



**HAL**  
open science

## Walking automata in free inverse monoids

David Janin

► **To cite this version:**

David Janin. Walking automata in free inverse monoids. [Research Report] LaBRI - Laboratoire Bordelais de Recherche en Informatique. 2015. hal-00738793v3

**HAL Id: hal-00738793**

**<https://hal.science/hal-00738793v3>**

Submitted on 5 Aug 2015 (v3), last revised 3 Oct 2015 (v4)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 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, as illustrated, for language of trees, by the pebble hierarchy problem. In this paper, we study a generic notion of walking automata over graphs which semantics, thanks to the notion of birooted graphs that are accepted by these automata, 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 into 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 that can be dropped and lifted on vertices, 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 for 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 raised by 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 may use induces 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 for the study of word automata are not easily applicable to tree walking automata. Despite the numerous mentioned results, little is known about the underlying mathematical framework 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 and study a generic notion of automata walking on edge-labeled graphs that acts as some kind of an observer of the (edge labeled) input graph it walks on, much like observational semantics have been defined in concurrency theory by Hennessy and Milner [14].

Unlike in most classical definitions, we do not require walking automata to start and end in the same vertex, neither we require the complete input structure to be traversed. 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 is the same way there are various observational semantics in [14], with or without observable failures.

The resulting languages recognized by walking automata are thus languages of birooted graphs, that is, graphs extended with a input root: the vertex where the run starts, and an output root: the vertex where the run ends. These languages are shown to closed under roots 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 which yields 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 still be uniformly encoded in by the weaker semantics (with unobservable reading failures).

As particular cases, walking automata in Cayley's graphs of presented groups are considered in Section 4. The recognized languages are subsets of monoids known as the free  $E$ -unitary inverse monoids [12,13]. Then, based on the underlying monoid structure, Kleene's like regular expressions are uniformly 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, in the absence of adequate references covering all, we provide our own presentation and proofs.

**Graphs and morphisms.** Let  $A = \{a, b, c, \dots\}$  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 every  $a \in A$ . 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  $f : V_1 \rightarrow V_2$  such that  $f(E_1(a)) \subseteq E_2(a)$ , for every  $a \in A$ . Such a morphism is denoted by  $f : G_1 \rightarrow G_2$ .

**Walking paths.** Let  $\bar{A} = \{\bar{a}, \bar{b}, \bar{c}, \dots\}$  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 allowing simpler statement, the inverse relation  $E^{-1}(a) = \{(x, y) \in V \times V : (y, x) \in E(a)\}$  is denoted by  $E(\bar{a})$  for every  $a \in A$ . The vertex  $x$  is the *source* of such a path. It is denoted by  $\mathbf{sr}(\pi)$ . The vertex  $y$  is the *target* of such a path. It is denoted by  $\mathbf{tg}(\pi)$ .

In such a labeling of 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*  $\pi^{-1}$  of the path  $\pi$  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  $\pi^{-1}$  is indeed a path in the graph  $G$  from  $x_n$  to  $x_0$ . The set of paths over the graph  $G$  is denoted by  $Path(G)$ .

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  $\pi$  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  $\pi$  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 in graph  $G$ . The *path induced subgraph*  $G|\pi$  of graph  $G$  induced by path  $\pi$  is defined by  $G|\pi = \langle V|\pi, E|\pi \rangle$  with the set of vertices  $V|\pi = \{x_0, x_1, \dots, x_n\}$  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 either  $xy$  or  $y\bar{a}x$  occurs as a subsequence in the path  $\pi$ . Then, the following lemma is immediate.

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

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

**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  $\langle \mathbf{sr}(\pi_1), G|\pi_1 \cup G|\pi_2, \mathbf{tg}(\pi_2) \rangle$  (with union defined over subsets of vertices and edges) when  $\mathbf{tg}(\pi_1) = \mathbf{sr}(\pi_2)$  and defined to be an additional birooted graph 0 otherwise defines yields to an inverse semi-group: 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 idempotent elements correspond to cyclic path.

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 defined here have no vertex label. Still, 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 relevance of this notion is justified by the following lemma which proof is immediate.

**Lemma 5.** *Let  $G = \langle V, E \rangle$  be a graph on the edge alphabet  $A$ . Let  $g : V \rightarrow \mathcal{P}(B)$  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 graph one-to-one and onto morphism from  $G$  into  $\varphi(\langle G, g \rangle)$ . Moreover,  $\varphi$  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  $\lambda_{V'} : V' \rightarrow \mathcal{P}(A \cup B)$  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 label 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 graphs 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, in the simpler case, 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  $\delta$  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  $\Delta$  applied to the pair of states resulting from the start state and the stop state of the subrun.

**Definition 7 (Automata configuration and run).** 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  $Q \times Q \times V$ . 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:  $z \in A \cup \bar{A}$ ,  $p_1 = p_2$ ,  $(q_1, q_2) \in \delta(z)$  and  $(x, y) \in E(z)$ ,
- (2) pebble drop:  $z = 1$ ,  $y = x$ ,  $\Gamma_2 = \Gamma_1 \cdot (p_1, q_1, x)$  and  $p_2 = q_2$ ,
- (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

$$\rho = \Gamma_0.(p_0, q_0, x_0) \vdash_{z_1} \Gamma_1.(p_1, q_1, x_1) \cdots \Gamma_{n-1}.(p_{n-1}, q_{n-1}, x_{n-1}) \vdash_{z_n} \Gamma_n.(p_n, q_n, x_n)$$

with  $x_0 = x$  and  $x_n = y$ , also denoted by  $\rho = \Gamma_0 \cdot (p_0, q_0, x_0) \vdash_u^* \Gamma_n \cdot (p_n, q_n, y_n)$  with  $u = z_1 z_2 \cdots z_n$ . The *path*  $\pi(\rho)$  induced by run  $\rho$  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  $\rho$  is an *accepting run* from  $x$  to  $y$  when  $\Gamma_0$  is the empty stack with  $p_0 = q_0 \in I$  (the first configuration is initial), and  $\Gamma_n$  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  $\rho$  is *k-accepting run* when it is accepting and  $|I_i| \leq k$  for every  $0 \leq i \leq n$ , where  $|I_i|$  is the length of the sequence  $\Gamma_i$ . For notational purpose, an accepting run with no bound on the number of allowed pebble is also called an  $\infty$ -*accepting run*.

**Remark 8.** 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  $\mathcal{A}$*

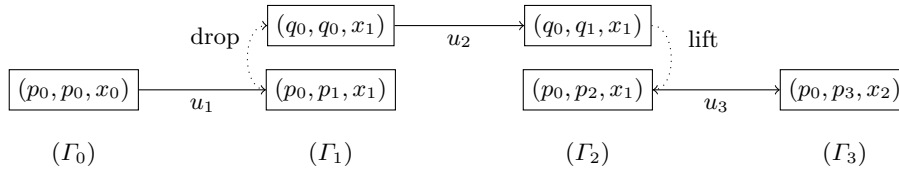
walked to the current vertex  $x$  reaching state  $q$ . The additional stack  $\Gamma$ , possibly empty, contains the configurations of formerly interrupted runs.

The rule describing an edge traversal step should be clear. The pebble handling rules require some more explanation.

Dropping a pebble in a transition step amounts to *interrupt* the current run, pushing its configuration  $(p_1, q_1, x)$  on the stack, and to *start a subrun*  $(p_2, p_2, x)$  in the same vertex.

On the contrary, lifting a pebble amounts to *terminate a subrun*  $(p_1, q_1, x)$ , and to *resume* the previous current run, updating the previous current state according to the set of possible (second order) transitions  $\Delta((p_1, q_1))$ .

**Example 9.** An example of such a run is depicted Figure 1 where configuration stacks are depicted vertically. In the case no pebble are dropped nor lifted while



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

reading  $u_1, u_2$  and  $u_3$ , the transition constraints are given by  $(p_0, p_1) \in \delta(u_1)$ ,  $(q_0, q_1) \in \delta(u_2)$ ,  $(p'_1, p_2) \in \delta(u_3)$  for the first order transitions, with an obvious extension of  $\delta$  to  $(A + \bar{A})^*$ , and  $(p_1, p'_1) \in \Delta((q_0, q_1))$  for the second order transitions.

**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  $L_{\mathcal{G}}^{\infty}(\mathcal{A})$  (resp.  $L_{\mathcal{G}}^k(\mathcal{A})$ ) of birooted graphs  $(G, x, y)$  with  $G \in \mathcal{G}$  and  $x, y$  two vertices of  $G$ , for which 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 studies of two-way automata on strings with his notion of bisections [15] (see also [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  $f : G_1 \rightarrow G_2$ . 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  $(G_2, f(x), f(y)) \in L^k(\mathcal{A})$ .

For every environment  $\Gamma$  (run  $\rho$ ) of the automaton  $\mathcal{A}$  on  $G_1$ , let  $f(\Gamma_1)$  (resp.  $f(\rho)$ ) be the environment (resp. the run) of the automaton  $\mathcal{A}$  on  $G_2$  obtained from  $\Gamma$  (from  $\rho$ ) by replacing all vertices  $x \in V_1$  by their images  $f(x) \in V_2$ . Then, a simple induction shows that for every run  $\rho = \Gamma_1 \vdash_u^* \Gamma_2$  of  $\mathcal{A}$  from  $x$  to  $y$  in graph  $G_1$ ,  $f(\rho) = (\Gamma_1) \vdash_u^* f(\Gamma_2)$  is a well defined run of  $\mathcal{A}$  from  $f(x)$  to  $f(y)$  in graph  $G_2$ . The rest of the proof is the routine. 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  $\rho$*  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  $\langle x, G, y \rangle \in L^k(\mathcal{A})$  via an accepting run  $\rho$ . Let  $G_\rho$  be the graph induced by the run  $\rho$ . Then  $\langle x, G_\rho, y \rangle \in L^k(\mathcal{A})$ .*

Let  $\rho$  be a run of  $\mathcal{A}$  from  $x$  to  $y$  in  $G$ . Then a simple induction on the length of the run  $\rho$  shows that  $\rho$  is also a run of  $\mathcal{A}$  from  $x$  to  $y$  in  $G|\pi(\rho)$  which concludes the proof. The more classical notion of accepting runs defined by complete traversals of the input structure is then recovered by the following definition.

**Definition 15 (Strict recognizability).** A birooted graph  $\langle x, G, y \rangle$  is *strictly recognized* (resp. *strictly  $k$ -recognized*) by an automaton  $\mathcal{A}$  when there is an accepting (resp.  $k$ -accepting) run  $\rho$  of  $\mathcal{A}$  over  $G$  from  $x$  to  $y$  such that  $\langle x, G_\rho, y \rangle$  and  $\langle x, G, y \rangle$  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  $L_S^k(\mathcal{A})$  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  $L_S^k(\mathcal{A})$ , that is,  $\langle G, x, y \rangle \in L^k(\mathcal{A})$  if, and only if, there exists  $\langle G', x', y' \rangle \in L_S^k(\mathcal{A})$  and a graph morphism  $f : G' \rightarrow G$  such that  $f(x') = x$  and  $f(y') = y$ .*

In particular, in the case the birooted structures under study are unrelated 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 exploited 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 an inverse monoid that is free in some sense (see [12] and also [13] for a general presentation). Various language composition operators can then be defined over subsets of this inverse monoid and related with languages recognizable by walking automata.

**Definition 17 (Cayley’s graph of a presented group).** Let  $G$  be a group generated by  $A \subseteq G$  and let  $\varphi : (A + \bar{A})^* \rightarrow G$  be the corresponding inverse preserving monoid morphism<sup>1</sup>. Then, the *Cayley graph* of the presented group  $G$  is defined to be the graph  $C_G = \langle V, E \rangle$  with vertices defined by  $V = G$  and, for every  $a \in A$ , edges defined by  $E(a) = \{(x, y) \in V \times V : x \cdot \varphi(a) = y\}$ .

For our 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  $\varphi$  is inverse preserving and  $G$  is a group, we indeed have  $E(\bar{a}) = E(a)^{-1}$  which conforms to the extension of the edge relation we have used so far.

**Remark 18.** Clearly, the Cayley’s graph  $C_G$  of the group  $G$  is a bideterministic graph as well as its birooted subgraphs. Depending on the chosen group, various examples of interest can be defined (see [13]).

Taking the free group  $FG(A)$  we have *birooted trees*. Taking the group defined from  $A = \{a, b, c, d\}$  with  $cc = 1$ ,  $dd = 1$  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 is already studied in [9].

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

In other words, such a 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 presented group with morphism  $\varphi$ , and let  $C_G$  be its Cayley’s 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  $V = \varphi(\text{Pref}(u))$  where  $\text{Pref}(u) = \{v \in (A + \bar{A})^* : \exists w \in (A + \bar{A})^*, u = vw\}$  is the set of word prefix of  $u$ , and set of  $a$ -labeled 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  $\pi \in \text{Path}(C_G)$  be a path in  $C_G$ . Let  $\lambda(\pi) \in (A + \bar{A})^*$  be its projection over  $(A + \bar{A})^*$  obtained from  $\pi$  by deleting all vertices. Then the birooted subgraph  $\theta_{C_G}(\pi)$  induced by the path  $\pi$  is graph  $C_G$  is isomorphic to the birooted graph  $\langle 1, C_G|_{\lambda(\pi)}, \varphi(u) \rangle$ .*

<sup>1</sup> The group  $G$  is presented by the morphism  $\varphi$ .

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  $B = \langle 1, V, E, x \rangle$  such that  $V \subseteq G$  is a finite subset of  $G$  such that  $1, u \in G$ ,  $E(a) \subseteq \{(x, y) \in V \times V : x \cdot \varphi(a) = y\}$  for every  $a \in A$ , such that, moreover, 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  $B_1 = \langle 1, V_1, E_1, x_1 \rangle$  and  $B_2 = \langle 1, V_2, E_2, x_2 \rangle$  is defined by

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

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 [12]).** *Then  $BSG(G)$  is an inverse monoid. Moreover, the mapping  $\theta_G : (A + \bar{A})^* \rightarrow BSG(G)$  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 [12], it is proved that  $BSG(G)$  is the freest inverse monoid generated by  $A$  which group image is the group  $G$ . This result is much stronger than the above statement.

A subset  $X \subseteq BSG(C_G)$  of the monoid  $BSG(G)$  is from now on called a  $G$ -*language*. Following our previous definitions, given  $0 \leq k \leq \infty$ , the language  $X$  is  $k$ -recognized (resp. strictly  $k$ -recognized) by the walking automaton  $\mathcal{A}$  when  $X = L^k(\mathcal{A}) \cap BSG(C_G)$  (resp.  $X = L_S^k(\mathcal{A}) \cap BSG(C_G)$ ). Let then  $k$ -PWA (resp.  $k$ -PWA<sup>S</sup>) 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:  $X_1 \cdot X_2 = \{x_1 \cdot x_2 \in BSG(G) : x_1 \in X, x_2 \in X\}$ ,
- (3) star:  $X^* = \bigcup X^n$ ,
- (4) inverse:  $X^{-1} = \{x^{-1} \in BSG(G) : x \in X\}$ ,
- (5) idempotent restriction:  $X^E = \{x \in X : xx = x\}$ ,

for every language  $X, X_1, X_2 \subseteq BSG(G)$ .

A  $k$ -*regular expression* is defined to be any finite expression built over 1, single letters of  $A + \bar{A}$ , 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  $k\text{-REG}$ . 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 a finite Kleene regular expressions equals  $0\text{-REG}$ . Last, for every class of language  $X$ , let also  $X^\downarrow$  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  $\varphi : M \rightarrow N$ , we certainly have  $\varphi(X^E) \subseteq \varphi(X)^E$  for every  $X \subseteq M$ , but the reverse inclusion may be false as illustrated by the expression  $ab\bar{a}\bar{b}$  that 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 which  $ab = ba$ .

**Theorem 25 (Hierarchy).** *For all presented group  $G$  generated by  $A$ , the following equalities and inequalities holds. The marked strict inequalities are known to hold in the free inverse monoid, that is, when  $G = FG(A)$ : the free group generated by  $A$ .*

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

*Proof.* Each horizontal inclusion follows from the definition. The separation result  $REC \subset 0\text{-REG}$  is known over languages of birooted trees [16]. 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 that is proved below. Over birooted trees, that is, in the case  $G = FG(A)$ , these equalities implies the separation result  $0\text{-REG} \subset 1\text{-REG}$ . Indeed, over birooted trees the language of idempotent tree is recognizable by a one pebble automaton while, a simple argument shows that it cannot be recognized without pebble.

The second row of (vertical) inclusion follows from the known fact [17] (see also [11]) that, for every birooted graphs  $x, y \in BSG(G)$ , there is a root preserving graph morphism  $f : y \rightarrow x$  if and only if  $x \leq y$  in the natural order, that is, if and only if  $x = e \cdot y$  for some idempotent element  $e$ .

It follows that, for every language  $X \subseteq BSG(G)$ , we have  $X^\downarrow = E(BSG(G)) \cdot X$  and the language  $E(BSG(G))$  of all idempotent elements of  $BSG(G)$  belongs

to 1-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 easily 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.  $\square$

**Lemma 26.** *For every  $\geq 0$ , 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 state  $p, q \in Q$ , let  $L_S^k(p, q) \subseteq BSG(G)$  be the class of language 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  $E_S^k(p, q)$  be restriction of that language to the case  $x = y$ , or, equivalently,  $E_S^k(p, q) = (L_S^k(p, q))^E$ . 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 easily 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 classical 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.  $\square$

We first prove the direct inclusion ( $\subseteq$ ). Let  $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$  be a finite state walking automaton. For every pair of state  $p, q \in Q$ , let  $L_S^k(p, q) \subseteq BSG(G)$  be the class of language 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  $E_S^k(p, q)$  be restriction of that language to the case  $x = y$ , or, equivalently,  $E_S^k(p, q) = (L_S^k(p, q))^E$ .

In the case  $k = 0$ , for every  $p, q \in Q$ , we have,

$$L_S^0(p, q) = \delta_{p, q} + \sum_{\substack{z \in A + \bar{A} \\ (p, r) \in \delta(z)}} \theta_G(z) \cdot L_S^0(r, q)$$

with  $\delta_{p,q} = \theta_G(1)$  if  $p = q$  and  $\emptyset$  otherwise. By applying standard argument, this proves that  $L_S^0(p, q)$  is 0-REG for every  $p, q \in Q$  hence  $L_S^0(\mathcal{A})$  as well.

Assume that the claim is true for every  $k' < k + 1$ . From the definition of runs, we easily check that

$$\begin{aligned} L_S^{k+1}(p, q) &= \delta_{p,q} + \sum_{\substack{z \in A + \bar{A} \\ (p, r) \in \delta(z)}} \theta_G(z) \cdot L_S^{k+1}(r, q) \\ &+ \sum_{\substack{(p', q') \in Q \times Q \\ (p, r) \in \Delta(p', q')}} E_S^k(p', q') \cdot L_S^{k+1}(r, q) \end{aligned}$$

since we just enumerate the two cases in which the run may start: either it reads an edge, or it drops a pebble. Then, by induction hypothesis, for every  $p', q' \in Q$ , the language  $E_S^k(p', q')$  is  $(k+1)$ -regular hence, by application standard reduction techniques, for every  $p, q \in Q$ , the language  $L_S^{k+1}(p, q)$  is also  $(k+1)$ -regular and thus so is  $L_S^{k+1}(\mathcal{A})$ .

We prove now the reverse inclusion case ( $\supseteq$ ). This is proved by induction on the syntactic complexity of  $k$ -regular expressions.

For the basic cases, let  $\mathcal{A}_1$  the automaton a single state, both initial and terminal, and with empty transition functions. Clearly,  $L_S^0(\mathcal{A}_1) = \{\theta_G(1)\}$ .

Similarly for every  $a \in A$ . Let  $\mathcal{A}_a = \langle D, I, T, \delta, \Delta \rangle$  be the automaton defined by  $Q = \{1, \varphi(a)\}$ ,  $I = \{1\}$ ,  $T = \{\varphi(a)\}$ ,  $\delta(a) = \{(1, \varphi(a))\}$ ,  $\delta(a^{-1}) = \{(\varphi(a), 1)\}$ , and  $\delta(z') = \emptyset$  for every  $z' \notin \{a, a^{-1}\}$ , and  $\Delta((t, s)) = \emptyset$  for every  $r, s \in Q$ . Clearly, we have  $L_S^0(\mathcal{A}_a) = \{\theta_G(a)\}$ .

Then, it suffices to show that the class of languages strictly  $k$ -recognized by finite walking automata is closed under sum, product and star, that, if  $X$  is strictly  $k$ -recognized by a finite walking automaton, then  $X^E$  is  $(k+1)$ -recognized by a finite walking automaton. Let then  $X_1, X_2 \subseteq BSG(G)$  be two languages strictly  $k$ -recognizable by the automaton  $\mathcal{A}_1 = \langle Q_1, I_1, T_1, \delta_1, \Delta_1 \rangle$  and the automaton  $\mathcal{A}_2 = \langle Q_2, I_2, T_2, \delta_2, \Delta_2 \rangle$ .

Clearly, the sum  $X_1 + X_2$  is strictly  $k$ -recognizable by the disjoint sum automaton  $\mathcal{A}_1 \uplus \mathcal{A}_2$ .

For the product, possibly splitting both languages into a disjoint sum with  $\{\theta_G(1)\}$  and applying classical identities, we may assume, without loss of generality, that none of these languages contains  $\theta_G(1)$  henceforth  $I_1 \cap T_1 = \emptyset$  and  $I_2 \cap T_2 = \emptyset$ . Let then  $\mathcal{A}_1 \cdot \mathcal{A}_2 = \langle Q, I, F, \delta, \Delta \rangle$  be the automaton defined by  $Q = Q_1 \uplus Q_2$ ,  $I = I_1$ ,  $T = T_2$ , for every  $z \in A + \bar{A}$ ,

$$\delta(z) = \delta_1(z) \cup \delta_2(z) \cup \{(p, q) \in Q \times Q : \exists q' \in T_1, (p, q') \in \delta_1(z), q \in I_2\}$$

and, for every  $r, s \in Q$ , the second order transitions  $\Delta((r, s))$  defined by

$$\Delta((r, s)) = \Delta_1((r, s)) \cup \{(p, q) : q \in I_2, \exists q' \in T_1, (p, q') \in \Delta_1((r, s))\}$$

when  $r, s \in Q_1$ ,  $\Delta((r, s)) = \Delta_2((r, s))$  when  $r, s \in Q_2$ , and  $\Delta((r, s)) = \emptyset$  otherwise. Then, we can check by induction on the length of runs, that  $L_S^k(\mathcal{A} \cdot \mathcal{A}_2) = X_1 \cdot X_2$ . The sensitive point, the direct inclusion, is handled by the fact that no runs can go back and forth between the copy of  $\mathcal{A}_1$  and the copy of  $\mathcal{A}_2$ , and, in an accepting run, moving from a state of  $\mathcal{A}_1$  to a state of  $\mathcal{A}_2$  is necessarily done with all pebble lifted.

The case of the star is slightly more complex because we must make sure that, at each iteration, no pebble is left on the input.

Still, this can be done by first normalizing the automaton  $\mathcal{A}$  (with disjoint set of initial and terminal states) into  $\mathcal{A}' = \langle Q', I', T', \delta', \Delta' \rangle$  built out of  $k+1$  disjoint copy of the automaton  $\mathcal{A}$  by taking  $Q' = Q \times \{0, 1, \dots, k\}$ ,  $I' = I \times \{0\}$ ,  $T' = T \times \{0\}$ , for every  $z \in A + \bar{A}$ ,  $\delta'(z) = \{(p, i), (q, i)\} \in Q' \times Q' : (p, q) \in \delta(z), 0 \leq i \leq k\}$ , and for every  $r, s \in Q$  and  $0 \leq i_1, i_2 \leq k$ , taking

$$\Delta((r, i_1), (s, i_2)) = \{(p, i), (q, i)\} \in Q' \times Q' : (p, q) \in \delta((r, s))\}$$

in the case  $i_1 = i_2 = i + 1$  for  $i \geq 0$ , and  $\Delta((r, i_1), (s, i_2)) = \emptyset$  otherwise.

Then, given the automaton  $\mathcal{A}'$  as above, we build an automaton  $\mathcal{A}^+$ , by adding a new initial state  $q_0$ , and by duplicating any first order transition from  $(q_1, 0)$  to  $(q_2, 0)$  to a transition from  $q_0$  to  $(q_2, 0)$  in the case  $q_1 \in I$ , and a transition from  $(q_1, 0)$  to  $q_0$  in the case  $q_2 \in T$ . Then, it is routine to check that  $\mathcal{A}^+$  strictly  $k$ -recognized  $X^+$ . Adding  $\theta_G(1)$  poses no difficulty.

The case of the idempotent projection is done as follows. Let  $\mathcal{A} = \langle Q, I, F, \delta, \Delta \rangle$  that strictly  $k$ -recognized  $X \subseteq BSG(G)$  with  $\theta_G(1) \notin X$ . Then we define  $\mathcal{A}^E = \langle Q \uplus \{q_i, q_f\}, \{q_i\}, \{q_f\}, \delta', \Delta' \rangle$  by defining for every  $z \in A + \bar{A}$ , the first order transitions

$$\delta'(z) = \delta(z) \cup \{(q_i, q) : \exists p \in I, (p, q) \in \delta(z)\} \cup \{(p, q_f) : \exists q \in T, (p, q) \in \delta(z)\}$$

and, for every  $r, s \in Q \uplus \{q_i, q_f\}$  we put

$$\Delta'((r, s)) = \{(q_i, q_f)\} \cup \Delta((r, s))$$

in the case  $r \in I$  and  $s \in T$ ,

$$\Delta'((r, s)) = \Delta((r, s))$$

in the case  $r, s \in Q$  with  $r \notin I$  or  $s \notin T$ , and  $\Delta'(r, s) = \emptyset$  otherwise.

An accepting run with  $k+1$ -pebble of the automaton  $\mathcal{A}^E$ , necessarily starts in the state  $q_i$  and ends in the state  $q_f$ . By construction, from the state initial  $q_i$ , a pebble must be dropped. Then an accepting run with at most  $k$ -pebble is ran in  $\mathcal{A}$  and that initial pebble can be lifted, reaching the terminal state  $q_f$ . This proves that  $L_S^{k+1}(\mathcal{A}^E) \subseteq X^E$ . Since the converse inclusion is immediate from the definition, this concludes the idempotent projection case and the proof of the lemma.

## 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, which products yields inverse semigroups.

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

In the particular case of Cayley's graphs of group, this leads to 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. How the induced hierarchies of languages of trees or graphs may be related is left as an intriguing open problem.

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]. As a matter of fact, transition monoids of walking automata induce a different type of premorphisms that could also be investigated as 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. Research report RR-1463-12, LaBRI, Université de Bordeaux, 04 2013.
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. On languages of labeled birooted trees: Algebras, automata and logic. *Information and Computation*, 2014.
10. D. Janin. Towards a higher dimensional string theory for the modeling of computerized systems. In V. Geffert et al., editor, *Int. Conf. on Current Trends in*

- Theo. and Prac. of Comp. Science (SOFSEM)*, volume 8327 of *LNCS*, pages 7–20. Springer, 01 2014.
11. D. Janin. Inverse monoids of higher-dimensional strings. In *Int. Col. on Theor. Aspects of Comp. (ICTAC)*, 2015.
  12. Stuart W. Margolis and John C. Meakin. E-unitary inverse monoids and the cayley graph of a group presentation\*. *J. Pure and Appl. Algebra*, 58:46–76, 1989.
  13. 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.
  14. R. Milner. *Communication and concurrency*. Prentice-Hall, 1989.
  15. J.-P. Pécuchet. Automates boustrophedon, semi-groupe de Birget et monoïde inversif libre. *ITA*, 19(1):71–100, 1985.
  16. P. V. Silva. On free inverse monoid languages. *ITA*, 30(4):349–378, 1996.
  17. J.B. Stephen. Presentations of inverse monoids. *Journal of Pure and Applied Algebra*, 63:81–112, 1990.