Walking automata in free inverse monoids

David Janin

Walking automata in free inverse monoids

David Janin

UMR CNRS LaBRI, Inria Bordeaux,
Bordeaux INP, University of Bordeaux
F-33405 Talence, FRANCE
janin@labri.fr

Abstract Walking automata, be they running over words, trees or even graphs, possibly extended with pebbles that can be dropped and lifted on vertices, have long been defined and studied in Computer Science. However, questions concerning walking automata are surprisingly complex to solve. In this paper, we study a generic notion of walking automata over graphs whose semantics naturally lays within inverse semigroup theory. Then, from the simplest notion of walking automata on birooted trees, that is, elements of free inverse monoids, to the more general cases of walking automata on birooted finite subgraphs of Cayley's graphs of groups, that is, elements of free E-unitary inverse monoids, we provide a robust algebraic framework in which various classes of recognizable or regular languages of birooted graphs can uniformly be defined and related one with the other.

1 Introduction

General context. Walking automata, be they running over words, trees or even graphs, possibly extended with pebbles, have long been defined and studied in Computer Science[7,8]. For instance, tree walking automata with pebbles have been an important subject of study the last decades since they are natural abstract models of machine for XML query languages such as XPATH, or XML transformation languages such as XSL [6].

Although based on well studied computation models: finite state machines or pushdown automata, questions about walking automata are often surprisingly complex to solve and, to a lesser extent, quite dependent on such or such details in automata's definition.

For instance, in the case of tree languages, bounding the number of pebbles an automaton leads to defining classes of recognizable languages. Various logical characterizations of these classes have been obtained [8] and difficult separation results have also been proved [2,3,1]. However, for separation results, proof arguments apply to the case of pebbles that are marked and visible [2,3], leaving open the cases of unmarked and/or invisible pebbles.

Even though walking automata are sequential machines much like string automata, the classical algebraic tools that have been developed to study word automata are not easily applicable to tree walking automata. Despite numerous

results, little is known about the underlying mathematical framework, say in algebra, that walking automata may induce.

Contribution of the paper. In this paper, we initiate the development of an algebraic framework, within inverse semigroup theory, for walking automata. We provide a generic notion of automata walking on edge-labeled graphs. They act as some kind of observers of their input graphs much in the same way observational semantics has been defined in concurrency theory by Hennessy and Milner [17].

Unlike most classical definitions, we do not require walking automata to start and end in the same vertex, neither do we require the complete traversal of input structures. Moreover, the capacity given to a walking automaton to check or not the absence of an (incoming or outgoing) edge labeled by a given letter induces two possible semantics for walking automata, much in the same way there are various observational semantics in [17], with or without observable failures.

The languages recognized by our walking automata are languages of birooted graphs, that is, graphs extended with an input root: the vertex where the run starts, and an output root: the vertex where the run ends. These languages are shown to be closed under root preserving graph morphisms (Lemma 12). Moreover, the sequential composition of partial runs of walking automata is shown to induce a composition of the traversed birooted graphs, yielding an inverse monoid structure (see Remark 3 and Theorem 22).

Then we prove (Theorem 16) that, in many cases, the stronger semantics (with observable reading failures) can be uniformly encoded in the weaker semantics (with unobservable reading failures).

As particular cases, walking automata in Cayley's graphs of finitely generated groups are considered in Section 4. The recognized languages are subsets of monoids known as freest E-unitary inverse monoids [15,16]. Then, based on the underlying monoid structure, an extension of regular expressions is defined and shown (Theorem 25) to characterize the classes of recognizable languages induced by limiting numbers of allowed pebbles.

2 Graphs

It is very likely that most concepts and properties detailed here have already appeared in the literature. However, for the sake of completeness we provide our own presentation.

Graphs and morphisms. Let $A = \{a, b, c, \cdots\}$ be a finite alphabet. A *(relational) graph* on the edge alphabet A is a pair $G = \langle V, E\rangle$ with a set of vertex V and sets of a-labeled (directed) edges $E(a) \subseteq V \times V$ for all $a \in A$. A *graph morphism*, or simply morphism, from $G_1 = \langle V_1, E_1\rangle$ to $G_2 = \langle V_2, E_2\rangle$ is a mapping $f : V_1 \to V_2$ such that $f(E_1(a)) \subseteq E_2(a)$, for every $a \in A$. Such a morphism is denoted by $f : G_1 \to G_2$.

Walking paths. Let $\bar{A} = \{\bar{a}, \bar{b}, \bar{c}, \cdots\}$ be a copy of the alphabet A. Let $(A+\bar{A})^*$ be the free monoid generated by $A + \bar{A}$ with the unit (empty word) denoted by 1 and the concatenation of two words $u, v \in (A + \bar{A})^*$ simply denoted by uv.

A (back and forth) walking *path* on the graph G from a vertex x to a vertex y is an alternating sequence of vertices of V and letters of $A + \bar{A}$ of the form

$$\pi = x_0 z_1 x_1 z_2 x_2 \cdots x_{n-1} z_n x_n$$

such that $x = x_0$, $y = x_n$ and, for every $1 \leq i \leq n$, we have $(x_{i-1}, x_i) \in E(z_i)$ where, for every $a \in A$, the relation $E(\bar{a})$ denotes the inverse relation $E^{-1}(a) = \{(x, y) \in V \times V : (y, x) \in E(a)\}$. The vertex x is the *source* of such a path. It is denoted by $\boldsymbol{sr}(\pi)$. The vertex y is the *target* of such a path. It is denoted by $\boldsymbol{tg}(\pi)$.

In such path, a letter $a \in A$ models a *forward traversal* of an a-labeled edge and a letter $\bar{a} \in \bar{A}$ models a *backward traversal* of an a-labeled edge. The *inverse path* π^{-1} of the path π is defined by

$$\pi^{-1} = x_n z_{n-1}^{-1} x_{n-1} \cdots x_2 z_1^{-1} x_1 z_0^{-1} x_0$$

with $(a)^{-1} = \bar{a}$ and $(\bar{a})^{-1} = a$ for every $a \in A$. We easily observe that π^{-1} is indeed a walking path in the graph G from x_n to x_0.

As a particular case, the graph G is *bideterministic* when for every $z \in A+\bar{A}$, for every $(p, q), (p', q') \in E(z)$, if $p = p'$ then $q = q'$. In this case, every path π as above, emanating from a given vertex x, is completely determined by its source x and the path label $\lambda(\pi) = z_1 z_2 \cdots z_n \in (A + \bar{A})^*$ obtained from π by deleting all vertices.

Path-induced birooted subgraph. Let $\pi = x_0 z_1 x_1 z_2 x_2 \cdots x_{n-1} z_n x_n$ be a path on the graph G. The *subgraph* $G|\pi$ of graph G induced by the path π is defined by $G|\pi = \langle V|\pi, E|\pi\rangle$ with the set of vertices $V|\pi = \{x_0, x_1, \cdots, x_n\}$ and, for every $a \in A$, the set of a-labeled edges $(E|\pi)(a)$ defined as the set of pairs $(x, y) \in V' \times V'$ such that either xay or $y\bar{a}x$ occurs as a subsequence in the path π. Then, the following lemma is immediate.

Lemma 1. *The graph $G|\pi$ induced by the path π is finite and the inclusion mapping $\iota : V|\pi \to V$ that maps every vertex to itself is a one-to-one morphism, i.e. graph $G|\pi$ is a finite* subgraph *of graph G.*

Definition 2 (Birooted induced subgraphs). Let π be a path of G. The triple $\theta_G(\pi) = (G|\pi, \boldsymbol{sr}(\pi), \boldsymbol{tg}(\pi))$ defined by distinguishing the source and the target of the path π in the subgraph $G|\pi$, is called the *birooted subgraph* of G induced by the path π.

Remark 3. It is an easy observation that the trivial birooted subgraph product $\theta_G(\pi_1) \cdot \theta_G(\pi_2)$ defined to be the graph $(G|\pi_1 \cup G|\pi_2, \boldsymbol{sr}(\pi_1), \boldsymbol{tg}(\pi_2))$ (with union defined over subsets of vertices and edges) when $\boldsymbol{tg}(\pi_1) = \boldsymbol{sr}(\pi_2)$ and 0 otherwise (with 0 an additional zero element) yields an inverse semigroup: for every element x there is a unique element x^{-1} such that $xx^{-1}x = x$ and $x^{-1}xx^{-1} = x^{-1}$.

Indeed, we have $0^{-1} = 0$ and $\theta_G(\pi)^{-1} = \theta_G(\pi^{-1})$, and, additionally, it can be shown that the non zero idempotent elements exactly correspond to the birooted graphs induced by cyclic paths.

The much more interesting case when the birooted subgraphs are invariant under translation (in Cayley's graphs) is detailed in Section 4 below.

Vertex-labeled graphs. So far, the graphs we consider have no vertex label. The following definition and lemma shows that this fact does not reduce the generality of our language theoretical study.

Definition 4 (Induced vertex label). The *vertex label* of a vertex $x \in V$ in a graph $G = \langle V, E\rangle$ is defined to be the set $\lambda_V(x) = \{a \in A : (x, x) \in E(a)\}$.

The following lemma, whose proof is immediate, emphasizes the relevance of this notion.

Lemma 5. *Let $G = \langle V, E\rangle$ be a graph on the edge alphabet A. Let $g : V \to \mathcal{P}(B)$ be a vertex labeling function with some new alphabet B disjoint from A. Let $\langle G, g\rangle$ be the resulting vertex-labeled graph and let $\varphi(\langle G, g\rangle) = \langle V', E'\rangle$ be the edge-labeled graph defined by $V' = V$, by $E'(a) = E(a)$ for every $a \in A$, and $E'(b) = \{(x, x) \in V' \times V' : b \in g(v)\}$ for every $b \in B$.*

Then, the vertex identity mapping from V into V' is a one-to-one and onto graph morphism from G into $\varphi(\langle G, g\rangle)$. Moreover, φ is a one-to-one mapping from the class of graphs with A-labeled edge and $\mathcal{P}(B)$-labeled vertices into the class of $(A \cup B)$-labeled edges such that, given the vertex labeling $\lambda'_V : V' \to \mathcal{P}(A + B)$ as defined above, then, for every $v \in V' = V$, we have $g(v) = \lambda'_V(v) \cap B$.

In other words, graphs with vertex labels are easily encoded into graphs without vertex labels. Moreover, in the case both A and B are finite, such a mapping induces a fairly simple MSO-transduction (see [4] Chap. 7). It follows that every MSO-definable language of graphs with A-labeled edges and $\mathcal{P}(B)$-labeled vertices can be encoded into an MSO-definable language of graphs with $A \cup B$-labeled edges. Since graphs with unlabeled vertices are particular case of graphs with labeled vertices this really says that, up to MSO definable languages, studying languages of edge-labeled graphs or languages of edge-and-vertex-labeled graphs is essentially the same.

3 Walking on graphs

In this paper, a walking automata is sort of a graph observer that traverses the input graph possibly dropping and lifting (in the reverse order) some pebbles. Since walking automata cannot jump between disconnected graphs, all graphs considered from this point are assumed to be connected via walking paths.

Definition 6 (Walking automata with pebbles). A *walking automata with pebbles* on the alphabet A is a tuple $\mathcal{A} = \langle Q, I, T, \delta, \Delta\rangle$ with set of states Q, initial states $I \subseteq Q$, terminal states $T \subseteq Q$, edge transitions $\delta(z) \subseteq Q \times Q$ for every $z \in A + \bar{A}$, and pebble transitions $\Delta((r, s)) \subseteq Q \times Q$ for every $(r, s) \in Q \times Q$.

Informally, from a given vertex, the automaton can traverse forward any outgoing a-labeled edge (reading a) or it can traverse backward any incoming a-labeled edge (reading $\bar{a}$). In both cases, the automaton state is updated according to the first-order transition function δ applied to the traversed edge label a or $\bar{a}$. Additionally, the automaton may drop a pebble on that vertex, interrupting the current run and starting a new subrun. It may also lift a pebble, ending the current subrun and resuming the former run. When ending a subrun, the automaton state is updated according to the second-order transition function Δ applied to the pair of states resulting from the start state and the stop state of the subrun.

Definition 7 (Automaton configuration). Let $\mathcal{A} = \langle Q, I, T, \delta, \Delta \rangle$ be a walking automaton. Let $G = \langle V, E \rangle$ be a graph on the alphabet A. An *automaton configuration* $\Gamma \in (Q \times Q \times V)^+$ is a non-empty stack of (dot separated) triples over $Q \times Q \times V$.

In a stack of the form $\Gamma.(p, q, x)$ with $p, q \in Q$ and $x \in V$, the triple (p, q, x) describes the current run configuration: *from state p, the automaton $\mathcal{A}$ walked to the current vertex x reaching current state q*. The additional stack Γ, possibly empty, contains the configurations of formerly interrupted runs.

As formalized in the next definition, when dropping a pebble on a vertex x, the automaton *interrupts* the current run, *pushes* its configuration (p_1, q_1, x) on the stack, and *starts a subrun* in a configuration (p_2, p_2, x).

On the contrary, when lifting a pebble from a vertex x, the automaton *terminates* a subrun in a configuration (p_1, q_1, x), *pops* the saved configuration (p_2, s, x), and *resumes* the former run in an updated configuration (p_2, q_2, x), chosen according to the (second-order) transition condition $(s, q_2) \in \Delta((p_1, q_1))$.

Definition 8 (Automaton transition and run). On a graph $G = \langle V, E \rangle$, a *transition step* from a configuration $\Gamma_1.(p_1, q_1, x)$ to a configuration $\Gamma_2.(p_2, q_2, y)$ reading $z \in \{1\} \cup A \cup \bar{A}$ is defined according to one of the following three cases:

(1) edge traversal: $z \in A \cup \bar{A}$, $\Gamma_1 = \Gamma_2$, $p_1 = p_2$, $(q_1, q_2) \in \delta(z)$ and $(x, y) \in E(z)$,
(2) pebble drop: $z = 1$, $y = x$, $\Gamma_2 = \Gamma_1 \cdot (p_1, q_1, x)$ and $q_2 = p_2$,
(3) pebble lift: $z = 1$, $y = x$, $\Gamma_1 = \Gamma_2 \cdot (p_2, s, x)$ and $(s, q_2) \in \Delta((p_1, q_1))$.

Such a transition step is denoted by $\Gamma_1.(p_1, q_1, x) \vdash_z \Gamma_2.(p_2, q_2, y)$.

An *run* of the automaton $\mathcal{A}$ on the graph G from a vertex x to a vertex y is then defined as a sequence of transition steps

$$\rho = \Gamma_0.(p_0, q_0, x_0) \vdash_{z_1} \Gamma_1.(p_1, q_1, x_1) \cdots \vdash_{z_n} \Gamma_n \cdot (p_n, q_n, x_n)$$

with $x_0 = x$ and $x_n = y$, also denoted by $\rho = \Gamma_0 \cdot (p_0, q_0, x_0) \vdash_u^* \Gamma_n \cdot (p_n, q_n, x_n)$ with $u = z_1 z_2 \cdots z_n$. The *path* $\pi(\rho)$ induced by run ρ is defined by

$$\pi(\rho) = x_0 z_1 x_1 z_2 x_2 \cdots x_{n-1} z_n x_n$$

Such a run ρ is an *accepting run* from x to y when Γ_0 is the empty stack with $p_0 = q_0 \in I$ (the first configuration is initial), and Γ_n is the empty stack with $p_n \in I$ and $q_n \in T$ (the last configuration is terminal).

Given an integer $k \geq 0$, the run ρ is *k-accepting run* when it is accepting and $|\Gamma_i| \leq k$ for every $0 \leq i \leq n$, where $|\Gamma_i|$ is the length of the sequence Γ_i. For notational purpose, an accepting run with no bound on the number of allowed pebbles is also called an *∞-accepting run*.

Example 9. An example of run is depicted Figure 1 where configuration stacks are depicted vertically.

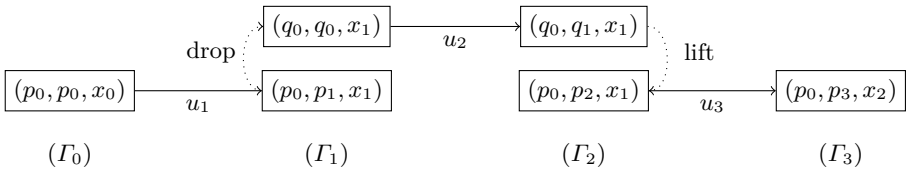

Figure1. A run $\Gamma_0 \vdash^*_{u_1} \Gamma_1 \vdash^*_{u_2} \Gamma_2 \vdash^*_{u_3} \Gamma_3$.

In this run, when no other pebble but the one depicted above is used, then first order transition conditions imply that $(p_0, p_1) \in \delta(u_1)$, $(q_0, q_1) \in \delta(u_2)$ and $(p'_1, p_2) \in \delta(u_3)$, with an obvious extension of δ to $(A + \bar{A})^*$, and second order transition conditions imply that $(p_1, p'_1) \in \Delta((q_0, q_1))$.

Definition 10 (Recognized languages). Given a class of graphs $\mathcal{G}$ (possibly omitted when clear from the context), the language *recognized* (resp. k-recognized) by the automaton $\mathcal{A}$ in the class of graph $\mathcal{G}$ is the set $L^\infty_\mathcal{G}(\mathcal{A})$ (resp. $L^k_\mathcal{G}(\mathcal{A})$) of birooted graphs (G, x, y) with $G \in \mathcal{G}$ and x, y two vertices of G, such that there is an accepting run (resp. a k-accepting run) of the automaton $\mathcal{A}$ over G from x to y.

Remark 11. The walking automata defined here are walking automata with unmarked and invisible pebbles in the sense of [6]. However, generalizing Pécuchet's study of two-way automata on strings [18] (see also [11,5]), we do not require that accepting runs starts and ends in the same vertex of the input structures. Moreover, our definition also differs from the definition proposed in [6,1] in the sense that, a priori, the absence of edges *cannot* be detected by the automata and the walking automaton is not required to traverse the entire structure. The consequences of these facts are discussed below.

Lemma 12. *Let $\mathcal{A}$ be a walking automaton on the alphabet A. Let $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ be two graphs on the same alphabet. Assume that there is a graph morphism $f : G_1 \to G_2$. Then, for every $0 \leq k \leq \infty$ and $x, y \in V_1$, if $(G_1, x, y) \in L^k(\mathcal{A})$ then $(G_2, f(x), f(y)) \in L^k(\mathcal{A})$.*

In general, the converse does not hold. However, as detailed below, the converse holds in the case the graph G_1 is the subgraph of G_2 induced by an accepting run.

Definition 13 (Graphs induced by a run). Let $\rho : \Gamma_1 \vdash \Gamma_2$ be a run in a graph G from x to y. The *graph induced by a run ρ* is defined to be the subgraph $G_\rho = G|\pi(\rho)$ induced by the path $\pi(\rho)$ traversed by $\mathcal{A}$ in G.

Then we have:

Lemma 14. *Let G be a graph such that $(G, x, y) \in L^k(\mathcal{A})$ via an accepting run ρ. Let G_ρ be the graph induced by the run ρ. Then $(G_\rho, x, y) \in L^k(\mathcal{A})$.*

The more classical notion of accepting runs defined by complete traversals of the input structure can be related with ours as follows.

Definition 15 (Strict recognizability). A birooted graph (G, x, y) is *strictly recognized* (resp. *strictly k-recognized*) by an automaton $\mathcal{A}$ when there is an accepting (resp. k-accepting) run ρ of $\mathcal{A}$ over G from x to y such that (G_ρ, x, y) and (G, x, y) are isomorphic.

As an immediate corollary of Lemma 1, Lemma 12 and Lemma 14, we thus have:

Theorem 16. *Let $\mathcal{A}$ be a walking automaton. For every $k \geq 0$, let $L^k_S(\mathcal{A})$ be the class of birooted graphs strictly k-recognized by $\mathcal{A}$ and let $L^k(\mathcal{A})$ the class of birooted graphs k-recognized by $\mathcal{A}$. Then $L^k(\mathcal{A})$ is the morphism closure of the language $L^k_S(\mathcal{A})$, that is, $(G, x, y) \in L^k(\mathcal{A})$ if, and only if, there exists $(G', x', y') \in L^k_S(\mathcal{A})$ and a graph morphism $f : G' \to G$ such that $f(x') = x$ and $f(y') = y$.*

In particular, when the birooted structures cannot be related by morphisms (as with end markers in two-way word automata [5]), studying strict recognizability just amounts to study recognizability.

4 Walking in Cayley's graphs of groups

So far, we have not much used the fact that walking automata recognize sets of birooted graphs. When the underlying graph G is the Cayley's graph of a (presented) group, then the (isomorphic classes of) finite birooted subgraphs induced by paths form a inverse monoid (see [15] for more details and also [16] for a general presentation). Based on the underlying monoid structure, various classes of languages can then be defined and characterized by means of certain restriction of walking automata.

Definition 17 (The Cayley graph of a presented group). Let G be a group generated by $A \subseteq G$ and let $\varphi : (A + \bar{A})^* \to G$ be the corresponding inverse-preserving monoid morphism[1]. Then, the *Cayley graph* of the presented group G is defined to be the graph $C_G = \langle V, E \rangle$ with vertex set defined by $V = G$ and, for every $a \in A$, edge set defined by $E(a) = \{(x, y) \in V \times V : x \cdot \varphi(a) = y\}$.

For convenience, we extend the edge relation function E to $(A + \bar{A})^*$ by taking $E(u) = \{(x, y) \in G \times G : x \cdot \varphi(u) = y\}$. As a particular case, since φ is inverse-preserving and G is a group, we indeed have $E(\bar{a}) = E(a)^{-1}$ which is consistent with our previous extension of the edge relations.

[1] The group G is *presented* by the morphism φ.

Remark 18. Clearly, the Cayley graph C_G of the group G is a (possibly infinite) bideterministic graph as well as its (finite) birooted subgraphs. Depending on the chosen group, various interesting examples can be defined (see [16]).

For instance, taking the free group $FG(A)$ we have *birooted trees*. Taking the group defined from $A = \{a, b, c, d\}$ by $cc = 1$, $dd = 1$ and $cd = 0$, we obtain *vertex-labeled birooted trees* with edges labeled by a or b and, following Remark 5, vertices labeled over $\mathcal{P}(\{c, d\})$. Thanks to the axiom $cd = 0$, only the birooted graph encoding zero has a vertex labeled by both c and d. The language theory of these vertex-labeled birooted graphs has been studied in [10,14].

Another example, taking the group defined from $A = \{a, b, c, d\}$ by $ab = ba$, $cc = 1$ and $dd = 1$ and $cd = 0$, we obtain *vertex-labeled birooted grids* with (say) horizontal edges labeled by a, vertical edges labeled by b, and, similarly, vertices labeled over $\mathcal{P}(\{c, d\})$.

In other words, this group-theoretic based approach to graphs leads to a vast variety of classes of birooted graphs.

Definition 19 (Induced graphs revisited). Let G be a group presented by a morphism φ, and let C_G be its Cayley graph. For every $u \in (A + \bar{A})^*$, let $C_G|u$ be the graph induced by u defined by $C_G|u = \langle V, E\rangle$ with set of vertices $V = \varphi(Pref(u))$ where $Pref(u) = \{v \in (A + \bar{A})^* : \exists w \in (A + \bar{A})^*, u = vw\}$ is the set of word prefixes of u, and sets of edges $E(a)$ defined as the union of $\{(\varphi(v_1), \varphi(v_2)) \in V \times V : v_1 a = v_2\}$ and $\{(\varphi(v_2), \varphi(v_1)) \in V \times V : v_1 a^{-1} = v_2\}$.

The next lemma, whose proof is immediate, relates our two definitions of induced subgraphs.

Lemma 20. *Let π be a path in C_G. Let $\lambda(\pi) \in (A + \bar{A})^*$ be the word of $(A + \bar{A})^*$ obtained from π by deleting all vertices. The birooted subgraph $\theta_{C_G}(\pi)$ induced by the path π isn the graph C_G is isomorphic to the birooted graph $(C_G|\lambda(\pi), 1, \varphi(u))$.*

This leads us to the following definition:

Definition 21 (Birooted subgraphs and their product). A birooted finite subgraph of the Cayley graph C_G of the presented group G is a quadruple $B = (V, E, 1, x)$ where $V \subseteq G$ is a finite subset of G such that $1, x \in V$, $E(a) \subseteq \{(x, y) \in V \times V : x \cdot \varphi(a) = y\}$ for every $a \in A$, and such that, the resulting subgraph $\langle V, E\rangle$ is connected. The set of such finite birooted subgraphs of C_G is denoted by $BSG(G)$. Then, the product of two birooted finite subgraphs $B_1 = (V_1, E_1, 1, x_1)$ and $B_2 = (V_2, E_2, 1, x_2)$ is defined by

$$B_1 \cdot B_2 = (V_1 \cup x_1 \cdot V_2, E, 1, x_1 \cdot x_2) \quad \text{with} \quad E(a) = E_1(a) \cup x_1 \cdot E_2(a)$$

with the notation $x_1 \cdot E_2(a) = \{(x, y) \in (x_1 \cdot V_2, x_1 \cdot V_2) : (x, y) \in E_2(a)\}$ for every $a \in A$.

Theorem 22 (Margolis,Meakin [15]). *The set $BSG(G)$ with birooted graph product is an inverse monoid. The mapping $\theta_G : (A + \bar{A})^* \to BSG(G)$ is an onto monoid morphism, and, for every $u \in (A + \bar{A})^*$, we have $\theta_G(u)^{-1} = \theta_G(u^{-1})$ and $\theta_G(u)$ is idempotent if and only if $\varphi(u) = 1$.*

Remark 23. In [15], it is proved that $BSG(G)$ is the freest inverse monoid generated by A whose group image is the group G. This result is much stronger than Theorem 22.

A subset $X \subseteq BSG(C_G)$ of the monoid $BSG(G)$ is called a *G-language*. Following our previous definitions, given $0 \leq k \leq \infty$, the G-language X is k-recognized (resp. strictly k-recognized) by a walking automaton $\mathcal{A}$ when $X = L^k(\mathcal{A}) \cap BSG(C_G)$ (resp. $X = L^k_S(\mathcal{A}) \cap BSG(C_G)$). Then, let k-PWA (resp. k-PWA^S) be the class of G-languages k-recognized (resp. strictly k-recognized) by a finite state walking automaton.

We aim at providing a Kleene-like characterization of these classes of languages by means of regular expressions. For such a purpose, the following operations are defined over G-languages:

(1) sum : $X_1 + X_2 = X_1 \cup X_2$,
(2) product: $X_1 \cdot X_2 = \{x_1 \cdot x_1 \in BSG(G) : x_1 \in X, x_2 \in X\}$,
(3) star: $X^* = \bigcup X^n$,
(4) inverse: $X^{-1} = \{x^{-1} \in BSG(G) : x \in X\}$,
(5) idempotent projection: $X^E = \{x \in X : xx = x\}$,

for all languages $X, X_1, X_2 \subseteq BSG(G)$.

A *k-regular expression* is defined to be any finite expression built over the alphabet $A \cup \bar{A} \cup \{1\}$, combined with sum, product, star and idempotent restriction operators such that the nesting depth of idempotent projection is at most k. A language $X \subseteq BSG(G)$ is a *k-regular language* when it can be defined by a k-regular expressions, mapping 1 to $\theta_G(1)$ and every letter $z \in A + \bar{A}$ to its birooted image $\theta_G(z) \in BSG(G)$.

The class of k-regular languages is denoted by k-REG. The class of languages recognizable by finite monoids M and morphisms from $BSG(G)$ onto M is denoted by REC. Observe that, by definition, the usual class REG of languages definable by finite Kleene regular expressions equals 0-REG. Last, for every class of languages X, let $X^{\downarrow}$ be the class of closure of the languages of X under root-preserving graph morphisms within $BSG(G)$.

Remark 24. Observe that the notion of k-recognizability is not necessarily preserved under (inverse) monoid morphisms. Indeed, given an inverse monoid morphism $\varphi : M \to N$, we certainly have $\varphi(X^E) \subseteq \varphi(X)^E$ for every $X \subseteq M$. However, the reverse inclusion may be false as illustrated by the expression $ab\bar{a}\bar{b}$. Indeed, it induces a non-idempotent birooted tree in the free inverse semigroup but a cycle in any E-unitary inverse semigroup induced by a group in over which the equation $ab = ba$ is satisfied.

Theorem 25 (Hierarchy). *For every presented group G generated by A, the following equalities and inequalities holds. In this figure, strict inequalities $\subset$ are only known to hold in the free inverse monoid, that is, when $G = FG(A)$: the*

free group generated by A. They have to be read non-strict in all other cases.

$$\begin{array}{rccccccc}
 & 0\text{-}PWA^S & \subset & 1\text{-}PWA^S & \subseteq & \cdots k\text{-}PWA^S \cdots & \subseteq & \omega\text{-}PWA^S \\
 & \| & & \| & & \| & & \\
REC \subset & 0\text{-}REG & \subset & 1\text{-}REG & \subseteq & \cdots k\text{-}REG \cdots & \subseteq & \bigcup_k k\text{-}REG \\
 & & & \cup & & \cup & & \cup \\
 & 0\text{-}REG^{\downarrow} & \subset & 1\text{-}REG^{\downarrow} & \subseteq & \cdots k\text{-}REG^{\downarrow} \cdots & \subseteq & \bigcup_k k\text{-}REG^{\downarrow} \\
 & \| & & \| & & \| & & \\
 & 0\text{-}PWA & \subset & 1\text{-}PWA & \subseteq & \cdots k\text{-}PWA \cdots & \subseteq & \omega\text{-}PWA
\end{array}$$

Proof. Each horizontal inclusion follows from the definition. The separation result $REC \subset 0\text{-}REG$ is known over languages of birooted trees [19]. The separation $0\text{-}PWA^S \subset 1\text{-}PWA^S$ follows from the language example of idempotent birooted trees that cannot, by a simple pumping argument, be recognized by an automaton without pebble but that can easily be recognized with a single pebble (see below).

The first row of (vertical) equalities follows from Lemma 26, proven below. Over birooted trees, that is, in the case $G = FG(A)$, these equalities imply the separation result $0\text{-}REG \subset 1\text{-}REG$. Indeed, over birooted trees the language of idempotent trees is recognizable by a one-pebble automaton while a simple argument shows that is cannot be recognized without pebble.

The second row of (vertical) inclusions follows from the known fact [20] (see also [13]) that, for all birooted graphs $x, y \in BSG(G)$, there is a root-preserving graph morphism $f : y \to x$ if and only if $x \leq y$ in the natural order defined by $x \leq y$ when $x = e \cdot y$ for some idempotent element e.

It follows that, we have $X^{\downarrow} = E(BSG(G)) \cdot X$ for all language $X \subseteq BSG(G)$, and the language $(BSG(G))^E$ of all idempotent elements of $BSG(G)$ belongs to $1\text{-}REG$ as shown by the one-state automaton $\mathcal{A} = \langle \{p, q\}, \{p\}, \{q\}, \delta, \Delta \rangle$ with $\delta(z) = \{(p, p)\}$ for every $z \in A + \bar{A}$ and $\Delta((r, s)) = \{(p, q)\}$ when $r = s = p$ and $\Delta((r, s)) = \emptyset$ otherwise. The fact these inclusions are strict follows, for language of birooted trees, from the fact that the language $\{\theta_G(a)\}$ is not closed under morphisms since it is not closed under natural order.

The last row of (vertical) equalities follows from the first row of vertical equalities and Theorem 16. □

Lemma 26. *For every ≥ 0, we have $k\text{-}PWA^S = k\text{-}REG$.*

Proof (sketch of). Direct inclusion ($\subseteq$). Let $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$ be a finite-state walking automaton. For every pair of states $p, q \in Q$, let $L^k_S(p, q) \subseteq BSG(G)$ be the class of languages strictly k-recognized by the automaton $\mathcal{A}$ from an initial configuration of the form (p, p, x) to a terminal configuration of the form (p, q, y) for some vertices x, y. Let $E^k_S(p, q)$ be restriction of that language to the case $x = y$, or, equivalently, $E^k_S(p, q) = (L^k_S(p, q))^E$. Then, much like in the proof of Kleene's theorem for regular languages of strings, by mimicking walking automata transition rules, we can define a system of equations relating

the languages $L_S^k(p,q)$ and $E_S^k(p,q)$ which resolution yields the expected regular expressions.

Reverse inclusion ($\supseteq$). This can be proved by induction on the syntactic complexity of regular expressions. More precisely, we first prove that the singleton languages $\{\theta_G(1)\}$ and $\{\theta_G(z)\}$ for every $z \in A + \bar{A}$, are strictly 0-recognizable by finite automata. Then, it suffices to show that the class of languages strictly k-recognized by finite walking automata is closed under sum, product and star, and that, if X is strictly k-recognized by a finite walking automaton, then X^E is $(k+1)$-recognized by a finite walking automaton.

It must be noticed that the existence of second order transitions makes these constructions slightly more complex than in the case of string languages. In particular, building an automaton $\mathcal{A}^*$ such that $L_S^k(\mathcal{A}^*) = (L_S^k(\mathcal{A}))^*$ is done from $k+1$ copies of the automaton $\mathcal{A}$. Indeed, this allows to count in any state the number of pebbles that have been dropped and to ensure, between two runs of the automaton $\mathcal{A}$ simulated in the automaton $\mathcal{A}^*$, that all pebbles have been lifted. □

5 Conclusion

We have defined walking automata on graphs. By allowing automata to start and stop in arbitrary graph vertices, we have defined the language recognized by a walking automaton in terms of birooted graphs that form inverse semigroups.

Although we do not require walking automata to perform complete traversal of their input structures, thanks to the preorder relation induced by root preserving graph homomorphisms, we eventually provide a correspondence between our notion of recognizability and the more classical one.

In the particular case of Cayley's graphs of groups, we obtain a rather rich array of classes of recognizable languages of birooted graphs, and a notion of k-regular expressions that characterizes the number of allowed pebbles in accepting runs (Theorem 25). How these induced hierarchies of languages of trees or graphs may be related is left as an intriguing open problem.

We conjecture that the hierarchy induced by the number of pebbles is strict for languages of birooted trees. The strictness of the hierarchies for languages of birooted graphs induced by other groups than the free group is also an open problem. Ideally, the algebraic framework proposed here may provide simpler arguments than in [3] for solving these questions.

It has already been observed that an adequate algebraic theory for inverse monoid morphisms can be developed by means of certain kind of premorphisms instead of morphisms [9,10,12,14]. As a matter of fact, transition monoids of walking automata induce a different type of premorphisms that could also be investigated as new language recognizers.

References

1. M. Bojańczyk. Tree-walking automata. In *2nd Int. Conf. on Language and Automata Theory and Applications (LATA)*, volume 5196 of *LNCS*. Springer, 2008.

2. M. Bojańczyk and T. Colcombet. Tree-walking automata do not recognize all regular languages. In *STOC*. ACM, 2005.
3. M. Bojańczyk, M. Samuelides, T. Schwentick, and L. Segoufin. Expressive power of pebble automata. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, 2006.
4. B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic, a language theoretic approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.
5. A. Dicky and D. Janin. Two-way automata and regular languages of overlapping tiles. *Fundamenta Informaticae*, 142:1–33, 2015.
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, 1999.
8. J. Engelfriet and H.J. Hoogeboom. Nested pebbles and transitive closure. In *Symp. on Theor. Aspects of Computer Science (STACS)*, 2006.
9. D. Janin. Quasi-recognizable vs MSO definable languages of one-dimensional overlapping tiles. In *Mathematical Found. of Comp. Science (MFCS)*, volume 7464 of *LNCS*, pages 516–528, 2012.
10. D. Janin. Algebras, automata and logic for languages of labeled birooted trees. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 7966 of *LNCS*, pages 318–329. Springer, 2013.
11. D. Janin. On languages of one-dimensional overlapping tiles. In *Int. Conf. on Current Trends in Theo. and Prac. of Comp. Science (SOFSEM)*, volume 7741 of *LNCS*, pages 244–256. Springer, 2013.
12. D. Janin. Towards a higher dimensional string theory for the modeling of computerized systems. In *Int. Conf. on Current Trends in Theo. and Prac. of Comp. Science (SOFSEM)*, volume 8327 of *LNCS*, pages 7–20. Springer, 2014.
13. D. Janin. Inverse monoids of higher-dimensional strings. In *Int. Col. on Theor. Aspects of Comp. (ICTAC)*, volume 9399 of *LNCS*, 2015.
14. D. Janin. On labeled birooted trees languages: Algebras, automata and logic. *Information and Computation*, 243:222–248, 2015.
15. S. W. Margolis and J. C. Meakin. E-unitary inverse monoids and the Cayley graph of a group presentation. *J. Pure and Appl. Algebra*, 58:46–76, 1989.
16. J. Meakin. Groups and semigroups: connections and contrasts. In *Groups St Andrews 2005, Volume 2*, London Mathematical Society, Lecture Note Series 340. Cambridge University Press, 2007.
17. R. Milner. *Communication and concurrency*. Prentice-Hall, 1989.
18. J.-P. Pécuchet. Automates boustrophedon, semi-groupe de Birget et monoide inversif libre. *ITA*, 19(1):71–100, 1985.
19. P. V. Silva. On free inverse monoid languages. *ITA*, 30(4):349–378, 1996.
20. J.B. Stephen. Presentations of inverse monoids. *Journal of Pure and Applied Algebra*, 63:81–112, 1990.