HAL open science

Inverse monoids of higher-dimensional strings

David Janin

To cite this version:

David Janin. Inverse monoids of higher-dimensional strings. 12th International Colloquium on Theoretical Aspects of Computing (ICTAC 2015), 2015, Cali, Colombia. hal-01165724v2

HAL Id: hal-01165724

https://hal.science/hal-01165724v2

Submitted on 5 Aug 2015

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.

Inverse monoids of higher-dimensional strings

David Janin

LaBRI CNRS UMR 5800,
Bordeaux INP,
Université de Bordeaux,
INRIA Bordeaux Sud-Ouest
F-33405 Talence, FRANCE
janin@labri.fr

Abstract. Halfway between graph transformation theory and inverse semigroup theory, we define higher dimensional strings as bi-deterministic graphs with distinguished sets of input roots and output roots. We show that these generalized strings can be equipped with an associative product so that the resulting algebraic structure is an inverse semigroup. Its natural order is shown to capture existence of root preserving graph morphism. A simple set of generators is characterized. As a subsemigroup example, we show how all finite grids are finitely generated. Finally, simple additional restrictions on products lead to the definition of subclasses with decidable Monadic Second Order (MSO) language theory.

1 Introduction

A never-ending challenge faced by computer science is to provide modeling concepts and tools that, on the one hand, allows for representing data and computations in a more and more abstract and richly structured way, but, on the other hand, remains simple enough to be taught to and used by application designers and software engineers [33].

A possible approach to this goal consists in generalizing to graphs the techniques that have already been developed for strings or trees such as the notion of recognizable languages and the associated notion of recognizers. In these directions, an enormous amount of techniques and works has been developed ranging from Lewis' graph composition techniques [27] and Courcelle's developments of recognizability to graph languages [8] (see also [9]) up to more recent advances based on category theoretical development (see [13, 6] to name but a few).

Despite numerous achievements in theoretical computer science, there is still room for polishing these techniques towards applications to computer engineering. The ideal balance to achieve between usage simplicity and mathematical coherence is a long-term goal [33]. While the underlying frameworks (the back end) of application tools to be designed can (and probably should) be based on robust mathematics, the interface (the front end) of these tools must be kept simple enough to be taught ad used.

Keeping in mind that strings, free monoids and related automata techniques are among the simplest and the most robust available models and are already

and successfully put in practice in system modeling methods like *event B* [2], we develop in this paper a notion of generalized strings, called *higher dimensional strings*, in such a way that:

1. higher dimensional strings are simple: they are finitely generated from elementary graphs composed via a single and associative product that generalizes string concatenation in free monoids (Theorem 25),
2. the resulting classes of generalized strings include large classes of finite graphs such as, in particular, hypercubes, hence the name higher dimensional (Section 5 for the case of grids),
3. the resulting semigroups are inverse semigroups (Theorems 17 and 19) henceforth mathematically rich enough to provide algebraic characterization of graph-based concepts such as, for instance, existence of graph morphisms characterized by natural order (Theorem 23) or acyclicity defined by a quotient with an adequate ideal (Lemma 33),
4. some well-defined and rich subclasses of these generalized strings still has efficient, expressive and decidable language theory (Theorem 32).

Technically, following the lines already sketched in [18], we use and generalize the concept of birooted graphs (with single input and output roots) defined and used in [31] into the notion of higher dimensional strings (with sets of input and output roots). This provides a better measure of the amount of overlaps that occurs in birooted graphs products can be better measured. Thus we can extend the notion of disjoint product [15, 17] and the applicable partial algebra techniques [5]). This yields to our main decidability result (Theorem 32).

In some sense, our proposal amounts to combining concepts and results arising from the theory of inverse semigroups [26, 29] with graph transformation approaches [27, 13, 6, 9].

Of course, various research developments have already shown that inverse semigroup theory is applicable to computer science, be it for data, computation, language or system modeling. Concerning data modeling, experiments in theoretical physics have already shown that structured data as complex as quasi-crystals can be described by means of some notion of (inverse) tiling semigroup [23–25]. Inverse semigroup theory has also been used to study reversible computations [10, 1]. More recently, various modeling experiments have been conducted in computational music [3, 21]. These last experiments also led to the definition of a Domain Specific (Programing) Language (DSL) which semantics is based on concepts arising from inverse semigroup theory [22, 14].

2 Preliminaries

Let $A = \{a, b, c, \cdots\}$ be a finite alphabet of graph edge labels. Every concept defined in the sequel could be extended to hypergraphs, that is, graphs with edges that possibly relate more than two vertices (see Footnote 1). However, restricting our presentation to standard (binary) graph structures allows us to keep statements (and proofs) simpler.

Relational graphs. A (relational) graph on the (binary symbols) alphabet A, simply called A-*graph* or even *graph* when A is clear from the context, is a pair $G = \langle V, \{E_a\}_{a\in A}\rangle$ with set of vertices V and a-labeled edge relation $E_a \subseteq V \times V$ for every $a \in A$.

Back and forth path labels. Let $\bar{A} = \{\bar{a}, \bar{b}, \bar{c}, \cdots\}$ be a disjoint copy of the alphabet A. A back and forth path label (or simply path label) is a word from the free monoid $(A + \bar{A})^*$ on the alphabet $A + \bar{A}$, with empty word denoted by 1 and the product of two words u and $v \in (A + \bar{A})^*$ denoted by $u \cdot v$ or simply uv. Then, the reverse mapping $w \mapsto \overline{w}$ from $(A + \bar{A})^*$ into itself is inductively defined by $\overline{1} = 1$, $\overline{a \cdot v} = \overline{v} \cdot \bar{a}$ and $\overline{\bar{a} \cdot v} = \overline{v} \cdot a$ for every $a \in A$, $x \in A + \bar{A}$ and $v \in (A+\bar{A})^*$. It is an easy observation that the reverse mapping is an involutive monoid anti-isomorphism, that is, we have $\overline{u \cdot v} = \overline{v} \cdot \overline{u}$ and $\overline{\overline{w}} = w$ for every $u, v, w \in (A + \bar{A})^*$.

Back and forth path actions. For every $X \subseteq V$ and $w \in (A + \bar{A})^*$, the set $X \cdot w \subseteq V$ of vertices reachable from X following w is inductively defined by $X \cdot 1 = X$, $X \cdot aw = \{y \in V : \exists x \in X, (x, y) \in E_a\} \cdot w$ and $X \cdot \bar{a}w = \{y \in V : \exists x \in X, (y, x) \in E_a\} \cdot w$, for every letter $a \in A$ and every string $v \in (A + \bar{A})^*$. In other words, $X \cdot w$ is the set of vertices that can be reached from a vertex in X along a path labeled by w, where a (resp. $\bar{a}$) denotes the forward (resp. backward) traversal of an a-labeled edge in the graph G.

One can check that $X \cdot 1 = X$ and and $X \cdot (u \cdot v) = (X \cdot u) \cdot v$ for every $X \subseteq V$ and every string $u, v \in (A + \bar{A})^*$. Rephrased in semigroup theoretical term, the edge relations of the graph G induce an *action* of the monoid $(A + \bar{A})^*$ on the powerset of the set of vertices of the graph G. It follows that parentheses can be removed without ambiguity in expressions like $(X \cdot u) \cdot v$..

Notation for the singleton case. When X is a singleton $\{x\}$, we may simply write $x \cdot w$ instead of $\{x\} \cdot w$. Similarly, when $x \cdot w$ itself is a singleton we may also treat it just as the element it contains. In other words, we may simply write $x \cdot w = y$ instead of $\{x\} \cdot w = \{y\}$, to denote both the fact that there exists a (back and forth) path from vertex x to vertex y labeled by w and the fact that this path is unique. Similarly, we may say that $x \cdot w$ is undefined (as a vertex) in the case $x \cdot w = \emptyset$ (as a set).

Graph morphism. The usual notion of graph morphism can then be (re)defined via path actions as follows. Let $G = \langle V, \{E_a\}_{a\in A}\rangle$ and $G' = \langle V', \{E'_a\}_{a\in A}\rangle$ be two graphs on the alphabet A. A morphism f from G to G', denoted by $f : G \to G'$, is a mapping $f : V \to V'$ such that we have $f(x \cdot a) \subseteq f(x) \cdot a$ and $f(x \cdot \bar{a}) \subseteq f(x) \cdot \bar{a}$ for every $x \in V$ and every $a \in A$. Then, by induction, we can easily prove that $f(x \cdot w) \subseteq f(x) \cdot w$ for every $x \in V$ and every $w \in (A + \bar{A})^*$.

Graph quotient. Let $G = \langle V, \{E_a\}_{a\in A}\rangle$ be a graph. Let $\simeq$ be an equivalence relation over the set V, that is, a reflexive, symmetric and transitive relation. Let $V/\simeq$ be the set of equivalence classes $\{[x]_\simeq \subseteq V : x \in V\}$ where $[x]_\simeq = \{x' \in V : x \simeq x'\}$. Then, the quotient of the graph G by the equivalence $\simeq$ is defined to be the graph $G/\simeq = \langle V', \{E'_a\}_{a\in A}\rangle$ with set of vertices $V' = V/\simeq_G$

and set of edges $E'_a = \{([x],[y]) \in V' \times V' : ([x] \times [y]) \cap E_a \neq \emptyset\}$. The mapping $\eta_{\simeq} : V \to V/\simeq$ defined by $\eta_{\simeq}(x) = [x]_{\simeq}$ for every $x \in V$ is a surjective morphism called the canonical morphism from the graph G onto the quotient graph $G/\simeq$.

3 Unambiguous graphs and connecting morphisms

We define and study in this section the category of unambiguous graphs and connecting morphisms. Though fairly simple, this study is quite detailed for it constitutes the foundation of the notion of birooted graphs defined in the next section.

Definition 1 (Unambiguous graphs). A graph $G = \langle V, \{E_a\}_{a \in A}\rangle$ is *unambiguous*[1] when, for every vertex $x \in V$, for every path $w \in (A + \bar{A})^*$, there is at most one vertex y such that $x \cdot w = \{y\}$.

Clearly, by simple inductive argument, G is unambiguous as soon as the above condition is satisfied for every one letter path.

Examples. Graphs examples are depicted in Figure 1 with ambiguous graph G_1 and unambiguous graphs I_2 and G_2. In this figure, vertices are named only for illustrative purposes. These vertex names should not be understood as labels. Only edges are labeled in relational graphs.

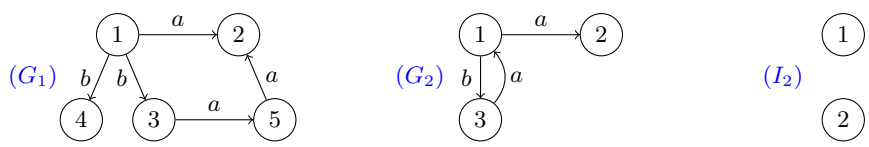

Fig. 1. Ambiguous graph G_1 and unambiguous graph G_2.

One can observe that graph G_1 is ambiguous for two reasons. First, the upper left vertex 1 is the source of two edges labeled by b. Second, the upper right vertex 2 is the target of two edges labeled by a.

Remark. Observe that when a graph G is seen as a graph automaton on the alphabet A, it is unambiguous when it is both deterministic and co-deterministic. In the connected case, these unambiguous graphs are the Schützenberger graphs studied and used in [31].

Definition 2 (Connecting morphisms). Let $f : G \to G'$ be a graph morphism between two graphs $G = \langle V, \{E_a\}_{a \in A}\rangle$ and let $G' = \langle V', \{E'_a\}_{a \in A}\rangle$. The morphism f is a *connecting morphism* when for every $x' \in V'$ there exist $x \in V$ and $w \in (A + \bar{A})^*$ such that $x' \in f(x) \cdot w$.

[1] unambiguity can be generalized to hypergraphs by viewing every binary relation of the form $\exists \boldsymbol{z_1 z_2 z_3}\, a(\boldsymbol{z_1}, x, \boldsymbol{z_2}, y, \boldsymbol{z_3})$ with tuples of FO-variables $\boldsymbol{z_1}$, $\boldsymbol{z_2}$ and $\boldsymbol{z_3}$ of adequate lengths as a primitive binary relation.

In other words, a morphism $f : G \to G'$ is a connecting morphism when every vertex of graph G' is connected to the image of a vertex of G in graph G'.

Examples. Clearly, every surjective (i.e. onto) morphism is a connecting morphism. Another example of (non surjective) connecting morphism $f : I_2 \to G$ is depicted in Figure 2.

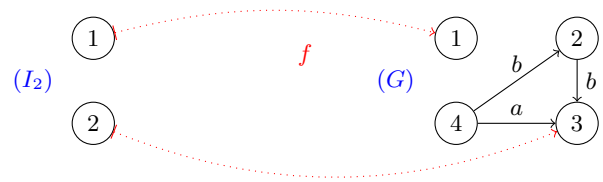

Fig. 2. A connecting morphism $\varphi : I_2 \to G$ with $\varphi(1) = 1$ and $\varphi(2) = 3$.

Remark. Observe that when both G and G' are unambiguous, then, for every $x \in V$, every $w \in (A + \bar{A})^*$, if $x \cdot w$ is not empty then so is $f(x) \cdot w$ and we have $f(x \cdot w) = f(x) \cdot w$. This leads us to the following Lemma.

Lemma 3 (Unique morphism completion). *Let G, G_1 and G_2 be three graphs. Let $f_1 : G \to G_1$ and $f_2 : G \to G_2$ be two graph morphisms. Assume that f_1 is connecting and that both G_1 and G_2 are unambiguous. Then there exists at most one morphism $g : G_1 \to G_2$ such that $g \circ f_1 = f_2$. Moreover, if f_2 is connecting, then so is g.*

Clearly, the composition of two connecting morphisms is a connecting morphism. Since the identity mapping over a graph is also a connecting morphism, this allows us to define the following categories.

Definition 4 (Induced categories). Let $\mathbf{CGrph}(A)$ (resp. $\mathbf{UCGrph}(A)$) be the category defined by finite graphs (resp. by finite unambiguous graphs) as objects and connecting morphisms as arrows.

We aim now at studying the properties of both category $\mathbf{CGrph}(A)$ and category $\mathbf{UCGrph}(A)$ and, especially, the way they are related. The notion of unambiguous congruence defined below allows us to transform any graph into its greatest unambiguous image. In group theory, this generalizes the notion of Stallings foldings [29].

Definition 5 (Unambiguous congruence). Let $G = \langle V, \{E_a\}_{a \in A} \rangle$ be a graph on the alphabet A. A relation $\simeq \subseteq V \times V$ over the vertices of G is an *unambiguous congruence* when it is an equivalence relation such that, for every $a \in A$, for every $x, y \in V$, if $x \simeq y$ then we have both $x \cdot a \times y \cdot a \subseteq \simeq$ and $x \cdot \bar{a} \times y \cdot \bar{a} \subseteq \simeq$.

The existence of a least congruence is stated in Lemma 6 and the associated universality property is stated in Lemma 7.

Lemma 6 (Least unambiguous congruence). *Let G be a graph, possibly ambiguous. Then there exists a least unambiguous congruence $\simeq_G$ over G. Moreover, in the case G is unambiguous, then $\simeq_G$ is the identity relation.*

The graph $G/\simeq_G$ is called the *greatest unambiguous graph image* of the graph G. Its maximality is to be understood in the following sense.

Lemma 7 (Maximal unambiguous image). *Let G be a graph. Let $\simeq_G$ be its least unambiguous congruence. Then, for every graph morphism $f : G \to H$ with unambiguous graph H, there exists a unique morphism $g : G/\simeq_G \to H$ such that $f = g \circ \eta_{\simeq_G}$. Moreover, if f is connecting then so is g.*

Example. An example of maximal graph image is provided by the graphs already depicted in Figure 1 where G_2 has not been chosen at random since $G_2 = G_1/\simeq_{G_1}$.

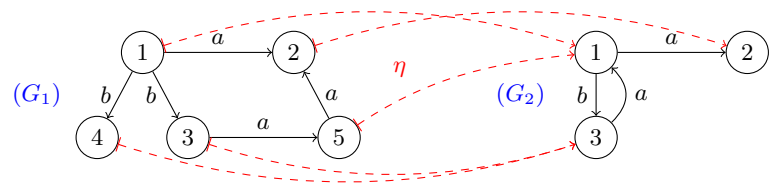

Fig. 3. Graph G_2 is the maximal unambiguous image of graph G_2.

The canonical onto morphism $\eta : G_1 \to G_1/\simeq_{G_1} = G_2$ is depicted in Figure 3, encoding the least unambiguous congruence on G_1 that glues 1 with 5, and 3 with 4.

Remark. The construction described above is a generalization of what is known in algebra as Stallings folding [29]. Observe that with $G = \langle V, \{E_a\}_{a\in V}\rangle$, the least unambiguous congruence $\simeq_G$ equals the least fixpoint of the mapping $F : V \times V \to V \times V$ defined by

$$F(R) = R \cup \bigcup \{(x \cdot a) \times (y \cdot a) \cup (x \cdot \bar{a}) \times (y \cdot \bar{a}) : (x, y) \in R, a \in A\}$$

that contains the equality. It follows, by applying classical fixpoint techniques, that $\simeq_G = \bigcup_{n\geq 0} F^n(=)$, henceforth it can be computed in quasi linear time. In other words, computing the maximal unambiguous image $G/\simeq_G$ of the graph G can be done in time quasi linear in the size of the graph G.

Clearly, the category $\mathbf{UCGrph}(A)$ is a subcategory of $\mathbf{CGrph}(A)$. The next lemma shows shows that maximal graph images extend to morphisms henceforth defining a projection functor from $\mathbf{CGrph}(A)$ into $\mathbf{UCGrph}(A)$.

Lemma 8 (Projected morphisms). *Let G and H be two graphs with a connecting morphism $f : G \to H$. Let $\eta_G : G \to G/\simeq_G$ and $\eta_H : H \to H/\simeq_H$ be the related canonical onto morphisms. Then there exists a unique connecting morphism $\varphi(f) : G/\simeq_G \to H/\simeq_H$ such that $\varphi(f) \circ \eta_G = \eta_H \circ f$.*

In other words, we can define the functor $\varphi : \mathbf{CGrph}(A) \to \mathbf{UCGrph}(A)$ by $\varphi(G) = G/\simeq_G$ for every graph G and by $\varphi(f)$ as given by Lemma 8 for every connecting morphism f. Then, we have $\varphi(G) = G$ for every unambiguous graph

G and $\varphi(f) = f$ for every connecting graph morphism f between unambiguous graphs. In other words, φ is a projection from $CG(A)$ into $\mathbf{UCGrph}(A)$ henceforth a left inverse of the inclusion functor from $\mathbf{UCGrph}(A)$ to $\mathbf{CGrph}(A)$.

We study a bit further the morphisms in these categories showing that they both admit pushouts. The following definition, classical in category theory, is given here for the sake of completeness.

Definition 9 (Pushouts). Let $\langle f_1 : G \to G_1, f_2 : G \to G_2\rangle$ be a pair of morphisms. A pair of morphisms $\langle g_1 : G_1 \to H, g_2 : G_2 \to H\rangle$ is a pushout of the pair $\langle f_1, f_2\rangle$ when $f_1 \circ g_1 = f_2 \circ g_2$, and, for every other pair of morphisms $\langle g'_1 : G_1 \to H', g'_2 : G_2 \to H'\rangle$, if $f_1 \circ g'_1 = f_2 \circ g'_2$ then there exists a unique morphism $h : H \to H'$ such that $g'_1 = h \circ g_1$ and $g'_2 = h \circ g_2$.

The first pushout lemma, in the category $\mathbf{CGrph}(A)$, is a slight generalization of the pushout in the category $\mathbf{Set}$.

Lemma 10 (Synchronization). *In category* $\mathbf{CGrph}(A)$*, every pair of morphisms with common source has a pushout.*

Proof (sketch of). Let $\equiv_{f_1,f_2}$ be the equivalence relation over the vertices of the disjoint sum $G_1 + G_2$ induced by $f_1(x) \equiv_{f_1,f_2} f_2(x)$ for every vertex x of G. Let $H = G_1 + G_2/\equiv_{f_1,f_2}$. Then, the pair $\langle \eta_{\equiv_{f_1,f_2}} \circ i_1, \eta_{\equiv_{f_1,f_2}} \circ i_2\rangle$ with canonical injection i_1 (resp. i_2) of G_1 (resp. G_2) into $G_1 + G_2$ is a pushout of $\langle f_1, f_2\rangle$ in category $\mathbf{CGrph}(A)$. □

Example. An example of such a pushout in the category $\mathbf{CGrph}(A)$ is depicted in Figure 4.

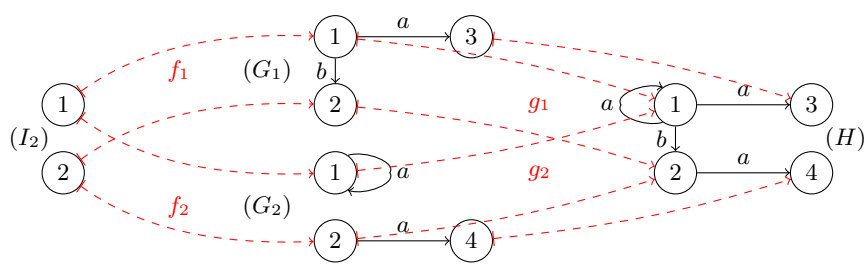

Fig. 4. A "synchronization" pushout example.

Remark. Existence of pushouts in $\mathbf{CGrph}(A)$ essentially follows from the existence of pushouts in the category $\mathbf{Set}$. These pushouts are called synchronization (or glueing) pushouts since, the pushout of $\langle f_1 : G \to G_1, f_2 : G \to G_2\rangle$ essentially glues the vertices of G_1 and G_2 that have common ancestors in G either via f_1 or via f_2.

The second pushout lemma, in the category $\mathbf{UCGrph}(A)$, is completed by a fusion phase (or glueing propagation) defined by taking the maximal unambiguous image of the graph resulting from the pushout in $\mathbf{CGrph}(A)$.

Lemma 11 (Synchronization and fusion). *In category* $\mathbf{UCGrph}(A)$*, every pair of morphisms with common source has a pushout.*

Proof (sketch of). Take $H = G_1 + G_2 / \simeq_{f_1,f_2}$ as for Lemma 10 with pushout $\langle g_1, g_2 \rangle$. Then, take $U = H / \simeq_H$ the greatest unambiguous image of H. The pair $\langle \eta_H \circ g_1, \eta_H \circ g_2 \rangle$ is a pushout of $\langle f_1, f_2 \rangle$ in $\mathbf{UCGrph}(A)$. □

Example. An example of a synchronization + fusion is depicted in Figure 5.

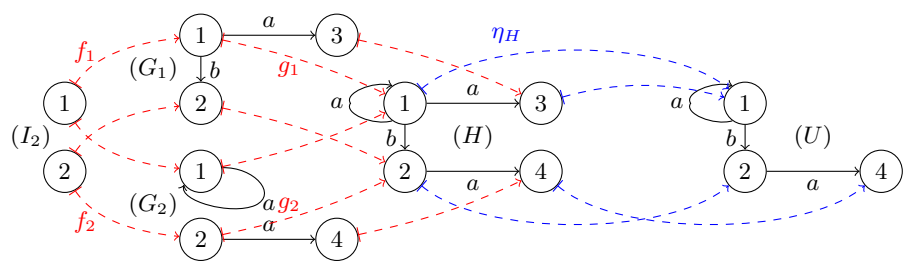

Fig. 5. A "synchronization + fusion" pushout example.

4 The inverse monoid of birooted graphs

We are now ready to define birooted graphs as certain cospans in the category $\mathbf{UCGrph}(A)$. For such a purpose, for every integer $k > 0$, let I_k be the unambiguous defined by k distinct vertices $\{1, 2, \cdots, k\}$ and empty edge relations, and let $id_k : I_k \to I_k$ be the identity isomorphism.

Definition 12 (Birooted graphs). A birooted graph B is a pair of connecting morphisms

$$B = \langle in : I_p \to G, out : I_q \to G \rangle$$

from two trivial graphs I_p and I_q to a common unambiguous graph G.

The morphism *in* is called the *input root morphism*, or, more simply, the *input root* of the birooted graph B. The morphism *out* is called the *output root morphism*, or, more simply, the *output root* of the birooted graph B.

The pair of positive integers (p, q) that defines the domains of root morphisms is called the *type* of the birooted graph. It is denoted by $dom(B)$. The underlying graph G is the codomain of the input and output morphisms. It is called the *graph* of B and it is also denoted by $cod(B)$.

Remark. A birooted graph of type (p, q) can simply be seen as a unambiguous graph $G = \langle V, \{E_a\}_{a \in A} \rangle$ enriched with two tuples of distinguished vertices $(x_1, x_2, \cdots, x_p) \in V^p$ and $(y_1, y_2, \cdots, y_q) \in V^q$ that label the vertices marked by the input and the output roots of the birooted graph.

This point of view is depicted in Figure 6 with two birooted graphs B_1 and B_2 of type $(2, 2)$. In such a figure, vertices of input roots are marked by dangling input arrows, and vertices of output roots are marked by dangling output arrows.

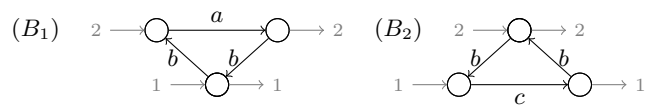

Fig. 6. Examples of $(2, 2)$-birooted graphs.

Remark. The name "birooted graphs" is borrowed from [31]. However, our definition is a clear generalization of the definition given in [31]. Indeed, Stephen's birooted graphs are only birooted graphs of type $(1, 1)$.

In category theoretical term, a birooted graph is a cospan (see for instance [4]). The existence of pushouts in the category $\mathbf{UCGrph}(A)$ allows us to define the product of birooted graphs as the product of their cospan. However, such a product is (so far) not uniquely determined since, a priori, it may depend on the chosen pushout.

Definition 13 (Birooted graph product instance). Let $B_1 = \langle in_1, out_1 \rangle$ and let $B_2 = \langle in_2, out_2 \rangle$ be two birooted graphs. Assume that B_1 is of type (p, q) and that B_2 is of type (q, r). Let $\langle h_1, h_2 \rangle$ be a pushout of the pair $\langle out_1, in_2 \rangle$. Then, the *product instance* of birooted graphs *via the pushout* $\langle h_1, h_2 \rangle$ is defined to be the birooted graphs $\langle h_1 \circ in_1, h_2 \circ out_2 \rangle$, and it is denoted by $B_1 \cdot_{h_1,h_2} B_2$.

A concrete example of a product instance built from the $(2, 2)$-birooted graphs given in Figure 6 is depicted in Figure 7.

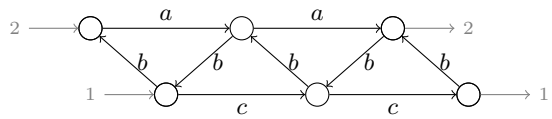

Fig. 7. A product instance of $B_1 \cdot B_2 \cdot B_1 \cdot B_2$.

We aim now at defining products of birooted graphs up to some adequate notion of birooted graph equivalence. This is done via the notion of birooted graph morphisms (Definition 14) and the proof that birooted graph product instances are stable under birooted graph morphisms (Lemma 15).

Definition 14 (Birooted graph morphisms). Let $B_1 = \langle in_1, out_1 \rangle$ and $B_2 = \langle in_2, out_2 \rangle$ be two birooted graphs. A birooted graph morphism from B_1 to B_2 is defined as root preserving graph morphism of their codomain, that is, a graph morphism $h : cod(B_1) \to cod(B_2)$ such that $in_2 = h \circ in_1$ and $out_2 = h \circ out_1$. Such a morphism is denoted by $h : B_1 \Rightarrow B_2$.

Two birooted graphs B_1 and B_2 are isomorphic when there is an isomorphism $h : B_1 \Rightarrow B_2$. Such a situation is denoted by $B_1 \sim B_2$.

Remark. Thanks to Lemma 3, there exists at most one morphism $h : B_1 \Rightarrow B_2$ between any two birooted graphs B_1 and B_2.

Lemma 15 (Product stability w.r.t. birooted graphs morphisms). *Let $f_1 : B_1 \Rightarrow C_1$ and $f_2 : B_2 \Rightarrow C_2$ be two birooted graphs morphisms and let $B_1 \cdot B_2$ and $C_1 \cdot C_2$ be two product instances. Then, there exists a (unique) birooted graphs morphisms $h : B_1 \cdot B_2 \Rightarrow C_1 \cdot C_2$.*

This stability property allows us to define the following birooted graph algebras.

Definition 16 (Birooted graph algebras). Let $HS(A)$ be the set of classes of isomorphic birooted graphs extended with the emptyset equipped with the product defined for every $X, Y \in H(S)$ as follows. In the case there is $B \in X$, $C \in Y$ and a product instance $B \cdot C$, then we take $X \cdot Y = [B]_\sim \cdot [Y]_\sim = [B \cdot Y]_\sim$ and we take $X \cdot Y = \emptyset$ in all other cases.

Notation. In the sequel we shall simply write B (or C) instead of $[B]$ (or $[C]$) and we shall simply write $B \cdot C$ for the product $[B]_\sim \cdot [C]_\sim$ of the corresponding classes of equivalent birooted graphs.

Theorem 17 (Semigroup property). *The algebra $HS(A)$ is a semigroup, that is, the product of birooted graphs is an associative operation.*

Lemma 18 (Idempotent property). *A non-zero birooted graph B of the form $B = \langle in, out \rangle$ is idempotent, that is, $B \cdot B = B$, if and only if $in = out$. Moreover, idempotent birooted graphs commute henceforth form a subsemigroup.*

Theorem 19 (Inverse semigroup property). *The semigroup $HS(A)$ is an inverse semigroup, that is, for every element B, there is a unique element B^{-1} such that*

$$B \cdot B^{-1} \cdot B = B \text{ and } B^{-1} \cdot B \cdot B^{-1} = B^{-1}$$

The inverse B^{-1} of a non-zero birooted graph $B = \langle in, out \rangle$ is simply given by $B^{-1} = \langle out, in \rangle$.

Inverses allow us to define left and right projections that, following inverse semigroup theory, characterize left and right Green classes.

Definition 20 (Left and right projection). Let $B \in HS(A)$ be a birooted graph. The left projection B^L of the birooted graph B is defined by $B^L = B^{-1} \cdot B$. The right projection B^R of the birooted graph B is defined by $B^R = B \cdot B^{-1}$.

Lemma 21. *Let $B = \langle in, out \rangle$ be a non-zero birooted graph. Then we have $B^L = \langle out, out \rangle$ and $B^R = \langle in, in \rangle$.*

Remark. As a general matter of fact, the relation $B \preceq C$ defined over birooted graphs when there exists a (root preserving) morphism $h : C \Rightarrow B$ is a (partial) order relation. We shall see now that it has an algebraic characterization in inverse semigroup theory: it is the natural order [26].

Definition 22 (Natural order). The natural order $\leq$ is defined over birooted graphs by $B \leq C$ when $B = B^R \cdot C$ (or, equivalently, $B = C \cdot B^L$).

Theorem 23 (Natural order vs birooted graph morphisms). *In the inverse semigroup $HS(A)$, the absorbant element 0 is the least element under the natural order and, for every pair of non zero birooted graphs B and C, $B \leq C$ if, and only if, there is a birooted graph morphism $h : C \Rightarrow B$.*

The inverse semigroup of birooted graphs gives a fairly simple though mathematically robust way to compose birooted graphs one with the other. Now we aim at characterizing a simple set of generators for this semigroup.

Definition 24 (Elementary birooted graphs). A elementary birooted graph is either zero or any birooted graph among I_m, $P_{m,i,j}$, $T_{m,a}$, $T_{m,\bar{a}}$ F_m or J_m defined below. In the case $m = 3$ these graphs are depicted in Figure 8.

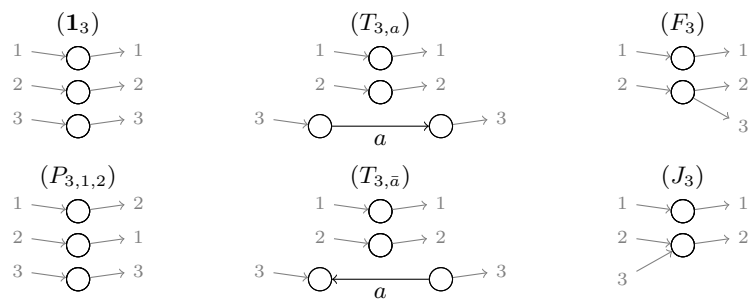

Fig. 8. Elementary birooted graphs.

Formally, the birooted graph $P_{m,i,j} = \langle id_m : I_m \to I_m, out : I_m \to I_m \rangle$ is defined for any $m > 0$ and $1 \leq i, j \leq m$ by $out(i) = j$, $out(j) = i$ and $out(k) = k$ for every other $1 \leq k \leq m$. It is called a *root permutation*. As a particular case, when $i = j$, since $P_{m,i,j} = \langle id_m, id_m \rangle$, the birooted graph $P_{m,i,i}$ is denoted by $\mathbf{1}_m$ instead and called a *root identity*.

The birooted graphs $F_m = \langle id_{m-1} : I_{m-1} \to I_{m-1}, out : I_m \to I_{m-1} \rangle$ and $J_m = \langle in : I_m \to I_{m-1}, id_{m-1} : I_{m-1} \to I_{m-1} \rangle$ are defined for any $m > 1$, by $in(m) = out(m) = m - 1$ and $in(k) = out(k) = k$ for every $1 \leq k \leq m - 1$. They are called a *root fork* and a *root join*.

The birooted graph $T_{m,a} = \langle int : I_m \to G_a, out : I_m \to G_a \rangle$ is defined for any $m > 0$ and $a \in A$, by G_a being the $m + 1$ vertex graph with set of vertices $V = \{1, \cdots, m, m + 1\}$ and sets of edges $E_a = \{(m, m + 1)\}$ and $E_b = \emptyset$ for every $b \neq a$, with $in(m) = m$, $out(m) = m + 1$ and $in(k) = out(k) = k$ for every other $1 \leq k < m$. It is called a *forward edge*. The birooted graph $T_{m,\bar{a}} = T_{m,a}^{-1}$ is called a *backward edge*.

Examples. Some birooted graphs generated by elementary graphs are depicted in Figure 9.

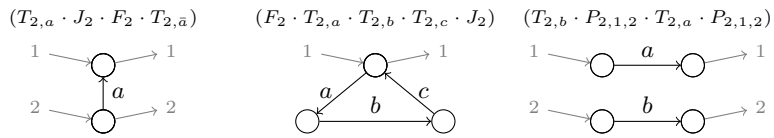

Fig. 9. Some elementary compositions.

Theorem 25. *Every birooted graphs $\langle in : I_p \to G, out : I_q \to G\rangle$ with n vertices in G is finitely generated from 0 and the elementary birooted graphs $\mathbf{1}_k$, $P_{k,i,j}$, $T_{k,a}$, $T_{k,\bar{a}}$, F_k and J_k with $1 \leq k \leq max(n, p+1, q+1)$.*

Definition 26 (Bounded birooted graphs algebras). For any given integer $m > 0$, let $HS_m(A)$ (resp. $HS_{\leq m}(A)$) be the algebraic structure defined as the subsemigroup of $HS(A)$ generated by $\mathbf{1}_m$, $P_{m,i,j}$, $T_{m,a}$, $T_{m,\bar{a}}$ (resp. $\mathbf{1}_k$, $P_{k,i,j}$, $T_{k,a}$, $T_{k,\bar{a}}$, F_k and J_k with $1 \leq k \leq m$).

As an corollary of Theorems 17 and 19, we have:

Theorem 27. *For every integer $m > 0$, the algebra $HS_m(A)$ is an inverse monoid with neutral element $\mathbf{1}_m$.*

Remark. As a particular case, it can be shown that $HS_1(A)$ is the free inverse monoid $FIM(A)$ generated by A. We shall see below that birooted grids of arbitrary size but of type $(2, 2)$ belong to $HS_{\leq 2}(A)$. In other word, in Theorem 25, the bound given for k, depending on the number of vertices of G is not optimal.

5 Languages of birooted graphs

Now we aim at developing the language theory of higher dimensional strings, that is to say, the study of the definability of subsets of $HS(A)$. For such a purpose, we consider the First Order (FO) logic or the Monadic Second Order (MSO) logic (see [9]) on birooted graphs. We refer the reader to the book [9] for a definition of MSO on graphs.

More precisely, we consider $HS_{\leq m}(A)$ so that the number of input and output roots on graphs is bounded. Then, one can enrich the signature A by $2 * m$ symbols, necessarily interpreted as singletons in order to describes these roots. Clearly, this is easily done within FO or MSO logic and we can thus consider the class of FO-definable or MSO-definable languages of birooted graphs.

Theorem 28 (Undecidability). *When $m \geq 2$, the language emptiness problem for FO-definable (hence also MSO-definable) languages of birooted graphs of $HS_{\leq m}(A)$ is undecidable.*

Proof (sketch of). The undecidability of FO follows from the fact that, as soon as $m \geq 2$, as depicted in Figure 10, grids of arbitrary size can be finitely generated with two edge relations a and b modeling horizontal and vertical directions, hence, together with additional edge relations for encoding arbitrary unary predicates on grid vertices, classical undecidability results apply [9]. □

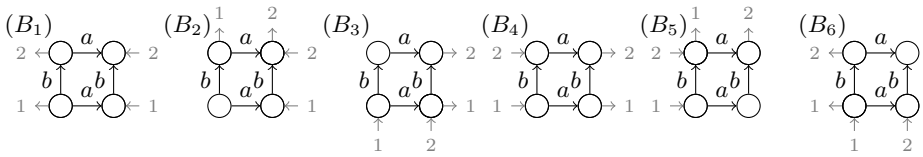

Fig. 10. A finite set of generators B_1,B_2, B_3, B_4, B_5 and B_6.

We first check, following the examples depicted in Figure 9, that these generators can indeed be defined by means of $P_{k,i,j}$, $T_{k,a}$, $T_{k,\bar{a}}$, F_k and J_k with $1 \leq k \leq 2$. For instance, we have $B_5 = (T_{2,\bar{b}} \cdot J_2)^R \cdot T_{2,a} \cdot T_{2,b} \cdot (T_{2,a} \cdot J_2)^R \cdot P_{2,1,2}$.

Then, as depicted in Figure 11, we can generate birooted grids of arbitrary size by taking the $(2,2)$-birooted graph $B_{m,n}$ defined by $G_{m,n} = (Z_m \cdot Y_m)^n$. Clearly, B_{mn} contains a grid of size m by $2 * n$.

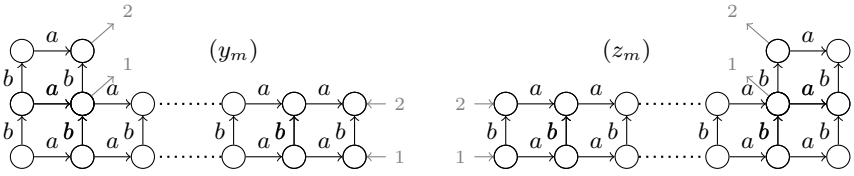

Fig. 11. The $(2,2)$-birooted graphs $Y_m = (B_1)^m \cdot B_2 \cdot B_3$ and $Z_m = (B_4)^m \cdot B_5 \cdot B_6$.

One may ask how generating such graphs of unbounded tree-width can be avoided. It occurs that this can simply be done by restricting the overlaps that are allowed in product instances.

Recently introduced in the context of birooted words [16] or trees [15, 17] languages, the definition of the disjoint product, extended to birooted graphs, makes this restriction of overlaps formal.

Definition 29 (Disjoint product). Let $B_1 = \langle in_1, out_1 \rangle$ and $B_2 = \langle in_2, out_2 \rangle$ be two birooted graphs. Let $\langle h_1, h_2 \rangle$ be a pushout of $\langle out_1, in_2 \rangle$ in $\mathbf{UCGrph}(A)$ and let $B_1 \cdot B_2 \langle in, out \rangle$ with $in = h_1 \circ in_1$ and $out = h_2 \circ out_2$ be the resulting product. Then this product is a *disjoint product* when the pair $\langle h_1, h_2 \rangle$ is also a pushout of in $\langle out_1, in_2 \rangle$ in the category $\mathbf{CGrph}(A)$. In this case, the disjoint product is denoted by $B_1 \star B_2$.

In other words, a birooted graph product is a disjoint product when the fusion phase in the underlying pushout computation is trivial. Although partially defined, this disjoint product is still associative in the following sense.

Lemma 30 (Partial associativity). *For all birooted graphs B_1, B_2, B_3 the disjoint product $B_1 \star (B_2 \star B_3)$ is defined if and only if the disjoint product $(B_1 \star B_2) \star B_3$ is defined and, in that case, the products are equal.*

Then, the closure under disjoint products and left and right projections are defined as follows.

Definition 31 (Disjoint closure and decomposition). Let $X \subseteq HS(A)$ be a set of birooted graphs. The *disjoint closure* of the set X is defined to be the least set Y of birooted graphs such that $X \subset Y$ and that Y is closed under disjoint product and left and right projections. This closure is denoted by $\langle X \rangle_{\star,L,R}$.

For every birooted graph $B \in \langle X \rangle_{\star,L,R}$, a combination of elements of X by disjoint products and let and right projection that equals B is called a *disjoint decomposition* of B over X.

Examples. The subset of $HS_1(A)$ generated by disjoint products of elementary birooted graphs I_1 and $T_{1,a}$ with $a \in A$ is just the free monoid A^*. Adding left and right projections, the disjoint closure of such a set is known in the literature as the free ample monoid $FAM(A)$ whose elements are positive birooted trees (see [12]). Adding backward edges $T_{1,\bar{a}}$ for every $a \in A$, the disjoint closure of the resulting set is the free inverse monoid $FIM(A)$ whose elements are arbitrary birooted trees.

Theorem 32 (Decidability and complexity). *Let $X \subseteq_{fin} HS(A)$ be a finite subset of $HS(A)$. Then, the emptiness problem for MSO-definable subsets of the disjoint closure $\langle X \rangle_{\star,R,L}$ is (non-elementary) decidable.*

Moreover, for any MSO-definable language $L \subseteq \langle X \rangle_{\star,R,L}$, the membership problem $B \in L$ for any $B \in HS(A)$ is linear in the size of any disjoint decomposition of B over X.

Proof (sketch of). Every disjoint product in $\langle X \rangle_{\star,R,L}$ is just a disjoint sum with a bounded glueing of roots. It follows that MSO decomposition techniques (see [30] or [32]) combined with partial algebra techniques [7] are available, as done in [5] for languages of labeled birooted trees, to achieve an algebraic characterization of MSO definable languages in terms of (partial algebra) morphisms into finite structures. Such an approach also proves the complexity claim for the membership problem. □

Remark. Of course, the membership problem is non elementary in the size of the MSO formula that defines L. This already follows from the case of MSO definable languages of finite words. Also, the problem of finding disjoint decompositions over X for birooted graphs may be delicate and is left for further studies.

As observed above, A^*, $FAM(A)$ and $FIM(A)$ are examples of subsemigroup of $HS(A)$ that are finitely generated by disjoint product, inverses and/or projections [15, 17]. By applying Theorem 32, this proves (again) that their MSO definable subsets have decidable emptiness problem.

6 The inverse monoid of acyclic birooted graphs

Towards application purposes, birooted graphs can be seen as models of computerized system behaviors with vertices viewed as (local) states and edges viewed as (local) transition. In this case, one is tempted to detect and forbid directed cycles which interpretation could be problematic (causally incoherent).

As an illustration of the power of the inverse semigroup framework that is proposed here, we show how these birooted acyclic graphs can simply be defined as the quotient of the inverse semigroup of birooted graphs by the semigroup ideal of cyclic ones. Then, in such a quotient, easily implementable, a product of acyclic birooted graphs is causally coherent if and only if it is non zero.

Lemma 33 (Semigroup ideal). *Let φ be a graph property that is preserved under graph morphisms. Let I_φ be the set $I_\varphi \subseteq HS(A)$ that contains 0 and all birooted graphs whose underlying graph satisfies φ. Then, I_φ is an semigroup ideal of $HS(A)$, that is,*

$$HS(A) \cdot I_\varphi \subseteq HS(A) \text{ and } I_\varphi \cdot HS(A) \subseteq HS(A)$$

and the Rees' quotient $HS(A)/I_\varphi$, that is, the set $HS(A) - I_\varphi + \{0\}$ equipped with the product defined as in $H(A)$ when the result does not belong to I_φ and defined to be 0 otherwise, is still an inverse semigroup.

In other words, much in the same way 0 already appears with products in $HS(A)$ that have no compatible types, when the property φ describes, in some concrete modeling context, a set of faulty models that is preserves under morphism, then the product in $HS(A)/I_\varphi$ equals 0 also when the resulting birooted graph is faulty.

Clearly, the existence of directed cycles is a property preserved by morphism. Then, the algebra of birooted acyclic graphs can simply be modeled as the inverse semigroup $HS(A)/I_C$ where $I_C \subseteq HS(A)$ is the resulting semigroup ideal containing 0 and all (directed) cyclic birooted graphs.

Such a situation is depicted in Figure 12 where

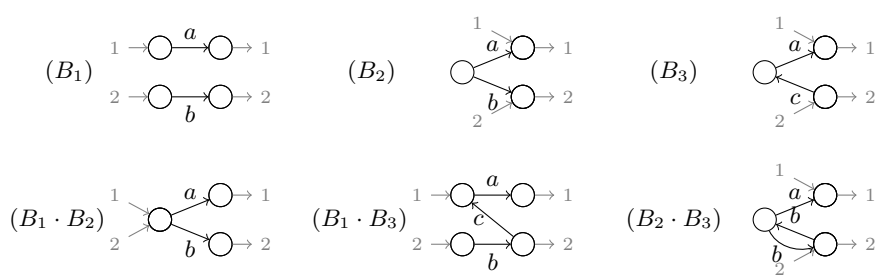

Fig. 12. Causal constraints propagation via products.

examples show how products of birooted graphs may propagate causality constraints eventually leading to non-causal graphs: the product $(B_2 \cdot B_2)$.

In other words, with the proposed approach, one can define a modeling software in such a way that non-causal models raised by combination of causal constraints are easily detected and forbidden, while, at the same time, the underlying algebraic framework still lays in the theory of inverse semigroups.

7 Conclusion

We have shown how a rather simple and intuitive composition operation on graphs, inherited from long standing ideas (see [27]), induces a rich algebraic structure, an inverse semigroup, from which one can define a natural order and other mathematically robust operators such as left and right projections, that capture graph theoretical concepts.

Of course, defining graph products by means of cospans products has already a long history in Theoretical Computer Science (see e.g. [13, 6, 4]). The originality of our approach consists in restricting to the category of unambiguous graphs and connecting morphisms that allow the resulting semigroup to be an inverse semigroup.

Still, this inverse semigroup is far from being understood in depth. Little is known about its subsemigroups. Thanks to [31], one can easily show that, all A generated E-unitary inverse semigroups (see also [28]) are subsemigroups of the monoid defined by birooted graphs of type (1,1). This suggests that the semigroup $HS(A)$ may satisfy some universality property that is still to be discovered. Also, we have no direct characterizations of the subsemigroups of $HS(A)$ that could be defined by bounding the number of roots on generators.

Following [5], by restricting the product to disjoint product, techniques arising from partial algebras [7] are applicable allowing us to inherit from the existing MSO-language theory of graphs of bounded tree-width [8, 9]. It is expected that tile automata, defined in [15, 16] over birooted words or trees, can easily be extended to higher dimensional strings and related with MSO-definability. Yet, closure property of MSO-definable languages remains to be detailled. It is by no means clear under which restrictions the product of two definable languages remains definable. Also, defining more suitable subsemigroups of (possible Rees' quotient of) $HS(A)$ that would also have decidable MSO languages is still to be investigated.

With a view towards application, beyond all experiments mentioned in the introduction, the modeling power of birooted graphs also needs to be investigated further in both practical modeling problems and more general modeling theories. For such a purpose, an implementation of the monoid $HS(A)$ with both graphical and programmatic views of its elements is scheduled. As already mentioned, multiple roots gives a flavor of concurrency. It is also expected that higher dimensional strings can be used as (explicitly concurrent) models of partially semi-commutative traces [11, 19] henceforth connecting higher dimensional strings with a part of concurrency theory.

Finally, it has been shown recently that (one head) tree and graph walking automata semantics is nicely described in terms of (languages of) birooted graphs

with single input and output roots [20]. The generalized birooted graphs presented here may provide nice semantical models of multi-head walking automata: partial runs of these automata clearly define languages of birooted graphs with multiple input and output roots.

Acknowledgements

The idea of developing a notion of higher dimensional strings has been suggested to the author by Mark V. Lawson in 2012. Their presentations have also benefited from numerous and helpful comments from anonymous referees of serval versions of this paper.

References

1. S. Abramsky. A structural approach to reversible computation. *Theor. Comp. Sci.*, 347(3):441–464, 2005.
2. J.-R. Abrial. *Modeling in Event-B - System and Software Engineering*. Cambridge University Press, Cambridge, 2010.
3. F. Berthaut, D. Janin, and B. Martin. Advanced synchronization of audio or symbolic musical patterns: an algebraic approach. *International Journal of Semantic Computing*, 6(4):409–427, 12 2012.
4. C. Blume, H.J.S. Bruggink, M. Friedrich, and B. König. Treewidth, pathwidth and cospan decompositions with applications to graph-accepting tree automata. *Journal of Visual Languages and Computing*, 24(3):192 – 206, 2013.
5. A. Blumensath and D. Janin. A syntactic congruence for languages of birooted trees. *Semigroup Forum*, 2014.
6. H.J.S. Bruggink and B. König. On the recognizability of arrow and graph languages. In *Graph Transformations*, volume 5214 of *LNCS*, pages 336–350. Springer, 2008.
7. P. Burmeister. *A Model Theoretic Oriented Approach to Partial Algebras*. Akademie-Verlag, 1986.
8. B. Courcelle. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theor. Comp. Sci.*, 80(2):153–202, 1991.
9. 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.
10. V. Danos and L. Regnier. Reversible, irreversible and optimal lambda-machines. *Theor. Comp. Sci.*, 227(1-2):79–97, 1999.
11. V. Diekert, M. Lohrey, and A. Miller. Partially commutative inverse monoids. *Semigroup Forum*, 77(2):196–226, 2008.
12. J. Fountain, G. Gomes, and V. Gould. The free ample monoid. *Int. Jour. of Algebra and Comp.*, 19:527–554, 2009.
13. F. Gadducci and R. Heckel. An inductive view of graph transformation. In *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, Selected Papers*, pages 223–237, 1997.
14. P. Hudak and D. Janin. Tiled polymorphic temporal media. In *Work. on Functional Art, Music, Modeling and Design (FARM)*, pages 49–60. ACM Press, 2014.

15. 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, 07 2013.
16. D. Janin. Overlaping tile automata. In *8th Int. Computer Science Symp. in Russia (CSR)*, volume 7913 of *LNCS*, pages 431–443. Springer, 06 2013.
17. D. Janin. On languages of labeled birooted trees: Algebras, automata and logic. *Information and Computation*, 2014.
18. 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.
19. D. Janin. Free inverse monoids up to rewriting. Research report, LaBRI, Université de Bordeaux, 2015.
20. D. Janin. Walking automata in the free inverse monoid. Research report, LaBRI, Université de Bordeaux, 2015.
21. D. Janin, F. Berthaut, and M. Desainte-Catherine. Multi-scale design of interactive music systems : the libTuiles experiment. In *Sound and Music Comp. (SMC)*, 2013.
22. D. Janin, F. Berthaut, M. DeSainte-Catherine, Y. Orlarey, and S. Salvati. The T-calculus : towards a structured programming of (musical) time and space. In *Work. on Functional Art, Music, Modeling and Design (FARM)*, pages 23–34. ACM Press, 2013.
23. J. Kellendonk. The local structure of tilings and their integer group of coinvariants. *Comm. Math. Phys.*, 187:115–157, 1997.
24. J. Kellendonk and M. V. Lawson. Tiling semigroups. *Journal of Algebra*, 224(1):140 – 150, 2000.
25. J. Kellendonk and M. V. Lawson. Universal groups for point-sets and tilings. *Journal of Algebra*, 276:462–492, 2004.
26. M. V. Lawson. *Inverse Semigroups : The theory of partial symmetries.* World Scientific, 1998.
27. H. R. Lewis. A new decidable problem, with applications (extended abstract). In *IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 62–73. IEEE Press, 1977.
28. 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.
29. 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.
30. S. Shelah. The monadic theory of order. *Annals of Mathematics*, 102:379–419, 1975.
31. J.B. Stephen. Presentations of inverse monoids. *Journal of Pure and Applied Algebra*, 63:81–112, 1990.
32. W. Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science*, volume 1261 of *LNCS*, pages 118–143. Springer, 1997.
33. W. Thomas. Logic for computer science: The engineering challenge. In *Informatics - 10 Years Back, 10 Years Ahead.*, volume 2000 of *LNCS*, pages 257–267. Springer, 2001.