, a, a\bar{b}\})$. One can observe that taking the pseudo inverse of a birooted tree just amount, with the birooted tree point of view, to swap the input and the output roots.

Natural order. Elements of $FIM(A)$ are ordered by the natural order that can be defined by $(u, P) \leq (u', P')$ when $u = u'$ and $P \supseteq P'$. In terms of birooted trees, this just means that there is an embedding of (u', P') in (u, P) that maps the input root 1 and output root u of (u', P') to the input root 1 and output root u of (u, P). It can be shown that $x \leq y$ if and only if $x = xx^{-1}y$ (or $x = yx^{-1}x$). Idempotents elements in $FIM(A)$ are all elements x such that $x \leq 1 = (1, \{1\})$.

Walks in birooted trees. A *(partial) walk* in a birooted tree $(u, P) \in FIM(A)$ is a word $w \in (A + \bar{A})^*$ such that $(u, P) \leq \theta(w)$. In that case word w induces a (partial) traversal of the birooted tree representation of (u, P) starting in the input

root 1 and ending in the output root u. Walk w is a *complete walk* in (u, P) when $\theta(w) = (u, P)$. In that case every vertex in (u, P) is reached at most once by (the traversal induced by) walk w.

Relational monoids. The *relational monoid* associated with any set Q, is defined by $\mathcal{P}(Q \times Q)$, with the product defined as $U \cdot V = \{(q_1, q_2) \in Q \times Q : \exists q \in Q, (q_1, q) \in U, (q, q_2) \in V\}$ for all U and $V \in \mathcal{P}(Q \times Q)$. The neutral element is the diagonal relation $I_Q = \{(q, q) \in Q \times Q : q \in Q\}$.

The inverse and star operators are defined, for every $U \subseteq Q \times Q$, by $U^{-1} = \{(q_1, q_2) \in Q \times Q : (q_2, q_1) \in U\}$ and $U^* = \bigcup_{n \in \mathbb{N}} U^n$ with $U^0 = \{I_Q\}$ and $U^{n+1} = U^n \cdot U$ for every $n \in \mathbb{N}$.

It occurs that the relational monoid ordered by reverse inclusion is particularly well suited for recognizability [9, 8] by means of premorphism.

2 Walking automata

Informally, a walking automaton is a finite state device that walks on a birooted tree, going from vertices to vertices traversing the directed edges reading their labels. At any step, the automaton records some finite amount of information on the history of its walk is its *local state* that ranges over finitely many values. This information is updated at every (foot) step via the notion of state transitions.

By convention, when a walking automaton traverses forward a directed edge labeled by some $a \in A$, from its source vertex to its target, we say that the automaton *read letter* a. When a walking automaton traverses backward an edge labeled by some $a \in A$, from its target vertex to its source vertex, we say that the automaton *read letter* $\bar{a}$, the syntactic inverse of a. For convenience, we may also allow *silent* transitions that stays in the current vertex. It that case, we say that the automaton *read empty word* 1.

Additionally, walking automata are extended with the capacity to perform *subwalks*: walks that *start in a given vertex* and *end in the same vertex*, in such a way that, the *new* information collected during a subwalk can be *combined* with the *old* information previously collected before performing the subwalk.

Definition. A (finite) *walking automaton* on the alphabet A is a tuple $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$ where Q is a (finite) *set of states*, $I \subseteq Q$ is a set of *initial states*, $F \subseteq Q$ is a set of *final states*, $\delta : (A + \bar{A} + 1) \to \mathcal{P}(Q \times Q)$ is a *transition table* possibly with *silent* transitions, and, additionally, $\Delta : Q \times Q \to \mathcal{P}(Q \times Q)$ is a *higher order transition table*.

Since a subwalk can occur within another subwalk, this induces a (limited) *recursion* mechanism that is handled by means of the higher order transition table and a configuration stack.

A *walking automata configuration* is a non empty sequence $C \in ((Q \times Q) \times FG(A))^+$. Intendedly, a configuration $C = ((p_0, q_0), u_0) \cdot ((p_1, q_1), u_1) \cdot \cdots \cdot ((p_k, q_k), u_k)$ describes the situation where k (nested) subruns have been launched. In such a configuration, for every i with $0 \leq i \leq k$, state p_i (resp. state q_i) is the starting state (resp. the current state) of the ith subrun (the initial run seen as the 0th subrun).

Of course, all subruns but the kth are actually stopped, waiting for the termination of the subrun they have launched. Every pair $((p_{i-1}, q_{i-1}), u_{i-1})$ with $0 < i \leq k$ recorded in the configuration stack can thus be interpreted as a ith *pebble* colored by (p_{i-1}, q_{i-1}) that has been dropped to mark the position where the ith subrun has been launched. The position of that ith pebble is recorded by its relative position $u_{i-1} \in FG(A)$ from the starting position of the $(i-1)$th subrun. Word $u_k \in FG(A)$ is the relative current position from the starting position of he kth subrun.

Definition. A *walking automaton step* from a configuration C to a configuration D reading $x \in A + \bar{A} + 1$, which is denoted by $C \xrightarrow{x} D$, is defined when one of the following condition is satisfied:

- *automaton reads x:* $x \in 1 + A + \bar{A}$, $C = E \cdot ((p, q_1), u)$ and there is $(q_1, q_2) \in \delta(x)$ such that $D = E \cdot ((p, q_2), red(u \cdot x))$,
- *automaton drops a pebble and starts a subrun:* $x = 1$, $C = E \cdot ((p, q), u)$ and there is $r \in Q$ such that $D = E \cdot ((p, q), u) \cdot ((r, r), 1)$, i.e. in a state r, pebble (p, q) is dropped on the current vertex (its value and relative position are stored in the stack),
- *automaton lifts a pebble and ends a subrun:* $x = 1$, $C = E \cdot ((p, q_1), u) \cdot ((r, s), 1)$ and there is $q_2 \in Q$ such that $(q_1, q_2) \in \Delta(p, q)$ such that $D = E \cdot (p, q_2), u)$, i.e. the last pebble dropped is lifted and the new current state is computing accordingly.

A *walking automaton run* from a given state p to a given state q is then defined as a sequence ρ of automaton steps $\rho = C_0 \xrightarrow{x_1} C_1 \xrightarrow{x_2} C_2 \cdots C_{n-1} \xrightarrow{x_n} C_n$ such that $C_0 = ((p, p), 1)$, i.e. the run starts in state p with position 1 from the starting point, $C_n = ((p, q), u_\rho)$ with $u_\rho \in FG(A)$, i.e. the run ends in state q at position u_ρ from the entry point of the run, with n defining the *length* $|\rho|$ of run ρ. The *pebble rank* of run ρ is the least integer k such that $|C_i| \leq k + 1$ for every $0 \leq i \leq n$. It other

words, it is the highest number of (true) pebbles that have been used at the same time in run ρ. A run of pebble rank *at most* k is called a *k-run*.

Remark. Observe that ending a subrun, we require that the relative position to the start of the subrun equals 1. This ensures that a subrun ends on the vertex it has started: we are using (so called) weak pebbles [4]. Moreover, since we also require that the last configuration of a run is of length one, this means that, in a run, *every pebble ever dropped has eventually been removed*.

Observe also that, in contrast with classical many pebble automata [7], the pebble automata defined here can only see (and access) the last pebble dropped on the input structure by lifting it. The pebbles used here are thus invisible and, presumably, k-invisible pebble automata have a weaker expressive power than classical k-visible pebble automata.

Given a run ρ as above, the element $u_\rho \in FG(A)$ reached at the end of run ρ is called the *exit root* (or just the *exit*) of run ρ. The word $w_\rho = x_1 x_2 \cdots x_n \in (A+\bar{A})^*$ is the *walk* red (in run ρ) by automaton $\mathcal{A}$ from state p to state q. The set $P_\rho = \{red(u) \in FG(A) : u \leq_p w_\rho\}$ is the *domain* traversed by automaton $\mathcal{A}$ in run ρ. One can check, by induction on the length of runs, that $u_\rho = red(w_\rho) \in P_\rho$. The (well defined) birooted tree $\theta(w_\rho) = (u_\rho, P_\rho)$ is the *birooted tree* red (in run ρ) by automaton $\mathcal{A}$ from state p to state q.

For every $k \in \mathbb{N}$, let $B^k_{p,q}(\mathcal{A}) \subseteq FIM(A)$ (resp. $W^k_{p,q} \subseteq (A+\bar{A})^*$), or just $B^k_{p,q}$ (resp. $W^k_{p,q}$) when $\mathcal{A}$ is implicit from the context, be the set of birooted trees (resp. walks) red by a k-run of $\mathcal{A}$ from state p to state q. Let also $B_{p,q} \subseteq FIM(A)$ (resp. $W_{p,q} \subseteq (A+\bar{A})^*$) be the set of birooted trees (resp. walks) red by a run of $\mathcal{A}$ from state p to state q.

The *language of walks* $W^k(\mathcal{A}) \subseteq (A+\bar{A})^*$ recognized by automaton $\mathcal{A}$ with at most k-pebbles is defined by $W^k(\mathcal{A}) = \sum_{(p,q)\in I\times F} W^k_{p,q}$. The language of walks $W(\mathcal{A}) \subseteq (A+\bar{A})^*$ recognized by automaton $\mathcal{A}$ is then defined by $W(\mathcal{A}) = \sum_{k\in\mathbb{N}} W^k(\mathcal{A})$.

From this point, there are two possible semantics for walking automata on birooted trees: the *complete run semantics* and the *partial run semantics*.

The *complete run semantics* amounts to defining recognized *languages of birooted trees* just by mapping (complete) recognized walks to the birooted trees they induces. In other words, we take $B^k(\mathcal{A}) = \sum_{(p,q)\in I\times F} B^k_{p,q} = \theta(W^k(\mathcal{A})$ with k pebbles at most, and $B(\mathcal{A}) = \sum_{k\in\mathbb{N}} B^k(\mathcal{A}) = \theta(W(\mathcal{A}))$ with no bound on the number of pebbles.

The *partial run semantics*, perhaps more generally considered in the literature [4], amounts to defining the recognized *languages of birooted trees* as the languages of trees with partial accepting walks. With natural order defined on birooted trees, this just amount to taking $C^k(\mathcal{A}) = \{x \in FIM(A) : \exists y \in B^k(\mathcal{A}), x \leq y\}$ with k

pebbles at most and $C(\mathcal{A}) = \{x \in FIM(A) : \exists y \in B(\mathcal{A}), x \leq y\}$ with no bound on the number of pebbles. In other words, the partial walk semantics is the downward closure of the complete walk semantics.

These two possible semantics essentially comes from the fact that frontiers of birooted tree (leaf vertices) are not marked while they are marked in more classical models of trees as these used in [4]. How to cope with this is discussed in the last section.

An easy observation nevertheless shows that the complete run semantics actually embed the partial run semantics.

Lemma 1 *For every finite walking automaton $\mathcal{A}$ there exists a finite walking automaton $\mathcal{A}_p$ such that, for every $k \in \mathbb{N}$, for all $x \in FIM(A)$, $x \in B^{k+1}(\mathcal{A}_p)$ if and only if there exists $y \in B^{k+1}(\mathcal{A})$ with $x \leq y$, i.e. $B^{k+1}(\mathcal{A}_p) = C^{k+1}(\mathcal{A})$.*

Proof. Define the walking automaton $\mathcal{A}_p$ as the automaton that starts by dropping a pebble in order to traverse all the input structure, return to the entry point, lift that pebble, and then just behave as automaton $\mathcal{A}$. □

The following Lemma tells that silent transition can be removed without loss of generality.

Lemma 2 *For every walking automaton $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$ there exists an automaton $\mathcal{A}' = \langle Q, I, F, \delta', \Delta'$ with $\delta'(1) = \emptyset$ such that, for every $k \in \mathbb{N}$, every $(p, q) \in Q \times Q$, $B^k_{p,q}(\mathcal{A}) = B^k_{p,q}(\mathcal{A}')$.*

Proof. For every $a \in A + \bar{A}$, let $\delta'(a) = \delta^*(1) \cdot \delta(A) \cdot \delta^*(1)$ and, for every p and $q \in Q$, let $\Delta'(p, q) = \delta^*(1) \cdot \Delta(p, q) \cdot \delta^*(1)$. Then the equivalence between $\mathcal{A}$ and $\mathcal{A}'$ is easily proved by induction on the lengths of runs. □

3 Walking automata and k-rational languages

In this section we provide a correspondance between walking automata with *complete run semantics* and (some notion of) rational languages. In this correspondance, the maximal number of pebbles allowed in automaton runs matches the nesting depth in rational expressions of some projection operator. Most of the material presented in the section is a generalization of a former work on linear unidirectional birooted trees [5].

The following definition just mimic on subsets of $FIM(A)$ the standard definition of Kleene rational (or regular) languages of finite words.

Definition. A language $B \subseteq FIM(A)$ is rational if it is a finite rational combination of finite languages, i.e. a combination of such languages with sum, product and iterated product (or star).

Theorem 3 *For every language $B \subseteq FIM(A)$, B is rational if and only if $B = B^0(\mathcal{A})$ for some finite walking automaton $\mathcal{A}$.*

Proof. Let $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$ be a finite state walking automaton.

One can observe that $W^0(\mathcal{A})$ is a regular languages on the alphabet $(A + \bar{A})^*$. It follows that, by applying Kleene's theorem, $W^0(\mathcal{A})$ is definable by a rational expression. We conclude then by observing that (the language extension of) morphism θ commutes with sum, product and star operators.

For the sake of completeness, especially using extended expressions below, we also provide a complete argument.

By applying Lemma 2, we may assume that $\delta(1) = 0$. Then, one can prove, as in classical proofs of Kleene theorem, by induction on the length of runs, that $\{B^0_{p,q}\}$ with $(p, q) \in Q \times Q$ is the least solution of the (set) equations

$$B^0_{p,q} = \delta_{p,q} + \sum_{a \in A} \left(\sum_{(p,r) \in \delta(a)} a \cdot B^0_{r,q} + \sum_{(p,r) \in \delta(\bar{a})} a^{-1} \cdot B^0_{r,q} \right)$$

with a denoting the birooted tree $(a, \{1, a\})$, a^{-1} denoting the birooted tree $(\bar{a}, \{1, \bar{a}\})$ and $\delta_{p,q} = 1$ if $p = q$ or $\emptyset$ otherwise.

Next, we conclude, as for classical word automata, solving this system by standard Gaussian elimination of set variables. Indeed, on subsets of $FIM(A)$, products distributes over sum, and, for every languages of birooted trees U and L, X is the least solution of the equation $X = UX + L$ if and only if $X = U^*L$.

Conversely, one can prove by induction on the size of rational expressions that every rational expression can be translated into a walking automaton.

The construction is just straightforward in the ground cases of $\emptyset$, a and a^{-1} for very $a \in A$. The inductive steps are then easily achieved, just as in the case of language of words as described, for instance, in [14], by adding extra initial and final states and combining small automata (with disjoint set of states) into bigger ones with silent transitions. □

Presumably, allowing no pebble in runs of walking automata decreases their expressive power. Can we add another language operator to capture the expressive power of walking automata with pebbles? It is an easy observation that adding the

point-wise inverse operation on languages, i.e. $X^{-1} = \{x^{-1} \in FIM(A) : x \in X\}$, does not increase the class of rational language. Indeed, for every X and $Y \subseteq FIM(A)$, $(X+Y)^{-1} = X^{-1} + Y^{-1}$, $(X \cdot Y)^{-1} = Y^{-1} \cdot X^{-1}$ and $(X^*)^{-1} = (X^{-1})^*$ hence the class of rational languages of birooted trees is closed under inverse.

This is (presumably) no longer true for the projection of languages on their subsets of idempotents. More precisely, idempotents in $FIM(A)$ are all elements of the form $(1, P)$. We can thus define the operator ^{E} by $X^E = \{(u, P) \in X : u = 1\}$ for every $X \subseteq FIM(A)$. This leads us to the following definition of k-rationality.

Definition. A language $B \subseteq FIM(A)$ is k-rational for some $k \in \mathbb{N}$ when either $k = 0$ and B is rational, or $k = p + 1$ for some $p \in \mathbb{N}$ and B is a finite rational combination of languages of the form C or C^E with p-rational languages C.

Again, adding the point-wise inverse operation on languages does not add any expressive power, nor changes the defined hierarchy, since, for every $X \subseteq FIM(A)$, we also have $(X^E)^{-1} = (X^{-1})^E$ since idempotent elements in $FIM(A)$ are self inverses.

Theorem 4 *For every language $B \subseteq FIM(A)$, B is k-rational if and only if $B = B^k(\mathcal{A})$ for some finite walking automaton $\mathcal{A}$.*

Proof. Let $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$ be a finite state two-way automaton. Again, without loss of generality, we may assume $\delta(1) = \emptyset$. The equations defined for $\{B^0_{p,q}\}$ with $(p, q) \in Q \times Q$ in the proof of Theorem 3 still holds and can be completed for $\{B^{k+1}_{p,q}\}$ with $(p, q) \in Q \times Q$ and $k \in \mathbb{N}$ by the (set) equations

$$B^{k+1}_{p,q} = \delta_{p,q} + \sum_{\substack{(p',q') \in Q \times Q \\ (p,s) \in \Delta(p',q')}} C^k_{p',q'} \cdot \left(\sum_{a \in A} \left(\sum_{(s,r) \in \delta(a)} a \cdot B^{k+1}_{r,q} + \sum_{(s,r) \in \delta(\bar{a})} a^{-1} \cdot B^{k+1}_{r,q} \right) \right)$$

with $C^k_{p',q'} = \left(B^k_{p',q'}\right)^E$. Indeed, we just enumerate above the possible starts of runs.

Then, by induction on the length of runs, one can prove that $B^k_{p,q}$s with $(p, q) \in Q \times Q$ and $k \in \mathbb{N}$ form the least solution (w.r.t. set inclusion) of this system. Moreover, since $B^k_{p,q}$ only depends on $B^{k'}_{p',q'}$ with $k' \leq k$, this least solution is still syntactically computable by Gaussian elimination of variables. The fact that the number of allowed pebbles matches the resulting nesting depth of E operator is immediate.

Conversely, building an automaton $\mathcal{A}$ for every k-rational language B such that $B^k(\mathcal{A}) = B$ can be done by induction on the syntactical complexity of rational expressions.

The inductive step for the E language operator, which is not classical, is described as follow. Assume B is k-rational with a finite state automaton $\mathcal{A} = \langle Q, I, F, \delta, \Delta\rangle$ such that $B = B^k(\mathcal{A})$.

Let $\mathcal{A}' = \langle Q', I', F', \delta', \Delta'\rangle$ defined by $Q' = Q \uplus \{q_0, q_F\}$, $I' = \{q_0\}$, $F' = \{q_F\}$, $\delta'(x) = \delta(x)$ for every $x \in A+\bar{A}+1$ and $\Delta'(p,q) = \Delta(p,q)$ when $(p,q) \in Q\times Q - I\times F$, $\Delta'(p,q) = \Delta(p,q) \cup \{(q_0, q_F)\}$ when $(p,q) \in I \times F$, and $\Delta'(p,q) = \emptyset$ otherwise. It then an easy observation that $(u,P) \in B^{k+1}(\mathcal{A}')$ if and only if $(u,P) \in B^k(\mathcal{A})$ and $u = 1$ or, equivalently, $(u,P) \in (B^k(\mathcal{A}))^E$. □

Observe that there we do not provide rational expressions for languages of the form $B(\mathcal{A})$ for finite state walking automata $\mathcal{A}$ with no bound on the number of allowed pebbles. However, it can be shown that:

Theorem 5 *For every language $B \subseteq FIM(A)$, B is definable in MSO if and only if $B = B(\mathcal{A})$ for some finite walking automaton $\mathcal{A}$.*

Proof. This amount to prove that, without bound on the number of allowed pebbles, (invisible) pebble automaton are equivalent to bottom up automata which is known for trees [6]. The arguments for birooted tree languages are essentially the same as the argument for trees presented in [2]. □

In order to complete the picture, let $\mathcal{R}$ be the class of languages of birooted trees definable as pre-images of finite sets by morphisms into finite monoids.

Lemma 6 *Languages of $\mathcal{R}$ are recognizable by 0-pebble walking automata.*

Proof. Let $\varphi : FIM(A) \to S$ with finite S. Without loss of generality we may assume that φ is onto. Let then $X \subseteq S$ and let $\mathcal{A}$ be the walking automaton defined by

$$\mathcal{A} = \langle S, \{1\}, X, \delta, \Delta\rangle$$

with $\delta(a) = \{(s, s\cdot\varphi(a)) \in S \times S : s \in S\}$ for every $a \in A + \bar{A}$ and $\Delta(p,q) = \emptyset$ for every $(p,q) \in S \times S$. Then, one can easily check that $\varphi^{-1}(X) = \mathcal{B}^0(\mathcal{A})$. □

Remark. In the above proof, one can say a little more about monoid S. Indeed, it is itself an inverse monoid.

For every $x \in FIM(A)$, $xx^{-1}x = x$ hence $\varphi(x)\varphi(x^{-1})\varphi(x) = \varphi(x)$. It follows that $\varphi(x)$ and $\varphi(x^{-1})$ are pseudo inverse one of the other, i.e. S is a regular monoid.

It remains to check that idempotents of S commutes. For this, it is enough to check that every idempotent of S is the image of an idempotent of $FIM(A)$.

Let $x \in FIM(A)$ such that $\varphi(x)$ is idempotent. First we observe that $\varphi(x^{-1})$ is also idempotent. Indeed, we have $\varphi(x)\varphi(x) = \varphi(x)$. But since $(xx)(x^{-1}x^{-1})(xx) =$

xx we also have $\varphi(x)\varphi(x^{-1})\varphi(x^{-1})\varphi(x) = \varphi(x)$. Multiplying both sides by $\varphi(x^{-1})$ both on the left and on the right, we obtain $\varphi(x^{-1})\varphi(x)\varphi(x^{-1})\varphi(x^{-1})\varphi(x)\varphi(x^{-1}) = \varphi^{-1}(x)\varphi(x)\varphi(x^{-1})$. But $x^{-1}xx^{-1} = x^{-1}$ hence, by simplification, $\varphi(x^{-1})\varphi(x^{-1}) = \varphi(x^{-1})$.

But again, since $\varphi(x)\varphi(x^{-1})\varphi(x^{-1})\varphi(x) = \varphi(x)$ and since both xx^{-1} and $x^{-1}x$ are idempotent in $FIM(A)$, they commute and thus, $\varphi(x^{-1})\varphi(x)\varphi(x)\varphi(x^{-1}) = \varphi(x)$. By applying idempotence of $\varphi(x)$, we have $\varphi(x^{-1})\varphi(x)\varphi(x^{-1}) = \varphi(x)$. Since $x^{-1}xx^{-1} = x^{-1}$ this shows that $\varphi(x^{-1}) = \varphi(x)$ and thus $\varphi(x) = \varphi(x^{-1}x)$ with $x^{-1}x$ idempotent in $FIM(A)$.

We are now ready to say a little more about the hierarchy induces by walking automata. Denoting by $\mathcal{B}^k$ the class of languages k-rational for every $k \in \mathbb{N}$ and denoting by $\mathcal{B}$ be the class of languages of birooted trees definable by walking automata with no bound on the number of allowed pebbles:

Theorem 7 $\mathcal{R} \subset \mathcal{B}^0 \subset \mathcal{B}^1 \subseteq \cdots \subseteq \mathcal{B}^k \subseteq \mathcal{B}^{k+1} \subseteq \cdots \subseteq \mathcal{B} = MSO$

Proof. Inclusions follows from the Lemma 6 (for the first one) and the definitions (for the other ones). The strictness of the first inclusion, between $\mathcal{R}$ (recognizable languages) and $\mathcal{B}^0$ (rational languages) among subsets of $FIM(A)$ is known for long [16].

Separation between $\mathcal{B}^0$ (rational walking languages) and $\mathcal{B}^1$ (1-pebble walking languages) is proved by the language $E(FIM(A)) = (FIM(A))^E$ of all context elements (or idempotents). Obviously, $E(FIM(A))$ belongs to $\mathcal{B}^1$ while, applying Theorem 3 and with a simple pupping lemma argument, it does not belong to $\mathcal{B}^0$. □

Remark. Restricted to one-dimensional unidirectional birooted trees (called overlapping tiles in [5]), a similar arguments show that $\mathcal{R} \subset \mathcal{B}^0 \subset \mathcal{B}^1$. However, in the case of tile languages, it can be shown that the pebble hierarchy collapses at level one [5], i.e. $\mathcal{B}^1 = \mathcal{B}^{k+1}$ for every $k \in \mathbb{N}$.

Remark. We conjecture that all these inequalities are strict. Of course, the strictness the invisible pebble hierarchy does not follows from the analogous known result on tree walking automata with visible pebble (see [2] for an overview). However, it can still be the case that hard languages requiring k visible pebbles are definable with k invisible pebbles.

4 Walking automata and quasi-recognizability

In this section, following the classic approach relating automata to algebra, we study the mapping that consists, for a given automaton, to map input structures to sets of pairs of states in the transition monoid that accepts the input structure. After discussing briefly the case of the *complete run semantics*, we show that the *partial run semantics* provide a mapping with sufficient properties to induce an algebraic characterization of walking automata.

Let $\mathcal{A}$ be a walking automaton $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$. Without lost of generality, we assume that $\delta(1) = \emptyset$, i.e. there are no silent transitions in $\mathcal{A}$.

Let us first consider mapping α that maps every birooted tree to the pairs of states induced by all *complete runs* of $\mathcal{A}$ on that birooted tree. More precisely, for every $x \in FIM(A)$, let $\alpha(x) \subseteq Q \times Q$ be define by $\alpha(x) = \sum_{k \in \mathbb{N}} \alpha^k(x)$ with $\alpha^k(x) = \{(p, q) \in Q \times Q : x \in B^k_{p,q}\}$ for every $k \in \mathbb{N}$.

Though straightforward, the next lemma is worth being stated for it tells to which extent the standard algebraic approach fails with complete run semantics for walking automata.

Lemma 8 *For all $k \in \mathbb{N}$, we have $\alpha^k(1) = I_Q$ (resp. $\alpha(1) = I_Q$), and, for all x and $y \in FIM(A)$, we have $\alpha^k(xy) \supseteq \alpha^k(x) \cdot \alpha^k(y)$ (resp. $\alpha(xy) \supseteq \alpha(x) \cdot \alpha(y)$).*

Proof. Let $k \in \mathbb{N}$. The fact $\alpha^k(1) = I_Q$ immediately follows from the definition of walking automata semantics and the fact that we assume there are no silent transitions, i.e. $\delta(1) = \emptyset$.

Let then $\gamma^k : (A + \bar{A})^* \to \mathcal{P}(Q \times Q)$, defined for all $w \in (A + \bar{A})^*$ by $\gamma^k(w) = \{(p, q) \in Q \times Q : w \in W^k_{p,q}\}$. By definition of walking automata semantics we have

$$\begin{array}{ccc} & (A + \bar{A})^* & \\ {\scriptstyle\theta}\swarrow & & \searrow{\scriptstyle\gamma^k} \\ FIM(A) & \xrightarrow[\alpha^k = \gamma^k \circ \theta^{-1}]{} & \mathcal{P}(Q \times Q) \end{array}$$

In other words, α^ks are relational morphisms and, since $\theta^{-1}(xy) \supseteq \theta^{-1}(x) \cdot \theta^{-1}(y)$ we have $\alpha^k(xy) \supseteq \alpha^k(x) \cdot \alpha^k(y)$. □

Remark. With $k = 0$, if there is equality then mapping α^0 is a monoid morphism hence $B^0(\mathcal{A})$ is a recognizable language. It follows that automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by taking $\Delta(Q \times Q) = \emptyset$ is na inverse automaton (see [16] for a study of these languages). As these are very specific automata, this is not the case in general.

Remark. Although mappings α^k (and α) are sub-multiplicative they are not, in general, premorphism in the sense of McAlister and Reilly [12]. Indeed, when $x \leq y$ in $FIM(A)$ there may be just no relationship between $\alpha^k(x)$ and $\alpha^k(y)$ hence α^k is in general not monotonic.

Instead of α, let us now consider mapping β that maps every birooted tree to the set of pairs of states induced by *partial runs* of $\mathcal{A}$ (henceforth addressing the *partial run semantics* of $\mathcal{A}$). More precisely, for every $x \in FIM(A)$, let $\beta(x) \subseteq Q \times Q$ be define by $\beta(x) = \sum_{k\in\mathbb{N}} \beta^k(x)$ with $\beta^k(x) = \{(p,q) \in Q \times Q : \exists y \in FIM(A), y \leq x \wedge y \in B^k_{p,q}\}$ for every $k \in \mathbb{N}$. In other words, let $\beta(x)$ (resp. $\beta^k(x)$) be defined as the image of the upper set $\uparrow x = \{y \in FIM(A) : x \leq y\}$ by (the set extension) of mapping α (resp. α^k).

As opposed to α, mappings β and β^k enjoys many more properties.

Lemma 9 *For every $k \in \mathbb{N}$, mapping β^k (and thus mapping β) is a premorphism. It satisfies in particular the following properties:*

1. $\beta^k(1) = I_Q$,
2. *for all x and $y \in FIM(A)$, if $x \leq y$ then $\beta^k(x) \supseteq \beta^k(y)$,*
3. *for all $x \in FIM(A)$, if $x \leq 1$ (equivalently x is idempotent) then $\beta^k(x) \supseteq I_Q$ and $\beta^k(x)$ is idempotent,*
4. *for all x and $y \in FIM(A)$, $\beta^k(xy) \supseteq \beta^k(x) \cdot \beta^k(y)$,*
5. *for all x and $y \in FIMA)$ if $y \leq 1$ and $x = xy$ (resp. $x = yx$) then $\beta^k(x) = \beta^k(x) \cdot \beta^k(y)$ (resp. $\beta^k(x) = \beta^k(y) \cdot \beta^k(x)$).*

Proof. (1) and (2) are immediate from the definition of β^k. Actually, (2) also follows from (3) and (4) since $x \leq y$ is equivalent to $x = yx^{-1}x$ with $x^{-1}x$ idempotent.

(3) follows from (4). Indeed, if $x \leq 1$ then, by (1) and (2), $\beta^k(x) \supseteq I_Q$ henceforth, by (4), $\beta^k(x) = \beta^k(xx) \supseteq \beta^k(x) \cdot \beta^k(x) \supseteq \beta^k(x) \cdot I_Q = \beta^k(x)$.

(4) Let $(p,q) \in \beta^k(x) \cdot \beta^k(y)$. By definition of product of relations, this means there is $r \in Q$ such that $(p,r) \in \beta^k(x)$ and $(r,q) \in \beta^k(y)$ hence $x' \in B^\infty_{p,r}$ and $y' \in B^\infty_{r,q}$ such that $x \leq x'$, $y \leq y'$ and $x'y' \in B^\infty_{p,q}$. But, since $\leq$ is stable under product one has $xy \leq x'y'$ and thus $(p,q) \in \beta^k(x)$.

(5) follows from (3) and (4). Indeed, if $x = yx$ then, by (4), we have $\beta^k(x) = \beta^k(yx) \supseteq \beta^k(y)\beta^k(x)$. But if moreover $y \leq 1$ then, by (3), we have $\beta^k(y) \supseteq I_Q$ hence $\beta^k(y)\beta^k(x) \subseteq I_Q.\beta^k(x) = \beta^k(x)$. The other when $x = xy$ is symmetric. □

The following lemma tells that both $\beta^k(x)$ and $\beta(x)$ reduce to the computation of their values on letters and idempotents.

Lemma 10 *Let $x \in FIM(A)$ with $x = (u, P)$ and $u = a_1 a_2 \cdots a_n$. Let $u_0 = 1$ and let $u_i = a_1 a_2 \cdots a_i$ for $1 \leq i \leq n$ and let $y_i = \theta(u_i)^{-1} \cdot x \cdot x^{-1} \cdot \theta(u_i)$ for $0 \leq i \leq n$. For every $k \in \mathbb{N}$ we have:*

$$\beta(x)^k = \beta^k(y_0)\beta^0(a_1)\beta^k(y_1)\beta^0(a_2)\beta^k(y_2)\cdots\beta^k(y_{n-1})\beta^0(a_n)\beta^k(y_n) \tag{1}$$

and

$$\beta(x) = \beta(y_0)\beta^0(a_1)\beta(y_1)\beta^0(a_2)\beta(y_2)\cdots\beta(y_{n-1})\beta^0(a_n)\beta(y_n) \tag{2}$$

Proof. We first prove equation (1). Let k be a fixed integer. We prove that for every $x = (u, P) \in FIM(A)$, if $u = av$ is the exit root of x, then $\beta^k(x) = \beta^k(xx^{-1}) \cdot \beta^0(a)\beta^k(a^{-1}x)$. Indeed, the exit root of $a^{-1}x$ is v and thus the expected result just follows from an iteration of the above step.

Let thus $x \in FIM(A)$ as above. We have $x = aa^{-1}x = xx^{-1}aa^{-1}x$, hence, by premorphism property (Lemma 9 (4)) and the fact that $\beta^k \supseteq \beta^0 \supseteq \alpha^0$ we do have $\beta^k(x) \supseteq \beta^k(xx^{-1}) \cdot \beta^0(a) \cdot \beta^k(a^{-1}x)$.

The converse inclusion follows from the fact that, in a partial run with k pebble at most from vertex 1 to vertex u in $x = (u, P)$, every dropped pebble on a given vertex must eventually be lifted coming back to that same vertex. In other words, such a partial run can be decomposed into a partial run with at most k pebbles on xx^-1 (described by $\beta^k(xx^{-1})$) followed by a reading (with no pebble) of a (described by $\alpha^0(a) \subseteq \beta^0(a)$), followed by a partial run with at most k pebble on the remaining birooted tree $a^{-1}x$ (described by $\beta^k(a^{-1}x)$). In other words: $\beta^k(x) \subseteq \beta^k(xx^{-1}) \cdot \alpha^0(a) \cdot \beta^k(a^{-1}x) \subseteq \beta^k(xx^{-1}) \cdot \beta^0(a) \cdot \beta^k(a^{-1}x)$.

Equation (2) just follows equation (1) summing up over all $k \in \mathbb{N}$. □

Definition. For every $x = (u, P) \in FIM(A)$, let $D(x) \subseteq FIM(A)$, the *domain* of x, be defined by $D(x) = \{\theta(v)^{-1} \cdot x \cdot x^{-1} \cdot \theta(v) \in FIM(A) : v \in P\}$ and let $A_x \subseteq A + \bar{A}$, the set of *immediate edges* from the input root of x, be defined by $A_x = \{a \in A + \bar{A} : aa^{-1}x = x\}$.

Remark. Observe that all element of $D(x)$ are idempotents. Moreover, they are in a one to one correspondance with the vertices of (the birooted representation of) x and, for every $z \in FIM(A)$, if $zz^{-1}x = x$ then $z^{-1}xx^{-1}z \in D(x)$.

The following lemmas and corollaries tell how to compute β^k and β on idempotents:

Lemma 11 *For every $x \in FIM(A)$, every $k \in \mathbb{N}$, the set $\{\beta^k(y)\}_{y \in D(x)}$ is the least solution (w.r.t. inclusion) of the (finite) set of (monotonic) fixpoint equations*

defined, for every $y \in D(x)$ *by:*

$$\beta^0(y) = I_Q + \left(\sum_{a \in A_y} \beta^0(a) \cdot \beta^0(a^{-1}ya) \cdot \beta^0(a^{-1}) \right) \cdot \beta^0(y) \tag{3}$$

when $k = 0$ *and*

$$\beta^k(y) = I_Q + \left(\Delta\left(\beta^{k-1}(y)\right) + \sum_{a \in A_y} \beta^0(a) \cdot \beta^k(a^{-1}ya)\beta^0(a^{-1}) \right) \cdot \beta^{k+1}(y) \tag{4}$$

when $k > 0$.

Proof. We prove equations (3) and (4). The inclusion $\supseteq$ follows, in both cases $k = 0$ or $k > 0$, from the following facts:

1. $\beta^k(y) \supseteq I_Q$ since y is idempotent (Lemma 9 (3)),
2. for every $a \in A_y$, we have $y = (yy^{-1})a(a^{-1}ya)a^{-1}$ hence, since β^k is a premorphism (Lemma 9 (4)), $\beta^k(y) \supseteq \beta^k(yy^{-1}) \cdot \beta^k(a) \cdot \beta^k(a^{-1}ya) \cdot \beta(a^{-1})$ with both $\beta^k(a) \supseteq \beta^0(s)$ and $\beta^k(a) \supseteq \beta^0(s)$ and inclusion order stable under product,
3. when $k > 0$, $\Delta(\beta^{k-1}(y)) \subseteq \beta^k$, since a pebble that is dropped must be lifted from the same vertex,
4. and $\beta^k(y)$ is idempotent (Lemma 9 (3)), i.e. $\beta^k(y) = \beta^k(y) \cdot \beta^k(y)$.

The reverse inclusion $\subseteq$ follows from the fact that we just mimic all possible cases of partial runs over the idempotent element y that consists to:

1. either doing nothing hence staying in the same state (which is allowed since $y \leq 1$),
2. or, optionally when $k > 1$,
   (a) dropping a pebble at the entry root of y,
   (b) performing a partial run with $k - 1$ pebbles (a partial run described by $\beta^{k-1}(y)$),
   (c) lifting that pebble back (upon returning back to the entry root of y),
   (d) and being ready to keep on performing another partial run on y with k pebbles,

3. or, starting to read a letter while staying domain of y hence:
   (a) reading some letter $a \in A_y$ (a reading described by $\alpha^0(a) \subseteq \beta^0(a)$),
   (b) and then, performing a partial run with k pebbles on $a^{-1}y$ which, by equation 1, amounts to:
      i. performing a run with k pebbles on $a^{-1}ya$ (described by $\beta^k(a^{-1}ya)$) since y is idempotent and thus $a^{-1}y(a^{-1}y)^{-1} = a^{-1}ya$,
      ii. completing that run by reading $\bar{a}$ (described by $\alpha^0(a^{-1}) \subseteq \beta^0(a^{-1})$),
   (c) and being ready to keep on performing another partial run on y with k pebbles,

The fact that $\{\beta^k(y)\}_{y \in D(x)}$ is the least solution of this finite system of equations immediately follows from an induction on the (even) length of partial runs on elements of $D(x)$. □

In other words:

Corollary 12 *For every $x \in FIM(A)$ and every $k \in \mathbb{N}$, $\beta^k(x)$ is effectively computable applying equation (1) and solving, by finite iteration, the finite system of least fixpoint equations (3) and (4) defining $\beta^{k'}(y)$ for every $0 \leq k' \leq k$ and $y \in D(x)$.*

Proof. For a given $x \in FIM(A)$ and a given $k \in \mathbb{N}$, equations (1) and (5) defines a finite set of least (w.r.t. to inclusion) fixpoint equations, defining $\beta^{k'}(y)$ for every $0 \leq k' \leq k$ and $y \in D(x)$, with monotonic function in a finite lattice.

It follows that the Knaster-Tarski fixpoint theorem applies: this system can be solves by a finite iteration procedure, initially taking $\beta^{k'}(y) = \emptyset$. □

Moreover:

Corollary 13 *For every $x \in FIM(A)$, $\beta(x)$ is effectively computable applying equation (2) and solving, by finite iteration, the finite system of least fixpoint equations (5) defined, for all $y \in D(x)$ by:*

$$\beta(y) = I_Q + \left(\Delta^{-1}\left(\beta(y)\right) + \sum_{a \in A_y} \beta^0(a) \cdot \beta(a^{-1}ya) \cdot \beta^0(a^{-1}) \right) \cdot \beta(y) \qquad (5)$$

Proof. By applying the Knaster-Tarski fixpoint theorem, in the finite iteration process computing β^k on the domain of x for a given $x \in FIM(A)$, since $D(x)$ is finite, there exists some $k > 0$ big enough such that, for all $y \in D(x)$, $\beta^k(y) =$

$\beta^{k+1}(y)$ hence $\beta(y) = \beta^k(y) = \beta^{k+1}$. Replacing β^k and β^{k+1} by β in equation (4) we deduce equation (5). □

In other words, mappings β and β^ks are finitely computable on any birooted tree. As they do capture the partial run semantics of walking automata it remains to show, even on birooted trees with unmarked vertices, that the partial run semantics can be used as relevantly as with classic tree walking automata.

5 Walking automata in Rees quotients of the free inverse monoid

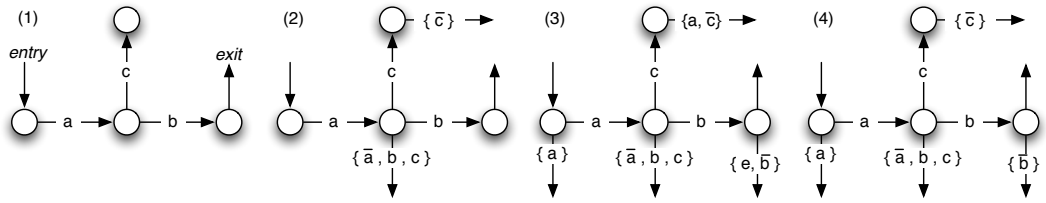
A priori, in the partial run semantics, it is just impossible to define a singleton language nor even a finite languages of birooted trees. Indeed, all definable languages are downward closed with respect to the natural order. This comes from the fact that, in birooted trees, we lack of a way to constrain the branching structure of trees. There is no way to tell that some vertices are leaves (i.e. vertices with at most one edge). Can we cope with that fact? It turns out that a classic algebraic tool, the Rees quotient of monoids, provide a fairly elegant solution to that problem.

Definition. A birooted tree on the alphabet $A+\mathcal{P}(A+\bar{A})$ is a *well-marked birooted* tree when for every vertex there is *at most* one ingoing or outgoing edge labeled by some $C \in \mathcal{P}(A + \bar{A})$ and:

- either this is an ingoing edge and the vertex has no other edges,
- or this is an outgoing edge and the vertex can only have edges of the following type:
  - either outgoing edge labeled by a with $a \in C \cap A$
  - or ingoing edge labeled by a with $\bar{a} \in C \cap \bar{A}$.

A birooted tree on the alphabet $A + \mathcal{P}(A + \bar{A})$ is a *completely well-marked* birooted tree when there is one ingoing or outgoing edge labeled by some $C \in \mathcal{P}(A + \bar{A})$ from *every* vertices. A birooted tree on the alphabet $A + \mathcal{P}(A + \bar{A})$ is a *maximally well-marked* birooted tree when it is completely well-marked and when every dangling edge on the alphabet $\mathcal{P}(A + \bar{A})$ describes all the ingoing or outgoing A-edges from its source vertex.

The following picture gives exemples of (1) unmarked, (2) partially marked, (3) completely marked and (4) maximally marked, well-marked birooted trees.



In other words, a well-marked birooted tree is a birooted tree on the alphabet A extended with optional dangling outgoing edges on $\mathcal{P}(A + \bar{A})$ that tells what is the branching structure of their source vertex.

Lemma 14 *Non well-marked birooted trees form a ideal $\perp_A$ of $FIM(A+\mathcal{P}(A+\bar{A}))$.*

Proof. As soon as a birooted tree is non well-marked its left or right product with any birooted tree yields a non well-marked birooted tree. □

Definition. The monoid $WFIM(A)$ of *well-marked* birooted tree is defined to be the Rees quotient $WFIM(A) = FIM(A+\mathcal{P}(A+\bar{A}))/\perp$, i.e. the monoid $FIM(A+\mathcal{P}(A+\bar{A}))$ where all non well-marked birooted are collapsed into a single $\perp$ element.

Remark. All results stated above can be extended to $WFIM(A)$. Indeed, for the extensions of Kleene theorem, all non well-marked birooted trees can just be removed from languages defined by rational expressions. For the characterization of walking automata semantics by premorphism, the transition monoid $\mathcal{P}(Q \times Q)$ can just be extended with (a new) 0, absorbant element and maximal w.r.t. the inclusion order, mapping $\perp$ in $WFIM(A)$ to 0 in $\mathcal{P}(Q \times Q)$.

Corollary 15 *For every language $B \subseteq WFIM(A)$ of (maximally well-marked) finite birooted trees, if B is definable in MSO then there exists a finite ordered monoid S and a (finitely generated) premorphism $\beta : WFIM(A) \to S$ such that $B = \beta^{-1} \circ \beta(B)$.*

Proof. Take automaton $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$ provided by Theorem 5. One can observe that maximally well-marked birooted trees are minimal elements w.r.t. the natural order among non zero elements of $WFIM(A)$. It follows that complete and partial run semantics for $\mathcal{A}$ coincides. Let then S be the transition monoid $\mathcal{P}(Q \times Q)$ of $\mathcal{A}_p$ extended with a zero (and equipped with the "recursion" mapping

$\Delta^{-1} : S \to S$). Mapping β defined in the previous section recognizes B and, by Corollary 13, it is finitely generated. □

Remark. As a concluding remark, one can observe that the "trick" to describe the branching structure of birooted trees can be extended in order to encode extra vertex labeling on any given finite alphabet.

In other words, up to some encoding via some Rees quotient, (the encoding of) every languages of finite trees definable in MSO is recognizable by means of a premorphism into a finite ordered monoid, i.e. this extend to regular languages of trees the algebraic framework proposed for tiles [9].
Nota: most proofs and some extra examples or remarks are provided in the appendix below.

References

[1] J.-C. Birget. Concatenation of inputs in a two-way automaton. *Theoretical Computer Science*, 63(2):141 – 156, 1989.

[2] M. Bojańczyk. Tree-walking automata. In *LATA*, volume 5196 of *LNCS*. Springer, 2008.

[3] M. Bojańczyk and T. Colcombet. Tree-walking automata do not recognize all regular languages. In *STOC*. ACM, 2005.

[4] M. Bojańczyk, M. Samuelides, T. Schwentick, and L. Segoufin. Expressive power of pebble automata. In *Automata, Languages and Programming (ICALP)*, 2006.

[5] A. Dicky and D. Janin. Two-way automata and regular languages of overlapping tiles. Technical Report RR-1463-12, LaBRI, Université de Bordeaux, 2012.

[6] J. Engelfriet, H. J. Hoogeboom, and B. Samwel. XML transformation by tree-walking transducers with invisible pebbles. In *Principles of Database System (PODS)*. ACM, 2007.

[7] J. Engelfriet and H.J. Hoogeboom. Tree-walking pebble automata. In J. Karhumäki, H. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are forever, contributions to Theoretical Computer Science in honor of Arto Salomaa*, pages 72–83. Springer-Verlag, 1999.

[8] D. Janin. Quasi-inverse monoids (and premorphisms). Technical Report RR-1459-12, LaBRI, Université de Bordeaux, 2012.

[9] D. Janin. Quasi-recognizable vs MSO definable languages of one-dimentionnal overlaping tiles. In *Mathematical Foundations of computer Science (MFCS)*, volume 7464 of *LNCS*, pages 516–528, 2012.

[10] D. Janin. On languages of one-dimensional overlapping tiles. In *International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume (to appear) of *LNCS*, 2013.

[11] M. Kunc and A. Okhotin. Describing periodicity in two-way deterministic finite automata using transformation semigroups. In *Developments in Language Theory*, volume 6795 of *Lecture Notes in Computer Science*, pages 324–336. Springer, 2011.

[12] D.B. McAlister and N. R. Reilly. E-unitary convers for inverse semigroups. *Pacific Journal of Mathematics*, 68:178–206, 1977.

[13] W. D. Munn. Free inverse semigroups. *Proceeedings of the London Mathematical Society*, 29(3):385–404, 1974.

[14] J.-E. Pin. Mathematical foundations of automata theory. Lecture notes, 2011.

[15] H. E. Scheiblich. Free inverse semigroups. *Semigroup Forum*, 4:351–359, 1972.

[16] P. V. Silva. On free inverse monoid languages. *ITA*, 30(4):349–378, 1996.