Two-way automata and regular languages of overlapping tiles

Anne Dicky, David Janin

LaBRI, CNRS UMR 5800
Laboratoire Bordelais de Recherche en Informatique

Rapport de recherche RR-1463-12

Two-way automata and regular languages of overlapping tiles

June 5, 2013

Anne Dicky, David Janin,
LaBRI, Université de Bordeaux

Two-way automata and regular languages of overlapping tiles

Anne Dicky, David Janin
Université de Bordeaux, LaBRI UMR 5800,
351, cours de la libération, F-33405 Talence, FRANCE
{dicky | janin}@labri.fr

June 5, 2013

Abstract

We consider classes of languages of overlapping tiles, i.e. subsets of the McAlister monoid: the class *REG* of languages definable by Kleene's regular expressions, the class *MSO* of languages definable by formulas of monadic second-order logic, and the class *REC* of languages definable by morphisms into finite monoids. By extending the semantics of finite-state two-way automata (possibly with pebbles) from languages of words to languages of tiles, we obtain a complete characterization of these classes.

We show that adding pebbles strictly increases the expressive power of two-way automata recognizing languages of tiles, but the hierarchy induced by the number of allowed pebbles collapses to level one.

Our study yields, as an immediate corollary, Shepherdson's result (and its extension to pebble automata) that, for languages of words, every finite-state two-way automaton is equivalent to a finite-state one-way automaton.

1 Introduction

1.1 Background

One-dimensional overlapping tiles already appear in the 70's in *inverse semigroup theory* [26]. As elements of particular quotients of free inverse monoids

[29, 22], known as monoids of McAlister [23, 22], the tiles came to the forefront again in the late 90's in *mathematical physics*, associated with tilings of the Euclidian space [18, 19, 20, 1]. Although implicitly, overlapping tiles also appear in *theoretical computer science* in the studies of zigzag codes and the underlying zigzag covers of finite, infinite or bi-infinite words [2, 6, 24, 1]. Oddly enough, our interest in languages of positive tiles came from application perspectives in *computational music theory* [11]. In particular, tiles and continuous variants may be used to describe advanced synchronization mechanisms between musical patterns [4, 3, 17]; an approach that leads to new programming features for music system design [16].

In *software engineering*, overlapping tiles may be seen as the possible concrete values of string objects extended with history-preserving memory capacities. This point of view turns out to provide a simple presentation of many properties satisfied by one-dimensional overlapping tiles; it also conveys most of the intuition that underlies the work presented here.

Let us thus assume that we are software developers trying to enrich the class of string objects with some history-preserving capacity.

More precisely, for every string object s, let $s \cdot a$ denote the result of *adding* some letter a to the right of the string s, and let $s \cdot \overline{a}$ denote the result of *removing* a from the right of s. With a standard string, $s \cdot a \cdot \overline{a} = s$, and thus $s \cdot a \cdot \overline{a} \cdot b = s \cdot b$ for any letter b. A history-preserving mechanism is a way to prevent a letter to appear in s if a different letter has previously occurred in the same position. Thus our extended strings should satisfy the property

$$s \cdot a \cdot \overline{a} \cdot b = \begin{cases} s \cdot b \text{ if } a = b \\ \text{undefined otherwise} \end{cases}$$

as if adding and removing the letter a to the right of the string s created some footprint of that letter in such a way that no other letter could ever be put on that position.

One-dimensional overlapping tiles faithfully describe the possible sequences of actions (additions or removals of letters) on these extended string objects; and thus the possible values of the objects themselves (as sequences of actions on the empty string). For instance, the sequence $\overline{a}abcba\overline{a}\overline{b}$ is described by the tile

$$a \; bc \; ba$$

where bc is the string to be added, while the left part a and the right part ba of the tile model the footprints left by the other actions. The composition of

actions yields a monoid structure that turns out to be the inverse monoid of McAlister [26, 23].

These examples show that the model of one dimensional overlapping tiles is a versatile model that can be used in many fields. However, the associated language theory can still be developed. Indeed, it occurs the classical tools of formal language theory somehow fail to apply to inverse monoids [25, 34]. To be more precise, the expressive power induced by the usual tools, e.g. automata or algebraic recognizability, collapses when applied to inverse monoids.

In this paper, we aim at developing a computer science-flavored language theory for overlapping tiles. Since adding or removing letters of extended string objects can be interpreted as movements of the head of a two-way automaton [31] on a classical string, finite-state two-way automata appear as natural and expressive candidates to define and study classes of tile languages.

1.2 Outline

The monoid of one-dimensional overlapping tiles is presented in Section 2. A special emphasis is put on the way non-zero tiles are generated from linear walks, thus rephrasing, in the context of one-dimensional tiles, the notion of free inverse monoid captured by the Wagner congruence (Lemma 5). The link with Pécuchet's notion of bisections [28] is specified at the end of the section.

To a specialist of inverse semigroups, most of the material presented in this section is quite straightforward. In particular, our presentation of the monoid of McAlister could be significantly simplified by defining it as a Reese quotient of the free inverse monoid, following the classical Scheiblich-Munn presentation of free inverse monoids [32, 27]. We preferred a direct, standalone presentation to address a more general public, providing an alternative to Lawson's presentation [23] (see also [22], chap. 9, for a relationship with various other classes of semigroups).

From Section 3 we study languages of tiles. The class MSO of languages definable in Monadic Second-Order Logic, a typical yardstick of expressiveness, is first defined and shown both robust (Theorem 9) and simple (Theorem 12). As a consequence of robustness, it is shown that the class REG (resp. k-REG) of languages of tiles definable by Kleene expressions (resp.

Kleene expressions extended with projection on idempotents with a nesting depth at most k) is included into the class MSO. As a consequence of simplicity, it is shown (Theorem 14) that the class MSO is captured by the class *1-REG*: extended regular expressions with no nested projection operator. The class REC of languages recognizable by finite monoids is also studied and related with the other classes. An example of a language in REC is given, illustrating the complete characterization of the class REC given in [15].

Languages definable by means of two-way tile automata are then presented in Section 4. Quite closely related with Pécuchet study [28], two-way tile automata are classical two-way automata over words with a semantics expressed in terms of tiles. Tiles are simply seen as domains of partial runs: runs that may start and stop anywhere on the input words. As a result, we show that the regular languages of tiles (definable by Kleene expressions) are exactly the languages of tiles recognizable by finite-state two-way automata (Theorem 22). We also prove that the (strictly larger, see Theorem 23) class of MSO-definable languages of tiles is the class of languages recognizable by finite-state many-pebble automata . Furthermore, one-pebble automata are shown to capture the whole class of many-pebble automata (Theorem 27). Shepherdson's theorem and analogous results for pebble automata are obtained as immediate corollaries (Corollaries 26 and 30).

To summarize, we prove that for every $k \in \mathbb{N}$

$$REC \subset REG = \textit{0-REG} \subset \textit{(k+1)-REG} = Bool(REG) = MSO$$

with strict inclusions. All these results support the long-standing intuition [28, 5, 21] that the theory of inverse monoids *is* a powerful tool in the study of two-way automata. Indeed, all proofs presented here are quite simple.

1.3 Related works

Two-way automata have been the subject of many studies. This can be explained by their intriguing combinatorial complexity.

For instance, Rabin-Scott-Shepherdson's result [33] that two-way automata are as expressive on words as one way automata was long considered difficult [36]. More precisely, the capacity of two-way automata to read each letter an unbounded number of times makes the structure of two-way automata runs difficult to analyze. This is particularly clear in Pécuchet and

Birget's algebraic studies of two-way automata [28, 5], in which two-way runs give rise to a rich combinatorial structure. A similar complexity is illustrated by Globerman and Harel's result [9] that the number of allowed pebbles in two-way automata induces a "succintness" hierarchy: each additional pebble provides inherent exponential power.

Still, gaining a full understanding of two-way automata, with or without pebbles, remains a challenging topic. The classical theory of (one-way) finite automata has benefited from a rich algebraic language theory that led, and still leads, to many decision algorithms [30]. But, as already observed by Birget [5], there is no similar algebraic characterization of two-way automata that does not amount to essentially reduce two-way automata to one-way automata. Further studies, be them on languages of overlapping tiles [10, 14] or on languages of birooted trees [15, 12], show that some progress can be done along Birget's long-standing open question [5].

Beyond two-way automata, there are two-way transducers. Though this is not the subject of our study, it is not implausible that our approach can be generalized to such transducers.

2 The monoid of overlapping tiles

Here we give a description of monoids of one-dimensional overlapping tiles. They are shown to be isomorphic to monoids of McAlister monoid [23]. The tight link between (two-way linear) walks on words and tiles is formalized by an onto morphism from walks to tiles whose kernel is indeed the Wagner congruence.

2.1 Preliminaries

Given a finite alphabet A, let A^* be the free monoid generated by A and let 1 be the neutral element. The concatenation of two words u and v is denoted by uv.

Let $\leq_p$ stands for the prefix order over A^*, let $\leq_s$ for the suffix order and let $\vee_p$ (resp. $\vee_s$) denotes the join operator for the prefix (resp. suffix) order. For all words u and v, we have that $u \vee_p v$ (resp. $u \vee_s v$) is the least word whose both u and v are prefixes (resp. suffixes). The extended monoid $A^* + \{0\}$ (with $0u = u0 = 0$ for every word u), ordered by $\leq_p$ (extended with $u \leq_p 0$ for every word u), is a lattice; in particular, $u \vee_p v = 0$ whenever

neither u is a prefix of v, nor v is a prefix of u. Symmetric properties hold in the suffix lattice.

Given $\overline{A}$ a disjoint copy of A, let $u \mapsto \overline{u}$ be the mapping from $(A+\overline{A})^*$ to itself inductively defined by $\overline{1} = 1$, $\overline{au} = \overline{u}\,\overline{a}$ and $\overline{\overline{a}u} = \overline{u}\,a$ for every letter $a \in A$ and every word $u \in (A+\overline{A})^*$. The mapping $u \mapsto \overline{u}$ is involutive, i.e. for all words $u \in (A+\overline{A})^*$ we have $\overline{\overline{u}} = u$. It is also an antimorphism of the free monoid $(A+\overline{A})^*$, i.e. for all words u and $v \in (A+\overline{A})^*$ we have $\overline{uv} = \overline{v}\,\overline{u}$. For all $u \in (A+\overline{A})^*$, the word $\overline{u}$ is called the syntactic inverse of u.

The free group $FG(A)$ generated by A is the quotient of $(A+\overline{A})^*$ by the least congruence $\simeq$ such that, for every letter $a \in A$, $a\overline{a} \simeq 1$ and $\overline{a}a \simeq 1$. Let $\preceq$ be the rewriting relation induced by the rules $1 \preceq a\overline{a}$ and $1 \preceq \overline{a}a$ for every $a \in A$. It is well-known that every class $[u] \in FG(A)$ contains a unique element $red(u)$ (the *reduced form* of u) irreducible with respect to $\preceq$, i.e. containing no factor of the form $a\,\overline{a}$ nor $\overline{a}\,a$.

The free inverse monoid $FIM(A)$ generated by A is the quotient of $(A+\overline{A})^*$ by the Wagner congruence $\simeq_W$, i.e. the least congruence such that $u\overline{u}u \simeq_W u$ and $u\overline{u}v\overline{v} \simeq_W v\overline{v}u\overline{u}$ for all $u, v \in (A+\overline{A})^*$.

2.2 Positive and negative tiles

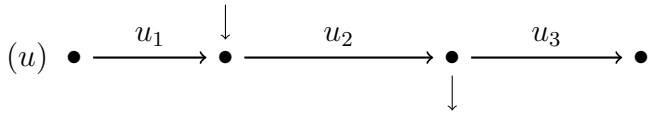
A *tile* over the alphabet A is a triple of words $u = (u_1, u_2, u_3) \in A^* \times (A^* + \overline{A}^*) \times A^*$ such that, if $u_2 \in \overline{A}^*$, its inverse $\overline{u_2}$ is a suffix of u_1 and a prefix of u_3.

When $u_2 \in A^*$ we say that u is a *positive tile*. When $u_2 \in \overline{A}^*$ we say that u is a *negative tile*.

From now on, let T_A, T_A^+ and T_A^- respectively denote the set of tiles, the set of positive tiles and the set of negative tiles over the alphabet A.

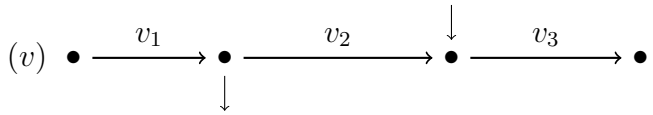
The *domain* of a tile $u = (u_1, u_2, u_3)$ is the reduced form of $u_1u_2u_3$ (always a word of A^*). Its *root path* is the word u_2. When $u_2 \in A^*$, the words u_1 and u_3 are the *contexts* of the tile u.

A positive tile $u = (u_1, u_2, u_3)$ is conveniently drawn as a (linear, unidirectional and left-to-right) Munn's birooted word tree [27]:

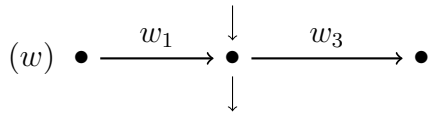


where the dangling input arrow, marking the beginning of the root path and called the *input root* of the tile, appears on the left of the dangling output

arrow, marking the end of the root path and called the *output root* of the tile. A negative tile of the form $v = (v_1v_2, \overline{v_2}, v_2v_3) \in A^* \times \overline{A}^* \times A^*$ is also drawn as a birooted word tree



where the input root now appears on the right of the output root. A tile that is both positive and negative, that is a tile of the form $w = (w_1, 1, w_3) \in A^* \times 1 \times A^*$ is then drawn as follows:



2.3 The inverse monoid of tiles

In this part, we abusively denote a tile (u_1, u_2, u_3) *by any triple* $(u'_1, u'_2, u'_3) \in ((A + \overline{A})^*)^3$ *such that* u_1 *(resp.* u_2*,* u_3*) is the reduced form of* u'_1 *(resp.* u'_2*,* u'_3*).*

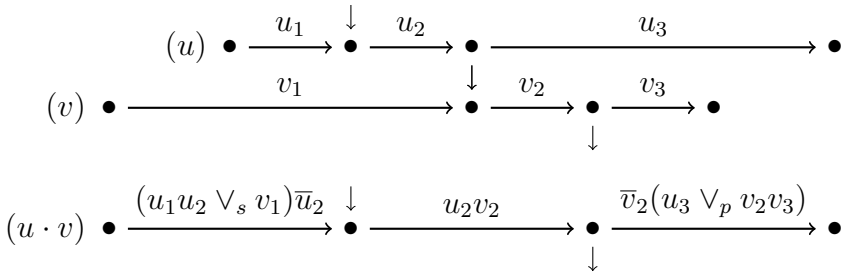
The sequential product of two tiles $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ is defined as

$$u \cdot v = ((u_1u_2 \vee_s v_1)\overline{u_2}, u_2v_2, \overline{v_2}(u_3 \vee_p v_2v_3))$$

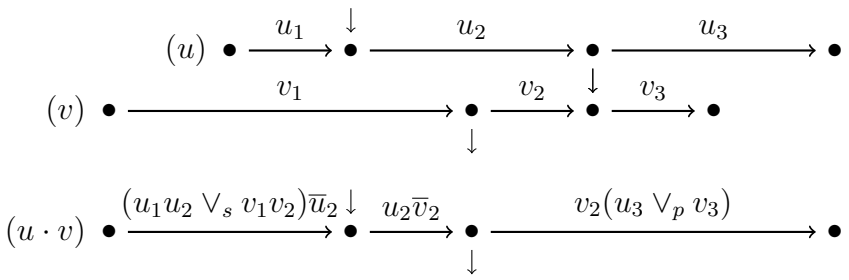
when both pattern-matching conditions $u_1u_2 \vee_s v_1 \neq 0$ and $u_3 \vee_p v_2v_3 \neq 0$. Otherwise, the product $u \cdot v$ is defined to be 0 for some new tile 0. The set T_A is thus extended with the zero tile, sometimes called the *undefined tile*, and we let $u \cdot 0 = 0 \cdot u = 0$ for every $u \in T_A$. In order to keep notation simple, we keep the notation T_A for the set T_A extended with 0. The fact we restrict to non zero tiles shall always be clear from the context.

Remark. Let a, b, c and d be distinct letters of A. Then $(a, b, c) \cdot (b, c, d) = (a, bc, d)$ whereas $(a, b, c) \cdot (a, c, d) = 0$. In the latter case, the left matching constraint is violated since neither of ab and a is a suffix of the other.

Graphically, the product of two tiles is obtained by their superposition (or synchronization) in such a way that the end of the root of the first tile coincides with the beginning of the root of the second tile then by their fusion, which requires pattern-matching conditions to the left and to the right of the synchronization point. With a positive tile $u = (u_1, u_2, u_3)$ and a positive tile $v = (v_1, v_2, v_3)$, the product $u \cdot v$ can be depicted as follows:



With a positive tile $u = (u_1, u_2, u_3)$ and a negative tile $v = (v_1v_2, \overline{v}_2, v_2v_3)$, the product $u \cdot v$ can be depicted, after simplification, as follows:



Observe that, in all cases, the domain of a non-zero product $u \cdot v$ contains both the domain of u and the domain of v. This is a key feature to ensure that the product is associative, as stated and proved in the next Theorem.

Remark. It is straightforward to check that any tile u of the form $u = (u_1, 1, u_3)$ satisfies $u \cdot u = u$. Conversely, if a tile $u = (u_1, u_2, u_3)$ satisfies $u \cdot u = u$, then $u_2u_2 = u_2$ and thus $u_2 = 1$. Thus the tiles that are both positive and negative are indeed the non-zero tiles idempotent for the product. In the sequel, the undefined tile 0 is thus considered to be both positive and negative. Then, tiles that are both positive and negative are just referred to as idempotent tiles.

Theorem 1 *T_A equipped with the sequential product of tiles is an inverse monoid with neutral element $1 = (1, 1, 1)$ and inverses given by $0^{-1} = 0$ and for every non-zero tile $u = (u_1, u_2, u_3) \in T_A$, we have $u^{-1} = (u_1u_2, \overline{u_2}, u_2u_3)$.*

Proof. We first prove that the sequential product is a sound (well-defined) associative operation.

Soundness. Let $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ be tiles such that $u.v \neq 0$. Since u_1 and v_3 are in A^*, the reduced forms of $(u_1u_2 \vee_s v_1)\overline{u_2}$ and $\overline{v}_2(u_3 \vee_s v_2v_3)$

are also in A^*. Thus if $red(u_2v_2) \in A^*$ we have that $u.v$ is a tile.
Suppose $red(u_2v_2) \in \overline{A}^*$. If both tiles u and v are negative, $\overline{v_2}\,\overline{u_2}$ is a suffix of $v_1\overline{u_2}$ (since $\overline{v_2}$ is a suffix of v_1) and a prefix of $\overline{v_2}u_3$ (since $\overline{u_2}$ is a prefix of u_3). If u is negative and v positive then v_2 is a prefix of $\overline{u_2}$ and thus $red(\overline{v_2}\,\overline{u_2})$ is a suffix of u_1 and a prefix of $red(\overline{v_2}\,u_3)$; similarly, if u is positive and v negative then u_2 is a suffix of $\overline{v_2}$ and thus $red(\overline{v_2}\,\overline{u_2})$ is a suffix of $red(v_1\overline{u_2})$ and a prefix of v_3. In all cases, $red(\overline{u_2v_2})$ is a suffix of $red((u_1u_2 \vee_s v_1)\overline{u_2})$ and a prefix of $red(\overline{v_2}(u_3 \vee_p v_2v_3))$, hence $u.v$ is a tile.

Associativity. The pattern-matching conditions are associative: let $u = (u_1, u_2, u_3)$, $v = (v_1, v_2, v_3)$ and $w = (w_1, w_2, w_3)$ non-zero tiles such that each of the products $(u.v).w$ and $u.(v.w)$ is a non-zero tile. This is equivalent to the fact that $u_1u_2v_2 \vee_s v_1v_2 \vee_s w_1 \neq 0$ and $u_3 \vee_s v_2v_3 \vee_s v_2w_2w_3 \neq 0$. In that case we have

$$(u.v).w = ((u_1u_2v_2 \vee_s v_1v_2 \vee_s w_1)\overline{v_2}\,\overline{u_2}, u_2v_2w_2, \overline{w_2}\,\overline{v_2}(u_3 \vee_s v_2v_3 \vee_s v_2w_2w_3))$$

hence we also have $(u.v).w = u.(v.w)$ by symmetry.

Obviously, the element $1 = (1, 1, 1)$ is the neutral element and 0 an absorbing element.

To prove that T_A is an inverse monoid, it suffices to prove [29, 22] that all elements have inverses, i.e. for every $x \in T_A$ there exists $y \in T_A$ such that $xyx = x$ and $yxy = y$, and that idempotents commute. Since 0 is obviously the unique inverse of itself and commutes with every element, we restrict to non-zero tiles.

Existence of inverses.
For every tile $u = (u_1, u_2, u_3) \in T_A$, let $u^{-1} = (u_1u_2, \overline{u_2}, u_2u_3)$. Observe that we indeed have $(u^{-1})^{-1} = u$. Moreover, it is straightforward to check that $u.u^{-1} = (u_1, 1, u_2u_3)$ and $u^{-1}.u = (u_1u_2, 1, u_3)$. Hence $u.u^{-1}.u = u$ and $u^{-1}.u.u^{-1} = u$.

Commutation of idempotents. Let $u = (u_1, 1, u_3) \in T_A$ be an idempotent tile: $u.u = u$ implies $u_2u_2 = u_2$, hence $u_2 = 1$, i.e. u is an idempotent tile. The commutativity of $\vee_s$ and $\vee_p$ implies the commutation of idempotents hence the set E_A of idempotent tiles is indeed a commutative submonoid of T_A. □

Remark. In the case A is a singleton then the product of two tiles is always well defined. There is thus no need of the undefined tile 0. The resulting monoid (without zero) is known to be the free inverse monoid of

one generator. Unless explicitly stated, we assume in this paper that A contains at least two distinct letters.

The following is immediate:

Proposition 2 *The mapping $u \mapsto (1, u, 1)$ from A^* to T_A is an injective morphism.*

In other words, the free monoid A^* can be seen as a submonoid of T_A. In the remainder of the text we may use the same notation for words of A^* and their images in T_A.

From the identities $u^{-1}.u = (u, 1, 1)$ and $u.u^{-1} = (1, 1, u)$ for every word $u \in A^*$, we easily deduce that:

Proposition 3 *The monoid T_A is finitely generated from (the tile images of) letters of A, product and inverses.*

On the contrary, the set T_A^+ of positive tiles (and, similarly, the set T_A^- of negative tiles) is obviously a submonoid of T_A. But this submonoid is *not* finitely generated.

This observation leads us to the following definition. For every $u = (u_1, u_2, u_3) \in T_A$, let $u^L = u^{-1}.u = (u_1u_2, 1, u_3)$ be the *left projection* associated with the tile u, and let $u^R = u.u^{-1} = (u_1, 1, u_2u_3)$ be its *right projection*. Let also $0^L = 0^R = 0$. By construction, we have $u^R.u = u.u^L = u$ for every $u \in T_A$.

Proposition 4 *The submonoid T_A^+ is finitely generated from (the tile images of) letters of A and the left and right idempotent operators.*

Proof. Follows from the identity $(u_1, u_2, u_3) = (1, u_1, 1)^L \cdot (1, u_2, 1) \cdot (1, u_3, 1)^R$ for every $u = (u_1, u_2, u_3) \in T_A^+$. □

2.4 Linear walks and the monoid of McAlister

We provide here an alternative proof that T_A is an inverse monoid by showing that it is isomorphic to the McAlister monoid [23]. Of course, this could be done by showing that Lawson's presentation of McAlister monoid is equivalent to ours. However, we believe that our proof, using the monoid W_A of linear walks, conveys a relevant intuition of the link with two-way automata.

Informally, a (non zero) *walk* over a word $u \in A^*$ is a word w of $(A + \overline{A})^*$ that corresponds to a back-and-forth reading of u (left-to-right reading is

modeled by letters of A, and right-to-left reading by letters of $\overline{A}$). Not all words of $(A+\overline{A})^*$ are walks. Obviously, no factor $a\bar{b}$ or $\bar{a}b$ with distinct letters a and b can occur in a walk. But a word like $ba\overline{a}\overline{c}$ with distinct a, b and c, is still not a walk. Indeed, walks may be defined by contraposition.

Formally, let $\perp$ be the set of all words $v \in (A+\overline{A})^*$ such that there exists a word $u \preceq v$, and some distinct letters a and b, such that either $a\bar{b}$ or $\bar{a}b$ is a factor of u. A (linear) walk is any word $u \in (A+\overline{A})^*$ such that $u \notin \perp$. Clearly, the set $\perp$ is closed by product with arbitrary elements of $(A+\overline{A})^*$. It is thus an ideal of $(A+\overline{A})^*$.

We define the *monoid of walks* W_A as the Rees quotient $(A+\overline{A})^*/\perp$. It is the monoid obtained from $(A+\overline{A})^*$ by merging all elements of $\perp$ into a zero, the undefined walk. Clearly, the set $\perp$ (hence the set $W_A - \perp$) is closed under the Wagner congruence.

Walks and tiles are related by the following lemma:

Lemma 5 *For every non-zero walk $w \in W_A$, there is a unique tile $\theta(w)$ of the form $\theta(w) = (u_1, u_2, u_3) \in T_A$ such that $\overline{u_1}u_1u_2u_3\overline{u_3} \simeq_W w$. Moreover, extending θ to a mapping from W_A to T_A by $\theta(\perp) = 0$, then θ is an onto monoid morphism.*

Proof. Unicity. Let $w \in W_A$ have two decompositions $w \simeq_W u = \overline{u_1}u_1u_2u_3\overline{u_3}$ and $w \simeq_W v = \overline{v_1}v_1v_2v_3\overline{v_3}$, where (u_1, u_2, u_3) and (v_1, v_2, v_3) are tiles. Since the Wagner congruence is compatible with the free group reduction, we have $red(u) = red(v)$, hence $u_2 = v_2$; and the walks $u_1u \simeq u_1u_2u_3\overline{u_3}$ and u_1v also satisfy $red(v_1u) = red(v_1v)$, i.e. $red(u_1u_2) = red(v_1u_2)$, hence $u_1 = v_1$. Symmetrically, $u_3 = v_3$.

Inversibility. If $\theta(w)$ is defined, then $(\theta(w))^{-1} = \theta(\overline{w})$: indeed, if $\theta(w) = (u_1, u_2, u_3)$, then

$$\begin{aligned}
\overline{(u_1u_2)}(u_1u_2)\overline{u_2}(u_2u_3)\overline{(u_2u_3)} &\simeq_W \overline{(u_1u_2)}u_1u_2u_3\overline{u_3}\,\overline{u_2} \\
&\simeq_W u_3\overline{u_3}\,\overline{(u_1u_2)}u_1u_2\overline{u_2} \\
&= u_3\overline{u_3}\,\overline{u_2}\,\overline{u_1}u_1u_2\overline{u_2} \\
&\simeq_W u_3\overline{u_3}\,\overline{u_2}u_2\overline{u_2}\,\overline{u_1}u_1 \\
&\simeq_W u_3\overline{u_3}\,\overline{u_2}\,\overline{u_1}u_1 \\
&\simeq_W \overline{w}
\end{aligned}$$

Existence. Let $w \in W_A$. The existence of $\theta(w)$, or equivalently $\theta(\overline{w})$, is proved by induction over the number $n(w)$ of turns (alternations of positive and negative letters) in w.

When $n(w) = 0$, either $w \in A^*$ and obviously $(1, w, 1) = \theta(w)$, or $w \in \overline{A}^*$ and $(\overline{w}, w, \overline{w}) = \theta(w)$.

Assume $n(w) > 0$. We have $w = xw'$ for some $x \in A^* + \overline{A}^*$ and $w' \in W_A$ such that $n(w') = n(w) - 1$. By the induction hypothesis, there exists a tile $\theta(w') = (u_1, u_2, u_3)$ such that $w' \simeq_W \overline{u_1}u_1u_2u_3\overline{u_3}$ and thus $w \simeq_W x\overline{u_1}u_1u_2u_3\overline{u_3}$. By symmetry (possibly taking $\overline{w}$ instead of w) we may assume that $u_2 \in A^*$. We prove that $\theta(w) = \theta(x).\theta(w')$:

- ▷ if $x \in A^*$, since $x\overline{u_1}$ is a walk, we have $x \vee_s u_1 \neq 0$. It follows that $(1, x, 1).\theta(w') = ((x \vee_s u_1)\overline{x}, xu_2, u_3)$ is a non-zero tile, and
  - either $u_1 = v_1x$ for some $v_1 \in A^*$: then $(1, x, 1).\theta(w') = (v_1, xu_2, u_3)$ and $w \simeq_W x\overline{(v_1x)}(v_1x)u_2u_3\overline{u_3} = x\overline{x}\,\overline{v_1}v_1xu_2u_3\overline{u_3}$; thus (by commutation of idempotents) $w \simeq_W \overline{v_1}v_1x\overline{x}xu_2u_3\overline{u_3} \simeq_W \overline{v_1}v_1xu_2u_3\overline{u_3}$;
  - or $x = yu_1$ for some $y \in A^*$: then $(1, x, 1).\theta(w') = (1, xu_2, u_3)$ and $w \simeq_W (yu_1)\overline{u_1}u_1u_2u_3\overline{u_3} \simeq_W yu_1u_2u_3\overline{u_3} = xu_2u_3\overline{u_3}$.
- ▷ if $x \in \overline{A}^*$, since $x\overline{u_1}u_1u_2u_3$ is a walk, we have $x \vee_p u_2u_3 \neq 0$. It follows that $(\overline{x}, x, \overline{x}).\theta(w') = (u_1\overline{x}, red(xu_2), red(\overline{u_2}(\overline{x} \vee_p u_2u_3)))$ is a non-zero tile, and
  - either $u_2 = \overline{x}v_2$ for some $v_2 \in A^*$: then $(\overline{x}, x, \overline{x}).\theta(w') = (u_1\overline{x}, v_2, u_3)$ and $w \simeq_W \overline{u_1\overline{x}}u_1\overline{x}v_2u_3\overline{u_3}$;
  - or $\overline{x} = u_2y$ and $u_3 = yv_3$ for some $y, v_3 \in A^*$: then $(\overline{x}, x, \overline{x}).\theta(w') = (u_1\overline{x}, \overline{y}, v_3)$ and $w \simeq_W \overline{x}\,\overline{u_1}u_1u_2yv_3\overline{u_3}$; replacing y with $y\overline{y}y$ we get $w \simeq_W \overline{(u_1\overline{x})}(u_1\overline{x})\overline{y}u_3\overline{u_3}$;
  - or $\overline{x} = u_2u_3y$ for some $y \in A^*$: then $(\overline{x}, x, \overline{x}).\theta(w') = (u_1\overline{x}, \overline{z}, z)$ with $z = u_3y$, and $w \simeq_W \overline{(u_2z)}\,\overline{u_1}u_1u_2u_3\overline{u_3}$; replacing $\overline{z}$ with $\overline{z}z\overline{z}$ and commuting $z\overline{z}$ and $\overline{(u_1u_2)}u_1u_2$, we get

    $$w \simeq_W \overline{z}\,\overline{u_2}\,\overline{u_1}u_1u_2z\overline{y}\,\overline{u_3}u_3\overline{u_3} \simeq_W \overline{u_1\overline{x}}(u_1\overline{x})\overline{z}$$

    and finally $w \simeq_W \overline{u_1\overline{x}}(u_1\overline{x})\overline{z}z\overline{z}$.

Compositionality. By the argument above, for any non-zero walk w, we have $\theta(w) = \theta(x_0).\theta(x_1).\ldots.\theta(x_{n(w)})$ where $x_0x_1\ldots x_{n(w)}$ is the decomposition of w into words of $A^*+\overline{A}^*$. Obviously, when x and $x' =$ are both in A^* or both in $\overline{A}^*$, $\theta(xx') = \theta(x).\theta(x')$. Since the product of tiles is associative, the identity

$\theta(ww') = \theta(w).\theta(w')$ (extending θ to 0) holds for all walks $w, w' \in W_A$. Thus θ is a monoid morphism (onto since for every non-zero tile $(u_1, u_2, u_3) \in T_A$ we have $(u_1, u_2, u_3) = \theta(\overline{u_1}u_1u_2u_3\overline{u_3})$). □

By the previous lemma, for all walks w_1 and w_2, we have $w_1 \simeq_W w_2$ if and only if $\theta(w_1) = \theta(w_2)$. Thus the Wagner congruence is the kernel of θ; since $\perp$ is closed under the Wagner congruence, it can also be seen as an ideal of the free inverse monoid $FIM(A)$, and thus $FIM(A)/\perp$ just corresponds to McAlister's original definition.

Corollary 6 *The monoid of tiles T_A, isomorphic to the quotient of walks $W_A/\simeq_W$ by the Wagner congruence, is also isomorphic to the McAlister monoid $FIM(A)/\perp$.*

The relationship between walks, tiles and elements of the free inverse monoid is summarized by the following commuting diagram:

$$\begin{array}{ccc} (A+\overline{A})^* & \overset{[\ ]_{\simeq_W}}{\twoheadrightarrow} & FIM(A) \\ \Big\downarrow{\scriptstyle /\perp} & & \Big\downarrow{\scriptstyle /\perp} \\ W_A & \overset{\theta}{\twoheadrightarrow} & T_A \end{array}$$

Remark. In some sense, Lemma 5 captures most of the combinatorial analysis of two-way automata runs made in [28, 5]. More precisely, according to Pécuchet's definition [28], a word bisection is any quadruple of words $((u_1, u_2), (v_1, v_2)) \in (A^* \times A^*) \times (A^* \times A^*)$ such that $u_1u_2 = v_1v_2$. One can check that the mapping that maps every non-zero tile $(u_1, u_2, u_3) \in T_A$ to the quadruple $((u_1, u_2u_3), (u_1u_2, u_3)) \in (A^* \times A^*) \times (A^* \times A^*)$ is a well-defined bijection from non-zero tiles to word bisections.

Note that although some connections with inverse semigroup theory were observed, the link with McAlister's monoids, defined in [26] but emphasized in [23], was left implicit in Pécuchet's or Birget's works. Here we are more interested in *what* two-way automata read than in *how* they perform readings, which was Pécuchet's and Birget's main interest. A similar observation could be made about the study of zig-zag codes [2, 6, 24].

3 Classes of definable languages of tiles

We define here various classes of definable languages of tiles (or walks), from the class REC of languages recognizable by finite monoids to the class MSO of languages definable by means of Monadic Second Order sentences, via the class *k-REG* of k-regular languages defined by an extended notion of Kleene's expressions.

3.1 Operations on languages of tiles

The monoid structure of T_A induces the following operations on languages of non zero tiles. For every subsets M and N of $T_A - 0$:

- ▷ sum: $M + N = \{u \in T_A : u \in M \vee u \in N\}$
- ▷ product: $M.N = \{u.v \in T_A - 0 : u \in M, v \in N\}$
- ▷ star: $M^* = \sum_{n \geq 0} M^n$ with $M = \{(1,1,1)\}$ and $M^{k+1} = M.M^k$ for every $k \in \mathbb{N}$.

The inverse monoid structure of T_A induces three more operations on languages of tiles:

- ▷ inverse: $M^{-1} = \{u \in T_A : u^{-1} \in M\}$
- ▷ idempotent projection: $M^E = M \cap E_A$
- ▷ left and right projections: $M^L = \{u^L \in T_A : u \in M\}$ and $M^R = \{u^R \in T_A : u \in M\}$

On purpose, we restrict to non-zero tiles. Still, the operations satisfy many usual properties of the operations on word languages (the same proofs apply). In particular:

Proposition 7 *For all M, N and $P \subseteq T_A - 0$:*

- ▷ $M \cdot (N + P) = M \cdot N + M \cdot P$ *and* $(M + N) \cdot P = M \cdot P + N \cdot P$
- ▷ $M^* \cdot N$ *is the least solution (with respect to inclusion) of the equation* $X = M \cdot X + N$.

The following identities, more specifically related to languages of tiles, are straightforward:

Proposition 8 *For all $M, N \subseteq T_A - 0$:*

- $(M + N)^{-1} = M^{-1} + N^{-1}$
- $(M \cdot N)^{-1} = N^{-1} \cdot M^{-1}$
- $(M^*)^{-1} = (M^{-1})^*$
- $(M^E)^{-1} = (M^{-1})^E = M^E$
- $(M^L)^{-1} = (M^{-1})^R$ *and* $(M^R)^{-1} = (M^{-1})^L$

3.2 MSO-definable languages

In formal language theory, definability in Monadic Second-Order Logic is a typical yardstick of expressiveness that can be defined independently of the underlying algebraic structures [35]. In this part, we quickly review how languages of tiles can be defined by means of MSO formulas and then we characterize these languages via the classical notion of regular languages of words.

Any tile $u \in T_A$ can be seen as a FO-structure on the signature $\{R_a\}_{a \in A}$ of binary relation symbols, extended with two constants *in* and *out* for the entry and exit points. For instance, the triple $u = (ba, aa, bb)$ depicted in the following picture

$$\begin{array}{ccccccccccccc} & & & & & & \downarrow & & & & & & \\ 0 & \xrightarrow{b} & 1 & \xrightarrow{a} & 2 & \xrightarrow{a} & 3 & \xrightarrow{a} & 4 & \xrightarrow{b} & 5 & \xrightarrow{b} & 6 \\ & & & & & & & & \downarrow & & & & \end{array}$$

can be seen as the FO-structure t_u with domain $dom(t_u) = \{0, 1, 2, 3, 4, 5, 6\}$, relations $R_a = \{(1, 2), (2, 3), (3, 4)\}$ and $R_b = \{(0, 1), (4, 5), (5, 6)\}$, and constants $in(t_u) = 2$ and $out(t_u) = 4$.

Then, a language $L \subseteq T_A$ is MSO-definable when there is a MSO formula of the form $\varphi(U, x, y)$ where U is a set variable and x and y are two FO-variables such that, for every $u \in T_A$, we have $u \in L$ if and only if $t_u \models \varphi_L(dom(t_u), in(t_u), out(t_u))$.

3.3 Closure properties

We prove here several closure properties of the class MSO of MSO-definable languages of tiles.

Theorem 9 (Robustness) *For all languages of non-zero tiles M and $N \subseteq T_A$, if M and N are MSO-definable then $M+N$, $M.N$, M^*, M^{-1}, M^E, M^L and M^R are also MSO-definable.*

Proof. Let $\varphi_L(U,x,y)$ and $\varphi_M(U,x,y)$ be two MSO formulas respectively defining L and M. Without loss of generality, we may assume that these formulas check that both x and y belong to U, and that, moreover, the set U is connected.

Sum: take $\varphi_{L+M}(U,x,y) = \varphi_L(U,x,y) \vee \varphi_M(U,x,y)$.

Product: take $\varphi_{L.M}(U,x,y) = \exists X \exists Y \exists z (U = X \cup Y) \wedge \varphi_L(X,x,z) \wedge \varphi_M(Y,z,y)$.

Inverse: take $\varphi_{L^{-1}}(U,x,y) = \varphi_L(U,y,x)$.

Idempotent projection: take $\varphi_{L^E}(U,x,y) = (x=y) \wedge \varphi_L(U,x,y)$.

Left and right projections: by application of Proposition 8 combining the formulas above for the product, inverse and idempotent projection.

Star: this case is the most delicate. To define L^*, the main idea is to consider the (MSO definable) reflexive and transitive closure $R^*(x,y)$ of the binary relation $R(x_1,x_2)$ defined by $\exists X, X \subseteq U \wedge \varphi_L(X,x_1,x_2)$. The formula $\varphi_{L^*}(U,x,y)$ must also check that the set U is completely covered by the domain of the subtiles that are defined when checking that $R^*(x,y)$ is true. As all these subdomains necessarily overlap via their connecting roots, it is sufficient to check that both extremities of the domain U, i.e. the leftmost vertex $\mathit{left}(U)$ and the rightmost vertex $\mathit{right}(U)$, belong to at least one of these sets X. But this is easily encoded by a disjunction of the three possible cases: extremities are reached in a single intermediate tile, the left extremity is reached first or the right extremity is reached first.

More precisely, we can define an MSO formula $\psi_L(U,X,x_1,x_2)$ checking that there is a non-empty finite sequence of k connected subtiles of the form $\{t_i\}_{i\in[1,k]}$ such that $t_i \models \varphi_L(\mathit{dom}(t_i), \mathit{in}(t_i), \mathit{out}(t_i))$ for every $1 \leq i \leq k$, with the connection property defined by $\mathit{dom}(t_k) = X$ and, for every $1 \leq i < k$, we have $x_1 = \mathit{in}(t_1)$, $x_2 = \mathit{ou}(t_k)$, $\mathit{out}(t_i) = \mathit{in}(t_{i+1})$ and $\mathit{dom}(t_i) \subseteq U$.

Then the expected formula $\varphi_{L^*}(U,x,y)$ is built as a disjunction: either U is a singleton, with $x=y$ (the unit tile belongs to L^*) or there exist three sets

X_1, X_2 (the domains of the considered tiles) and X (the domain of the last tile) with $x_1 \in X_1$ and $x_2 \in X_2$ (the intermediate output roots) such that the property $\psi_L(U, X_1, x, x_1) \wedge \psi_L(U, X_2, x_1, x_2) \wedge \psi_L(U, Xx_2, y)$ is satisfied with either $\mathit{left}(U) \in X_1$ and $\mathit{right}(U) \in X_2$ (the leftmost vertex is encountered first) or $\mathit{left}(U) \in X_2$ and $\mathit{right}(U) \in X_1$ (the rightmost vertex is encountered first). □

Corollary 10 *For all regular languages of words $L, C, R \subseteq A^*$, the language of tiles $L \times C \times R$ is MSO-definable.*

Proof. By Büchi-Elgot-Trakhtenbrot's theorem [35], any regular language of words is MSO-definable. A MSO formula $\varphi(U)$ defining a language of words X can be seen as a formula defining the language of tiles $\{1\} \times X \times \{1\}$. Thus X embedded in T_A is a MSO-definable language of tiles, and by the closure properties of MSO, so are $X^L = X \times \{1\} \times \{1\}$ and $X^R = \{1\} \times \{1\} \times X$. More generally, if L, C and R are regular languages of words, then their product $L \times C \times R = L^L \cdot C \cdot R^R$ is MSO-definable. □

3.4 A word congruence for languages of tiles

We aim at providing a simple characterization of MSO-definable languages of tiles. As tiles are essentially words additionally equipped with input and output roots, this is achieved via the following notion of word congruence induced by a language of tiles.

Given a language $L \subseteq T_A$ of non-zero tiles, we define the *word* congruence $\simeq_L$ induced by L as the property that in any tile, two words can replace each other without altering the membership to L.

Formally:

Definition. [Induced word congruence] Let $L \subseteq T_A$ be a language of tiles. For all $u_0, v_0 \in A^*$, we say that the word u_0 is equivalent to the word v_0 w.r.t. the tile language L, which is denoted by $u_0 \simeq_L v_0$, when for all w_1, w_2, w_3 and $w_4 \in A^*$, if $u = (w_1u_0w_2, w_3, w_4)$ and $v = (w_1v_0w_2, w_3, w_4)$, or if $u = (w_1, w_2u_0w_3, w_4)$ and $v = (w_1, w_2v_0w_3, w_4)$, or if $u = (w_1, w_2, w_3u_3w_4)$ and $v = (w_0, w_2, w_3v_3w_4)$ then $u \in L \Leftrightarrow v \in L$ and $u^{-1} \in L \Leftrightarrow v^{-1} \in L$.

By construction, the relation $\simeq_L$ is indeed a congruence in the free monoid A^*. Let then $[u]_L$ denote the congruence class of a word $u \in A^*$.

Theorem 11 (Word congruence property) *For every language $L \subseteq T_A$ of non-zero tiles:*

$$\begin{aligned} L &= \Sigma_{(u_1,u_2,u_3)\in L\cap T_A^+}[u_1]_L \times [u_2]_L \times [u_3]_L \\ &\quad +\Sigma_{(u_1,u_2,u_3)^{-1}\in L\cap T_A^-}([u_1]_L \times [u_2]_L \times [u_3]_L)^{-1} \end{aligned}$$

Moreover, we have that L is definable in MSO if and only if $\simeq_L$ is of finite index.

Proof. The expression of L is an immediate consequence of the definition of $\simeq_L$.

By Myhill-Nerode's theorem, the word language $[w]_L \subseteq A^*$ is regular for every $w \in A^*$. By Corollary 10, for every u_1, u_2 and $u_3 \in A^*$, the three languages $L_1 = [u_1]_L \times \{1\} \times \{1\}$, $L_2 = \{1\} \times [u_2]_L \times \{1\}$ and $L_3 = \{1\} \times \{1\} \times [u_3]_L$ are also MSO-definable; and by the closure properties of MSO (Theorem 9), the language $L_1 \cdot L_2 \cdot L_3 = [u_1]_L \times [u_2]_L \times [u_3]_L$ and its inverse image are MSO-definable. If $\simeq_L$ is of finite index, then L is a finite sum of MSO-definable languages, thus by Theorem 9 L itself is MSO-definable.

Conversely, assume that L is MSO-definable. Given $L^+ = L \cap T_A^+$ and $L^- = L \cap T_A^-$, we observe that both L^+ and $(L^-)^{-1} \subseteq A^* \times A^* \times A^*$ can be encoded into languages of words M^+ and $M^- \subseteq A_L^* A_C^* A_R^*$ where A_L, A_C and A_R are three disjoint copies of the alphabet A used to encode the left, center and right elements of a tile.

Now, since L is definable in MSO, so are L^+ and L^- and thus, their encodings M^+ and M^- are also definable in MSO. By Büchi-Elgot-Trakhtenbrot's theorem, both languages M^+ and M^- are regular, and thus their syntactic congruences $\simeq_{M^+}$ and $\simeq_{M^-}$ are of finite index. This implies that $\simeq_L$ is also of finite index. Indeed, for all words $u, v \in A^*$, we have $u \simeq_L v$ if and only if $u_X \simeq_{M^+} v_X$ and $u_X \simeq_{M^-} v_X$ where X denotes either of L, C and R, and the word $w_X \in A_X^*$ is the re-encoding of any word $w \in A^*$ in the alphabet A_X. □

As an immediate corollary:

Theorem 12 (Simplicity) *A language $L \subseteq T_A$ of non-zero tiles is MSO-definable if and only if L is a finite sum of languages of the form $L \times C \times R$ or $(L \times C \times R)^{-1}$, where L, C and R are regular languages of words.*

3.5 Regular and k-regular languages

Definition. [k-regular languages of tiles] A language $M \subseteq T_A$ of non zero tiles is *regular* if it can be defined as the result of finitely many additions, multiplications and star operations of finite languages of non-zero tiles. The class of regular languages of tiles is denoted by *REG*.

For every $k \in \mathbb{N}$, a language $M \subseteq T_A$ is *k-regular* if either $k = 0$ and M is regular, or $k = k' + 1$ and M can be defined as the result of finitely many additions, multiplications, star and inverse operations over k'-regular languages and idempotent projections of k'-regular languages. The class of k-regular languages is denoted by *k-REG*.

Remark. The set of positive tiles is regular: $T_A^+ = (A^{-1}.A + A + A.A^{-}1)^*$ (where A is embedded in T_A). Thus the set of negative tiles $T_A^- = (T_A^+)^{-1}$ is regular, and the set $E_A = (T_A^+ + T_A^-)^E$ of idempotent tiles is 1-regular. We prove later (Proposition 23) that E_A is not regular.

The following fact is well-known in the theory of inverse monoids:

Proposition 13 *For every $k \in \mathbb{N}$, the class k-REG of k-regular tile languages is closed under the inverse operation.*

Proof. This follows from Proposition 8: the inverse operation commutes with the sum, product, star and idempotent projection. □

Theorem 14 *For every $k > 0$, we have k-REG $= MSO =$ 1-REG.*

Proof. By construction *1-REG* $\subseteq$ *2-REG* $\subseteq \cdots \subseteq$ *k-REG* $\subseteq \cdots$ and by the Robustness Theorem 9, we have *k-REG* $\subseteq MSO$ for every $k \in \mathbb{N}$.

Every regular language of words X, embedded in T^A, is a regular language of tiles (defined by the same regular expression), and satisfies the properties $X^L = (X^{-1} \cdot X)^E$ and $X^R = (X \cdot X^{-1})^E$. It follows that if L, C and R are regular languages of words, then $(L \times C \times R) = (L^{-1} \cdot L)^E \cdot C \cdot (R \cdot R^{-1})^E$ and its inverse $(L \times C \times R)^{-1}$ are 1-regular. By the Simplicity Theorem 12, every MSO-definable language is a finite sum of such 1-regular languages, thus $MSO \subseteq$ *1-REG*. □

3.6 Recognizable languages of tiles

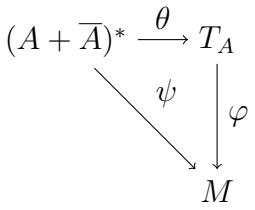
Here we consider the algebraic notion of recognizability. The usual notions on words may be transposed to tiles: say that a language of non-zero tiles

$L \subseteq T_A$ is *recognizable* if there exists a monoid M, a monoid morphism $\varphi : T_A \to M$, and a finite subset F of M, such that $L = \varphi^{-1}(F)$; or, equivalently, if the syntactic congruence defined by $u \sim_L v \iff \forall x, y \in T_A$ $(x.u.y \in L \Leftrightarrow x.v.y \in L)$ is of finite index. The class of recognizable languages of tiles is denoted by *REC*.

The following characterization is well-known (see [34]):

Theorem 15 *A language L of non-zero tiles is recognizable if and only if $\theta^{-1}(L)$ is a regular language of words.*

Proof.If: let $W = \theta^{-1}(L)$, and let $\sim_W$ denote the syntactic congruence of W in $(A + \overline{A})^*$, extended to W_A by $w \sim_W \perp \Leftrightarrow w = \perp$.
For all $u, v \in W_A$, we have $u \sim_W v \Rightarrow \theta(u) \sim_L \theta(v)$: indeed, since θ is onto, we have $\theta(W) = L$ and for all $x, y \in T_A$ there exist $\alpha, \beta \in W_A$ such that $\theta(\alpha) = x$ and $\theta(\beta) = y$; if $u \sim_W v$ we have $x.\theta(u).y \in L \Rightarrow \theta(\alpha u \beta) \in L \Rightarrow \alpha u \beta \in W \Rightarrow \alpha v \beta \in W \Rightarrow x.\theta(v).y \in L$ and by symmetry, we have $x.\theta(v).y \in L \Rightarrow x.\theta(u).y \in L$.
Thus $[u]_{\sim_w} \mapsto [\theta(u)]_{\sim_L}$ is a well-defined mapping of $W_A/\sim_W$ onto $T_A/\sim_L$. If W is a regular language of words then $W_A/\sim_W$ is finite and thus $\sim_L$ is of finite index.
Only if: let $\varphi : T_A \to M$ be a monoid morphism, let F a finite subset of M and let $L = \varphi^{-1}(F)$. Then $\psi = \varphi \circ \theta$ is a monoid morphism, and $\theta^{-1}(L) = \psi^{-1}(F)$ is recognized (as a language of words) by the monoid M and the morphism ψ since the following diagram commutes.



□

Corollary 16 *$REC \subseteq REG$.*

Proof. Since θ is an onto monoid morphism, any regular expression of $\theta^{-1}(L)$ yields a regular expression of L. □

But not all regular languages are recognizable as proved in the following proposition.

Proposition 17 *The regular tile language $L = (b \cdot a^*)^{-1} \cdot b \cdot a^*$ is not recognizable.*

Proof. L is the set of tiles of the form $(ba^m, a^{n-m}, a^{max(m-n,0)})$ with $m, n \in \mathbb{N}$, i.e. $L = ba^* \times a^* \times \{1\}$. We show that the syntactic congruence $\sim_L$ is of infinite index: for all $m, n \in \mathbb{N}$, let $u_m = (ba^m, 1, 1)$ and $v_k = (a^k, 1, 1)$. Then $u_m.v_k \in L$ if and only if $k \leq m$. If $u_m \sim_L u_n$ for some $m, n \in \mathbb{N}$, then for every $k \in \mathbb{N}$, $k \leq m$ if and only if $k \leq n$, hence $m = n$. □

Though some simple regular languages are not recognizable, the class *REC* does contain non-trivial languages of tiles. In [13], it is shown that recognizable languages are strongly related with bi-infinite periodic words. We just give an example, defined from the bi-infinite word ${}^{\omega}(ab)(ab)^{\omega}$.

Let $M = \{0, 1, (a, 1, b), (b, 1, a), (b, a, b), (a, b, a)\}$, let $\odot$ be the product defined over M by the following table (with 1 neutral and 0 absorbing):

$\odot$	$(a,1,b)$	$(b,1,a)$	(b,a,b)	(a,b,a)
$(a,1,b)$	$(a,1,b)$	0	0	(a,b,a)
$(b,1,a)$	0	$(b,1,a)$	(b,a,b)	0
(b,a,b)	(b,a,b)	0	0	$(b,1,a)$
(a,b,a)	0	(a,b,a)	$(a,1,b)$	0

Proposition 18 *$(M, \odot)$ is an inverse monoid.*

Proof. One can check that the product $\odot$ is associative, hence M is a monoid. In the set $E(M) = \{0, 1, (a, 1, b), (b, 1, a)\}$ of idempotents, the commutation follows from the unique non-trivial case $(a, 1, b) \odot (b, 1, a) = (b, 1, a) \odot (a, 1, b) = 0$. Finally, we check that $(a, b, a) \odot (b, a, b) \odot (a, b, a) = (a, b, a)$ and $(b, a, b) \odot (a, b, a) \odot (b, a, b) = (b, a, b)$. It follows that $(a, b, a)^{-1} = (b, a, b)$ and $(b, a, b)^{-1} = (a, b, a)$. Any other element is idempotent and thus self-inverse. □

Let $\varphi : T_A \to M$ be defined by $\varphi(0) = 0$, by $\varphi(1) = 1$, and for every $(u, v, w) \in T_A$ such that $uvw \neq 1$, by $\varphi(u, v, w) = 0$ if uvw is not a factor of $(ab)^{\omega}$ and, otherwise, when u is a positive tile by:

1. $\varphi(u, v, w) = (a, 1, b)$ when $|v|$ is even with $a \leq_s u$,$b \leq_p v$, $a \leq_s v$ or $b \leq_p w$,

2. $\varphi(u, v, w) = (b, 1, a)$ when $|v|$ is even with $b \leq_s u$,$a \leq_p v$, $b \leq_s v$ or $a \leq_p w$,

3. $\varphi(u,v,w) = (b,a,b)$ when $|v|$ is odd with $a \leq_p v$,

4. $\varphi(u,v,w) = (a,b,a)$ when $|v|$ is odd with $b \leq_p v$,

and $\varphi(u,v,w) = (\varphi(uv, \overline{v}, vw))^{-1}$ when (u,v,w) is a negative tile.

Proposition 19 *The mapping $\varphi : T_A \to M$ is an onto morphism.*

Proof. Indeed, for all u and $v \in T_A$, we have $\varphi(u) \odot \varphi(v) = \varphi(u.v) = \varphi(\varphi(u).\varphi(v))$. □

Let $L_S = (ab)^* + b(ab)^*$, $L_C = (ab)^*$, and $L_P = (ab)^* + (ab)^*a$. By the previous proposition, the non-trivial tile language $L_S \times L_C \times L_P - 1$ is recognizable, since it is precisely $\varphi^{-1}((b,1,a))$.

4 Two-way automata and languages of tiles

We prove here that regular languages of tiles are just the languages recognizable by finite two-way automata.

4.1 Two-way automata

A finite-state two-way automaton (or 2WA for short) on an alphabet A is a quadruple $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ with a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition table $\Delta : (A + \overline{A}) \to \mathcal{P}(Q \times Q)$.

A *run* of $\mathcal{A}$ over a string of $(A + \overline{A})^*$ is a finite sequence:

$$\rho = q_0 a_1 q_1 \dots q_{n-1} a_n q_n$$

where $n \geq 0$, $q_0, \dots, q_n \in Q$ and $a_1, \dots, a_n \in A + \overline{A}$, such that for every $1 \leq i \leq n$, we have $(q_{i-1}, q_i) \in \Delta(a_i)$.

The run ρ is *accepting* if $q_0 \in I$ and $q_n \in F$. In that case, we say that the associated string $s_\rho = a_1 \cdots a_n \in A^*$ is accepted by the automaton $\mathcal{A}$ (seeing $\mathcal{A}$ as a standard one-way automaton on the alphabet $A + \overline{A}$). The language of strings accepted by $\mathcal{A}$ is written $S(\mathcal{A})$.

We say that ρ is a run over a walk when, additionally, the string s_ρ is a walk. In that case, we write w_ρ for the string s_ρ. The set of walks accepted by $\mathcal{A}$ is written $W(\mathcal{A})$. In other words, $W(\mathcal{A}) = S(\mathcal{A}) \cap W_A$.

Remark. Two-way automata may be defined with the possibility to stand still at the same position while changing state. Clearly, these "silent" transitions are just syntactic sugar that can be replaced with extra non-silent transitions (by the same techniques used to eliminate classical silent transitions in one-way automata). Our definition follows [9].

The following lemma emphasizes the difference between two-way automata (interpreted on walks) and ordinary finite-state automata (interpreted on strings):

Proposition 20 *There exists a two-way automaton $\mathcal{A}$ such that $W(\mathcal{A})$ is not a regular language.*

Proof. Assume $A = \{a, b\}$ with two distinct letters, let $Q = \{q_0, q_1, q_2, q_3\}$ be a four-state set, and let $\mathcal{A} = \{Q, \{q_0\}, \{q_3\}, \Delta\}$ with transition function Δ defined by $\Delta(a) = \{(q_0, q_1)\}$, $\Delta(b) = \{(q_1, q_1)\}$, by $\Delta(\overline{b}) = \{(q_1, q_2), (q_2, q_2)\}$ and by $\Delta(\overline{a}) = \{q_2, q_3\}$.

We have $S(\mathcal{A}) = ab^+\overline{b}^+\overline{a}$ and thus $W(\mathcal{A}) = \{ab^n\overline{b}^n\overline{a} : n > 0\}$ that is not regular. □

4.2 Word and tile languages recognized by 2WA

The *language of words* $L(\mathcal{A})$ recognized by a two-way automaton $\mathcal{A}$ on the alphabet A (completed by $\overline{A}$) is the set of *words* $u \in A^*$ such that there is an accepting run of $\mathcal{A}$ corresponding to a back-and-forth reading of u, i.e. a walk $w \in \mathcal{A}$ such that $w \simeq_W u$.

In a similar way, we define the *language of tiles* $T(\mathcal{A})$ recognized by a two-way automaton $\mathcal{A}$ as the set of tiles $u = (u_1, u_2, u_3) \in T_A$ such that there is an accepting run of $\mathcal{A}$ corresponding to a back-and-forth reading of $\overline{u_1}u_1u_2u_3\overline{u_3}$: in other words, we have $T(\mathcal{A}) = \theta(W(\mathcal{A}))$.

Remark. The characterization of algebraically recognizable languages of tiles given in Theorem15 may be interpreted in terms of automata: a language L of tiles is algebraically recognizable if and only if there is a two-way automaton $\mathcal{A}$ such that $S(\mathcal{A}) = \theta^{-1}(L)$, i.e. the accepting runs of $\mathcal{A}$ are *all* possible back-and-forth readings over the domain of a tile of L, starting at the entry point of u and ending at its exit point.

We now prove a Kleene theorem for tile languages.

Theorem 21 *The regular tile languages are exactly the tile languages recognizable by finite-state two-way automata.*

Proof. This follows from Lemmas 22 and 25 below. □

Lemma 22 *Every regular language of tiles is recognized by some finite-state two-way automaton.*

Proof. For any finite tile language $M \subseteq T_A$, any one-way automaton over $A+\overline{A}$ recognizing the finite word language $\{\overline{u_1}u_1u_2u_3\overline{u_3} : (u_1, u_2, u_3) \in M\}$ can be viewed as a two-way automaton recognizing M. Thus all finite languages of non-zero tiles are recognizable.

More generally, any finite-state one-way automaton $\mathcal{A}$ over the alphabet $A + \overline{A}$, recognizing a word language $L \subseteq (A + \overline{A})^*$, can be viewed as a 2WA recognizing the tile language $\theta(L \cap W_A)$.

For all word languages $L, M \subseteq (A + \overline{A})^*$ we have

- $\theta((L + M) \cap W_A) = \theta(L \cap W_A) + \theta(M \cap W_A)$ (obvious)
- $\theta(LM \cap W_A) = \theta(L \cap W_A).\theta(M \cap W_A)$ (from Lemma 5)
- $\theta(L^* \cap W_A) = (\theta(L \cap W_A))^*$ (from Lemma 5).

We conclude the proof by induction on the structure of regular expressions. □

Lemma 22 yields a pumping argument to prove the following:

Proposition 23 *The set E_A of idempotent tiles is not regular.*

Proof. Let $\mathcal{A}$ be a finite-state 2WA such that $E_A \subseteq T(\mathcal{A})$, and let $a \in A$. For any $n \in \mathbb{N}$, since $(a^n, 1, 1) \in E_A$, there is in $\mathcal{A}$ an accepting run over a word $u \simeq_W \overline{a}^n a^n$; this run may be split into $\rho_n\rho'_n$ where the run ρ_n (resp. ρ'_n) is over a prefix v_n (resp. a suffix v'_n) of u such that $red(v_n) = \overline{a}^n$ and $red(v'_n) = a^n$. Since $\mathcal{A}$ has a finite number of states, there are $m, n \in \mathbb{N}$ such that $m < n$ and that the runs ρ_m and ρ_n end in the same state. Then $\rho_m\rho'_n$ is an accepting run over $v_mv'_n$, thus $\theta(v_mv'_n) \in T(\mathcal{A})$, but $\theta(v_mv'_n)$ is not an idempotent tile since $red(v_mv'_n) = a^{n-m}$. It follows that no finite-state 2WA recognizes E_A. □

Let $\mathcal{A} = \langle Q, I, F, \Delta\rangle$ be a two-way automaton on the alphabet A. For every pair of states $(p, q) \in Q \times Q$, let $T_{p,q}$ denote the language of tiles recognized by the two-way automaton $\mathcal{A}_{p,q} = \langle Q, \{q\}, \{p\}, \Delta\rangle$.

Proposition 24 *The sets $T_{p,q}$ with p and $q \in Q$ are the least solution (w.r.t. inclusion order) of the system of equations*

$$T_{p,q} = \delta_{p,q} + \sum_{a \in A+\overline{A}} \sum_{(p,r) \in \Delta(a)} \theta(a).T_{r,q} \qquad (E_{p,q})$$

where $\delta_{p,q} = \{1\}$ if $p = q$ and $\emptyset$ otherwise.

Proof. For all states p and q, and every integer $k \geq 0$, let $W^k_{p,q}$ denote the set of walks of length at most k accepted by $\mathcal{A}_{p,q}$. The identities $W_{p,q} = \delta_{p,q}$ and

$$W^{k+1}_{p,q} = W^k_{p,q} + \sum_{a \in A+\overline{A}} \sum_{(p,r) \in \Delta(a)} (aW^k_{r,q} \cap W_A)$$

immediately follow from the definition of the walk acceptance. Then, with $W_{p,q} = \sum_{k \in \mathbb{N}} W^k_{p,q}$, by Tarski's fixpoint theorem, the sets $W_{p,q}$ form the least fixed point of the system of equations:

$$W_{p,q} = \delta_{p,q} + \sum_{a \in A+\overline{A}} \sum_{(p,r) \in \Delta(a)} (aW_{r,q} \cap W_A) \qquad (E'_{p,q})$$

We conclude the proof by applying θ to the equations. □

Lemma 25 *$T(\mathcal{A})$ is a regular tile language.*

Proof. The least solution of the system of equations $E_{p,q}$ can be computed by a Gaussian elimination of variables, since the least solution of an equation $X = U.X + V$ in $\mathcal{P}(T_A)$ is $U^*.V$. This gives a regular expression of every $T_{p,q}$ and thus for $T(\mathcal{A})$ as well since $T(\mathcal{A}) = \Sigma_{(p,q) \in I \times F} T_{p,q}$. □

Corollary 26 (Shepherdson's theorem) *Every language of words recognizable by a two-way automaton is regular.*

Proof. Let $\mathcal{A}$ be a finite two-way automaton. The *word* language recognized by $\mathcal{A}$ is $L(\mathcal{A}) = \{u \in A^* : (1, u, 1) \in T(\mathcal{A})\}$. Now let $\#$ be a new letter. By Lemma 25, the *tile* language $\#.T(\mathcal{A}).\#$ is regular, thus MSO-definable by Theorem 14. However, since $\# \notin A$, any tile in $\#.T(\mathcal{A}).\#$ must have empty contexts, i.e. $\#.T(\mathcal{A}).\# \subseteq \{1\} \times \#A^*\# \times \{1\}$. It follows that $\#.T(\mathcal{A}).\# = \{1\} \times \#L(\mathcal{A})\# \times \{1\}$. By Theorem 12, we know that $\#L(\mathcal{A})\#$ is a regular language of words, thus the language $L(\mathcal{A})$ is also regular. □

4.3 Many-invisible pebble automata

Finite-state two-way automata may be extended with a pebble-handling mechanism. Here we consider *invisible* pebbles in the sense of [7]: at any moment, only the last pebble left may be observed by the automaton, and this observation can only be done by removing the pebble. Also, as we will not allow automata with an unbounded number of pebbles, the pebble we use are unmarked.

This makes our k-(invisible, unmarked)-pebble automata presumably less expressive than the classical k-(visible,marked)-pebble automata [9, 8], whose transitions are also governed by the presence (or absence) of pebbles on the current node. The more general case of infinitely many invisible (marked) pebbles is considered in [15] when studying walking automata on birooted trees.

Formally, a finite-state pebble 2-way automaton (or P2A for short) on an alphabet A is a quadruple $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ with a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, and a transition table $\Delta : (A + \overline{A} + \{1^+, 1^-\}) \to \mathcal{P}(Q \times Q)$. For every $a \in A + \overline{A}$, the set of transitions $\Delta(a)$ tells how the letter a can be read as in a two-way automaton. Newly, the set of transitions $\Delta(1^+)$ tells how a pebble can be left on the current position and the set transitions $\Delta(1^-)$ tells how a pebble can be removed. In other words, a pebble automaton is a two-way automaton that has the capacity, from time to time, to leave and remove pebbles placed *between* letters of the underlying word.

The run of such a P2A automaton is then defined as follows. A *position configuration* is a non-empty finite sequence of (positive or negative) integers $p = n_0.\cdots.n_k \in \mathbb{Z}^+$. The intended meaning of position configuration p is that n_i records the relative number of letters (positive or negative) read from the ith pebble left on the input word, with the initial starting point modeled as a sort of a 0th pebble.

Moreover, since any non-zero pebble is eventually removed in a run, we only record the position relative to the last pebble left. Pushing a pebble on the stack freezes the previous recorded relative positions. It follows that, at any time, we have $n_i + n_{i+1} + \cdots + n_k$ will denote the number of positive (or negative) letters that separate the automaton head from the position of the ith pebble. In particular, when $n_k = 0$, the kth pebble can be removed.

Formally, a *run* of the pebble automaton $\mathcal{A}$ is a finite word $\rho \in ((Q \times \mathbb{Z}^+).(A + \overline{A} + 1))^*.(Q \times \mathbb{Z}^+)$ such that, for every factor of ρ of the form

$(q,p).a.(q',p')$, with $a \in A+\overline{A}+1$, one of the following conditions is satisfied:

- $(q,q') \in \Delta(a)$, $a \in A+\overline{A}$, $p = p''.x$ for some possibly empty sequence of integers p'' and some $x \in \mathbb{Z}$ and $p' = p''.(x+\delta_a)$ with $\delta_a = 1$ if $a \in A$ and $\delta_a = -1$ if $a \in \overline{A}$,
- $(q,q') \in \Delta(1^+)$, $a = 1$ and $p' = p.0$,
- $(q,q') \in \Delta(1^-)$, $a = 1$ and $p = p'.0$,

As before, we also assume that the projection w_ρ of ρ to $(A+\overline{A})^*$ is a non-zero walk.

We observe that the position configurations are handled as a (left to right) stack. Leaving a pebble amounts to pushing 0, the new relative position of the head from that pebble. Reading $a \in A+\overline{A}$, the relative position from the last left pebble is changed by δ_a. Removing a pebble amounts to popping the last relative position of the head from that pebble. This relative position is forced to 0. This way, we model the fact that the head must have moved back to the position where the pebble has been left.

The number of pebbles used in a run $\rho \in ((Q\times\mathbb{Z}^+).(A+\overline{A}))^*.(Q\times\mathbb{Z}^+)$ is defined as the least integer $k \in \mathbb{N}$ such that $\rho \in ((Q\times\mathbb{Z}^{=k+1}).(A+\overline{A}))^*.(Q\times\mathbb{Z}^{=k+1})$ where $\mathbb{Z}^{=k}$ stands for the sequences of integers of length at most k.

Still writing w_q for the projection of run ρ on the alphabet $A+\overline{A}$, we say that a triple $u = (u_1,u_2,u_3)$ is accepted by automaton $\mathcal{A}$ with at most k-pebble when there exists run ρ using at most k pebbles such that $w_\rho \simeq_W \overline{u_1}u_1u_2u_3\overline{u_3}$ with start state of the form $(q,0)$ with $q \in I$ and end state of the form (q',i) with $q' \in F$. A simple check of our definition shows that, in that case $i = |u_2|$ when $u_2 \in A^*$ and $i = -|u_2|$ when $u_2 \in \overline{A}^*$.

4.4 Pebbles vs. tile idempotent operators

From now on, a k-pebble automaton is defined as a many-invisible pebble automaton whose runs are allowed to use at most k pebbles.

Theorem 27 *For every $k \in \mathbb{N}$, the k-regular tile languages are exactly the tile languages recognizable by finite-state k-pebble automata.*

Proof. Follows from the Lemmas 28 and 29 below. □

Lemma 28 *Every language of tiles recognized by a finite k-pebble automaton is k-regular.*

Proof. Let $\mathcal{A} = \langle Q, I, F, \delta \rangle$ be a finite many-pebble automaton. For every pair of states $(p, q) \in Q \times Q$, every integer $k \geq 0$, let $T^k_{p,q} \subseteq T_A$ be the language of tiles recognized by $\mathcal{A}$ with at most k-pebbles from state p to state q. Let also $C^k_{p,q}$ be the associated set of idempotent tiles defined by $C^k_{p,q} = T^k_{p,q} \cap E_A = (T^k_{p,q})^E$.

First, one can prove as in the previous section that sets of triples $T^k_{p,q}$ form the least solution of the set of equations defined, for every p and $q \in Q$, by $T^k_{p,q} = T_{p,q}$ as before and, for every $k \geq 0$, by:

$$\begin{aligned} T^{k+1}_{p,q} &= \delta_{p,q} + \sum_{a \in A + \overline{A}} \sum_{(p,r) \in \Delta(a)} \theta(a).T^{k+1}_{r,q} \\ &\quad + \sum_{(p,p') \in \delta(1^+)} \sum_{(r',r) \in \delta(1^-)} C^k_{p',r'}.T^{k+1}_{r,q}(\mathcal{A}) \end{aligned}$$

Indeed, we just mimic in these equations all the possible cases to build a run. Either some letter $a \in A + \overline{A}$ is red, or a pebble is used. Of course, we check that $T^k_{p,q}$ only depends on languages of the form $T^{k'}_{p',q'}$ with $k' \leq k$, or $C^{k'}_{p',q'} = (T^{k'}_{p,q})^E$ with $k' < k$. That later case shows in particular that no circular dependency involves projections on idempotent tiles. It follows that this system can be solved by induction on $k \in \mathbb{N}$ by Gaussian elimination of variables. Then, for every $k \in \mathbb{N}$, given the k-regular expressions defining languages $T^k_{p,q}$s we conclude by taking $T^k(\mathcal{A}) = \sum_{(p,q) \in I \times F} T^k_{p,q}$. □

Lemma 29 *Every k-regular language of tiles is recognized by some finite k-pebble automaton.*

Proof. As for regular languages of tiles (proof of Lemma 22), we proceed by induction on the syntactic complexity of k-regular expressions, combining many-pebble automata. We just detail the construction for the idempotent projection. Given a finite automaton $\mathcal{A} = \langle Q, I, F, \Delta \rangle$ k-recognizing a language $T = T^k(\mathcal{A})$, we define $\mathcal{A}' = \langle Q', I', F', \Delta' \rangle$ by $Q' = Q \cup \{q_0, q_f\}$ with q_0 and q_f two new states with $I' = \{q_0\}$, $F' = \{q_f\}$ and, for every $a \in A + \overline{A}$, $\Delta'(a) = \Delta(a)$, $\Delta'(1^+) = \Delta(1^+) \cup (\{q_0\} \times I)$ and $\Delta'(1^-) = \Delta(1^-) \cup (F \times \{q_f\})$. It is straightforward to check that $\mathcal{A}'$ recognizes the language $T^E = T^k(\mathcal{A}) \cap E_A$, i.e. $T^{k+1}(\mathcal{A}') = T^E$ with at most $k+1$ pebbles. □

Corollary 30 *Every language of words recognized by a k-pebble automaton is regular.*

Proof. Like the proof of Corollary 26 (Shepherdson's theorem). Let $\mathcal{A}$ be a finite k-pebble automaton. The language of words recognized by $\mathcal{A}$ is $L(\mathcal{A}) = \{u \in A^* : (1, u, 1) \in T^k(\mathcal{A})\}$. Given $\# \notin A$, the language $\#.T^k(\mathcal{A}).\#$ is k-regular, and since $\#.T^k(\mathcal{A}).\# = \{1\} \times \#L(\mathcal{A})\# \times \{1\}$, we conclude that $L(\mathcal{A})$ is regular. □

Conclusion and further works

Studying languages of overlapping tiles, equivalently subsets of McAlister monoids, we have considered several classes of languages: recognizable languages, regular languages, k-regular languages and MSO-definable languages, obtaining a strict though finite hierarchy

$$REC \subsetneq REG = \textit{0-REG} \subsetneq \textit{1-REG} = \textit{(k+1)-REG} = MSO$$

for every $k \in \mathbb{N}$. An intriguing related class of languages of tiles is the class $BOOL(REG)$ of finite boolean combinations of regular languages. It is obviously included in the class of MSO-definable languages, but it is by no means clear whether the inclusion is strict. Another further work would be to relate the hierarchy with classes of algebraically recognizable languages of tiles, as defined in [10, 14].

References

[1] F. S. Almeida. *Algebraic Aspects of Tiling Semigroups*. PhD Thesis, Universidade de Lisboa, Faculdade de Ciências Departamento de Matemática, Lisboa, Portugal, 2010.

[2] M. Anselmo. Automates et code zigzag. *ITA*, 25:49–66, 1991.

[3] F. Berthaut, D. Janin, and M. DeSainteCatherine. libTuile : un moteur d'exécution multi-échelle de processus musicaux hiérarchisés. In *Actes des Journées d'Informatique Musicale (JIM 2013)*, 2013.

[4] 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, 2012.

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

[6] Do Long Van, B. LeSaëc, and I. Litovsky. On coding morphisms for zigzag codes. *ITA*, 26:565–580, 1992.

[7] 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.

[8] 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.

[9] N. Globerman and D. Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theoretical Comp. Science*, 169(2):161–184, 1996.

[10] D. Janin. Quasi-recognizable vs MSO definable languages of one-dimensional overlapping tiles. In *Mathematical Found. of Comp. Science (MFCS)*, volume 7464 of *LNCS*, pages 516–528, 2012.

[11] D. Janin. Vers une modélisation combinatoire des structures rythmiques simples de la musique. *Revue Francophone d'Informatique Musicale (RFIM)*, 2, 2012.

[12] D. Janin. Algebras, automata and logic for languages of labeled birooted trees. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 7966 of *LNCS*, pages 318–329. Springer, 2013.

[13] D. Janin. On languages of one-dimensional overlapping tiles. In *Int. Conf. on Current Thrends in Theo. and Prac. of Comp. Science (SOFSEM)*, volume 7741 of *LNCS*, pages 244–256, 2013.

[14] D. Janin. Overlaping tile automata. In *8th International Computer Science Symposium in Russia (CSR)*, volume 7913 of *LNCS*, pages 431–443. Springer, 2013.

[15] D. Janin. Walking automata in the free inverse monoid. Technical Report RR-1464-12 (revised april 2013), LaBRI, Université de Bordeaux, 2013.

[16] D. Janin, F. Berthaut, M. DeSainte-Catherine, Y. Orlarey, and S. Salvati. The T-calculus : towards a structured programming of (musical) time and space. Technical Report RR-1466-13, LaBRI, Université de Bordeaux, 2013.

[17] D. Janin, F. Berthaut, and M. DeSainteCatherine. Multi-scale design of interactive music systems : the libtuiles experiment. In *Sound and Music Computing (SMC)*, 2013.

[18] J. Kellendonk. The local structure of tilings and their integer group of coinvariants. *Comm. Math. Phys.*, 187:115–157, 1997.

[19] J. Kellendonk and M. V. Lawson. Tiling semigroups. *Journal of Algebra*, 224(1):140 – 150, 2000.

[20] J. Kellendonk and M. V. Lawson. Universal groups for point-sets and tilings. *Journal of Algebra*, 276:462–492, 2004.

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

[22] M. V. Lawson. *Inverse Semigroups : The theory of partial symmetries*. World Scientific, 1998.

[23] M. V. Lawson. McAlister semigroups. *Journal of Algebra*, 202(1):276 – 294, 1998.

[24] B. LeSaëc, I. Litovsky, and B. Patrou. A more efficient notion of zigzag stability. *ITA*, 30(3):181–194, 1996.

[25] S. W. Margolis and J.-E. Pin. Languages and inverse semigroups. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 172 of *LNCS*, pages 337–346. Springer, 1984.

[26] D.B. McAlister. Inverse semigroups which are separated over a subsemigroups. *Trans. Amer. Math. Soc.*, 182:85–117, 1973.

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

[28] J.-P. Pécuchet. Automates boustrophedon, semi-groupe de Birget et monoide inversif libre. *ITA*, 19(1):71–100, 1985.

[29] M. Pietrich. *Inverse semigroups*. Wiley, 1984.

[30] J-.E. Pin. Chap. 10. Syntactic semigroups. In *Handbook of formal languages, Vol. I*, pages 679–746. Springer-Verlag, 1997.

[31] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114 –125, 1959.

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

[33] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3:198–200, 1959.

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

[35] W. Thomas. Chap. 7. Languages, automata, and logic. In *Handbook of Formal Languages, Vol. III*, pages 389–455. Springer-Verlag, Berlin Heidelberg, 1997.

[36] M. Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, 30:261–264, 1989.