Series-parallel languages and the bounded-width property[1]

K. Lodaya
Institute of Mathematical Sciences
CIT Campus
Chennai 600 113 – India.
`kamal@imsc.ernet.in`
`http://www.imsc.ernet.in/~kamal`

P. Weil
LaBRI
Université Bordeaux 1 and CNRS
351 cours de la Libération
33405 Talence Cedex – France.
`Pascal.Weil@labri.u-bordeaux.fr`
`http://www.labri.u-bordeaux.fr/Annuaire/weil`

Résumé. Afin de modéliser la concurrence, nous étendons la théorie des automates des langages de mots habituels, aux langages d'ensembles partiellement ordonnés série-parallèles, ou *sp*-langages. Nous utilisons les propriétés algébriques de l'ensemble des ensembles série-parallèles pour étudier la notion de reconnaissabilité, basée sur les congruences d'indice fini, et une notion de régularité basée sur une nouvelle classe d'automates, appelés automates branchants. Les principaux résultats de cet article concernent les *sp*-langages de largeur bornée. Nous montrons que ces langages sont reconnaissables si et seulement si ils sont réguliers. De plus, un *sp*-langage est régulier et de largeur bornée si et seulement si il peut être décrit par une expression série-rationnelle, c'est-à-dire une expression utilisant seulement les lettres, les opérations de produit séquentiel et de produit parallèle, et l'itération séquentielle. Nous donnons enfin une caractérisation strictement algébrique de la propriété de largeur bornée, en termes d'algèbres série-parallèles nilpotentes en profondeur.

Abstract. In order to model concurrency, we extend automata theory from the usual languages of words to sets of labelled series-parallel posets, or *sp*-languages. We use the algebraic properties of the set of series-parallel posets to investigate the notion of recognizability, based on finite-index congruences, and a notion of regularity based on a new class of "branching" automata. The main results of the paper concern the *sp*-languages having bounded-width. We show that such languages are recognizable if and only if they are regular. In addition, the regular bounded-width *sp*-languages are exactly those which can be described by series-rational expressions, that is, expressions using the letters, the operations of sequential and parallel product, and sequential iterations. We also give a strictly algebraic characterization of the bounded-width property, in terms of depth-nilpotent series-parallel algebras.

[1]Part of this work was done while the second author was visiting the Institute of Mathematical Sciences, in Chennai. A preliminary version of this paper appeared in *STACS*, Paris (M. Morvan, C. Meinel, D. Krob, eds.), *Lect. Notes Comp. Sci.* **1373**, Springer (1998) 555–566.

Introduction

In this paper, we seek to extend automata theory from the usual languages of words, modelling sequential processes, to languages of labelled series-parallel posets, modelling hierarchies of concurrent processes.

Let A be a finite nonempty alphabet. A language over A accepted by a finite automaton is said to be *regular*. Languages can also be described by expressions using concatenation, union and iteration (Kleene star); such languages are called *rational*. Yet another way of describing languages is as unions of classes of a finite-index congruence in an algebra; such languages are *recognizable*. These terms are often used in an interchangeable way. Since we need to distinguish them in the paper, we have fixed on this terminology.

The cornerstones of automata theory are the Kleene theorem, which states that a language is regular if and only if it is rational, and the Myhill-Nerode theorem, which states that a language is regular if and only if it is recognizable. This possibility of considering and operating on regular languages from different points of view (not to mention the connection with logic, through the Büchi and McNaughton theorems) has been the basis for the success of automata theory: automata can be found everywhere in computer science, from text processing algorithms, to compression, to verification.

The model for concurrent processes which we consider in this paper, is that of finite partially ordered sets, or posets, labelled by the elements of alphabet A [27, 22]. This generalizes words over A, which may be viewed as particular posets, namely linearly ordered sets. More general structures have been considered in the literature. For instance, posets equipped with a conflict relation give rise to event structures [18], which are finer descriptions of concurrent behavior.

To get away from sequentiality, we consider the operation $\|$ of parallel product on posets. The posets obtained from one-element sets using only the operations of sequential and parallel product form the class of series-parallel posets (briefly, *sp*-posets) [11]. The formalism we develop in this paper is set entirely within the framework of *sp*-posets. Restricting ourselves to this class of posets has a patent weakness: series-parallel posets can only model a hierarchy of parallelism, but not communication by messages. As we shall see, this limitation leaves us in an algebraic framework which we can put to work for us.

Several methods have been considered in the literature to represent (languages of) posets. Grabowski [12] uses partial words. A different idea is to enrich the alphabet with an independence relation to yield traces [16]. It is quite common to reduce the study of poset languages to that of the usual word languages, by considering the set of linearizations of the posets in question (e.g. Garg and Ragunath [9]). In this paper, we deliberately choose to not follow this route and to operate on the posets themselves. This is made easier by the fact that we are working with *sp*-posets and *sp*-languages (sets of labelled *sp*-posets). Indeed the set $SP(A)$ of *sp*-posets labelled by alphabet A can be seen as a free algebra over A ([1], see the next section), which leads to some analogies with the set of A-labelled words and the free semigroup A^+.

We use the classical definition of recognizability for *sp*-languages, in terms of finite-index congruences on $SP(A)$, or equivalently, in terms of morphisms into finite algebras of the appropriate type. Also we propose a model of automata accepting *sp*-languages, thus defining regular *sp*-languages. It is important to note that the usual languages of words — which are special cases of *sp*-languages consisting only of linearly ordered sets — are recognizable (resp. regular) in this framework if and only if they are regular subsets of A^+ in the usual sense.

In the general case of *sp*-languages, the notions of recognizability and regularity do not coincide. This is hardly surprising whenever a commutative operation such as the parallel product is introduced: see Parikh's theorem on subsets of the free abelian semigroup and more generally the situation for trace languages [5]. The *sp*-language $(a\|b)^{\oplus}$ of all parallel products of equally many copies of a and b can be shown to be regular and not recognizable.

However there is an important family of *sp*-languages for which the two notions coincide. The width of a poset is defined as the maximal cardinality of an antichain: if the poset represents a concurrent process, it corresponds intuitively to the number of processors required to realize that process. An *sp*-language is said to have bounded width if there is a uniform bound on the width of its elements. One of the main results of the paper is the fact that if L is a bounded-width *sp*-language, then L is regular if and only if L is recognizable. Moreover, this is equivalent to L being series-rational: that is, L can be expressed from the singleton languages using a finite number of times the operations of union, sequential and parallel product, and sequential iteration (Kleene star). Thus, for the class of bounded-width *sp*-languages, we have analogues of both the Myhill-Nerode and the Kleene theorems.

We also explore in this paper the properties of recognizable and regular *sp*-languages. In particular, the class of regular *sp*-languages is closed under Boolean operations, sequential and parallel product, sequential and parallel iteration, direct and inverse morphism, and regular substitutions.

The paper is organised as follows. Section 1 lays down the foundation for our study of *sp*-posets and languages. In Section 2, recognizable *sp*-languages are defined and their closure properties are examined. Section 3 deals with bounded-width *sp*-languages: we give a characterization of these languages in terms of the properties of their syntactic algebra (whether the languages considered are recognizable or not). Then we use this algebraic characterization to establish that an *sp*-language is series-rational if and only if it is recognizable and it has bounded width. Next, in Section 4, we introduce our model of branching automata, we study the closure properties of regular *sp*-languages, and we show that the series-rational languages are exactly the regular *sp*-languages having bounded-width. In the concluding section, we discuss the relationship between rational, recognizable and regular languages if we drop the bounded-width assumption.

Before we dive into the body of the paper, let us go back to the discussion of related work in the literature. It should be clear to the reader familiar with concurrency theory that we are looking at Petri nets and related objects [23]. Our branching automata are a restricted form of nets, and the restriction imposed on automata to characterize bounded-width corresponds to 1-safe behaviour. We do not know of a characterization of the class of the sort of nets we are dealing with.

As we mentioned earlier, we are not looking at Petri net languages as sets of words representing the behavior of a net (as in, say, [9]). Our approach follows that of Grabowski [12], retaining concurrency information at the language level. Grabowski characterizes 1-safe Petri nets using poset (in his terminology, "partial word") languages, using a much richer syntax, including synchronization and renaming operations.

The reader might also think of our languages as a special kind of trace languages, on which there is a major body of work (see [5] for a recent summary). In traces, the alphabet is extended with an independence relation which allows commutativity of actions to represent concurrency. Ochmański showed that rational trace languages (defined using a concurrent star operation [19]) correspond to recognizable ones, and Zielonka introduced automata [28] to equate (in our terminology) the regular trace languages to the recognizable ones. Droste and Gastin discussed Zielonka-type automata for languages of pomsets without auto-concurrency in [7].

However, while the independence relation allows coding to represent communication between parallel processes — which is completely missing from our approach — it also prevents two actions from being serially as well as parallelly related. Thus an *sp*-language such as the solution of the equation $X = (a||bX) + c$ cannot be represented using traces. This *sp*-language, which is easily seen to have unbounded width, is briefly discussed in Section 5.

A very general algebraic approach using operations on hypergraphs has been developed by Courcelle [4] in connection with logic. It is easy to express series-parallel languages in this approach, but automata models operating on graphs have proved hard to find. Our notion of branching automata may be seen as a step in this direction.

1 *sp*-algebras and *sp*-posets

In this section, we make precise the algebraic and combinatorial framework in which we will work. We introduce series-parallel posets and algebras, and series-rational expressions.

1.1 *sp*-posets

Let A be an alphabet, that is, a finite non-empty set. An *A-labelled poset* is a poset $(P, \leq)$ equipped with a labelling function $\lambda\colon P \to A$. This is a generalization of words, which are finite linearly ordered A-labelled sets. We are in fact considering isomorphism classes of labelled posets (also known as *pomsets* [22]), so unless otherwise specified, the underlying sets of distinct posets are always assumed to be pairwise disjoint. All our posets are finite.

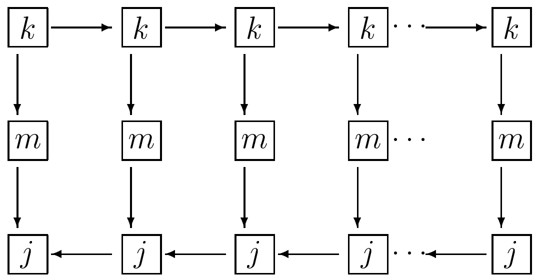


Figure 1: Series-parallel poset

Let $(P, \leq_P, \lambda_P)$ and $(Q, \leq_Q, \lambda_Q)$ be A-labelled posets. Their *parallel product* is

$$(P, \leq_P, \lambda_P)||(Q, \leq_Q, \lambda_Q) \quad = \quad (P \cup Q, \leq, \lambda)$$

where λ is uniquely defined by the fact that its restrictions to P and Q are respectively equal to λ_P and λ_Q, and the order relation on $P \cup Q$ is exactly the union of the order relations on P and Q.

The *sequential product* of the same posets is

$$(P, \leq_P, \lambda_P) \circ (Q, \leq_Q, \lambda_Q) \quad = \quad (P \cup Q, \leq, \lambda)$$

where the labelling function λ is the same as in the parallel product, and the order relation on $P \cup Q$ is obtained from that of $P||Q$ by requiring in addition that every element of P be less than every element of Q. That is, $\leq \; = \; \leq_P \cup \leq_Q \cup (P \times Q)$.

It is easily verified that both products are associative, that they have the same unit, namely the empty set, and that the parallel product is commutative. Note that there is no distributivity property between the two products.

We say that an A-labelled poset is *series-parallel*, an *sp-poset* for short, if either it is empty, or it can be obtained from singleton A-labelled posets by using a finite number of times the operations of sequential and parallel product.

Example 1.1 Figure 1 shows an example of an *sp*-poset. Indeed, let P_n be the poset in Figure 1 having n occurrences of k. Then

$$P_1 = \{k\} \circ \{m\} \circ \{j\} \text{ and } P_{n+1} = \{k\} \circ (\{m\}||P_n) \circ \{j\}.$$

Figure 2 shows a producer-consumer system, a simple example of message-passing. One can show that this poset is not series-parallel.

The trouble can be traced to the N-shaped subposet shown in Figure 3. A poset is *N-free* if it does not contain N as a subposet. That is, more formally, if it does not contain 4 distinct elements a, b, c and d among which the only order relations are $a < c$, $a < d$ and $b < d$. Observe that this property is independent of any labelling of the poset. It is well known that an A-labelled poset $(P, \leq, \lambda)$ is an *sp*-poset if and only if $(P, \leq)$ is N-free [12, 26].

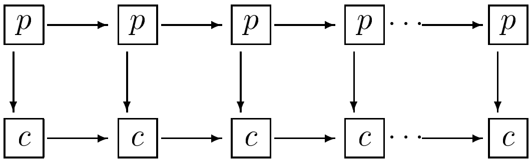


Figure 2: Producer-consumer

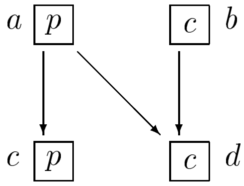


Figure 3: N

1.2 *sp*-algebras

The notion of *sp*-poset is generalized in the following fashion. An *sp-algebra* is a set S equipped with two binary operations, denoted $\circ$ and $\|$ and called respectively *sequential* and *parallel product*, such that $(S, \circ)$ is a semigroup (that is, the sequential product is associative), and such that $(S, \|)$ is a commutative semigroup. The discussion in the previous section shows that the class of (non-empty) *sp*-posets labelled by the alphabet A, forms an *sp*-algebra. If it does not create any ambiguity, we denote the sequential product as xy instead of $x \circ y$.

It is clear from this definition that *sp*-algebras form a variety of algebras (of signature $(2, 2)$), so there exists a free *sp*-algebra over each set A, denoted $SP(A)$. This free algebra was characterized by Bloom and Ésik [1, Thm. 3.3].

Proposition 1.2 *$SP(A)$ is (isomorphic to) the sp-algebra of non-empty A-labelled sp-posets.*

An *identity element* in an *sp*-algebra S is an element e which is an identity for both the sequential and the parallel product. Such an element, if it exists, is unique, and it is usually denoted 1. We say that an *sp*-algebra S is *with identity* if it has such an element. If S is an *sp*-algebra, we denote by S^1 the *sp*-algebra with identity which is equal to S if S has an identity, and to $S \cup \{1\}$ otherwise, where the added element 1 does not lie in S and acts as an identity for both products. For instance, $SP(A)^1$ is the set of all (possibly empty) *sp*-posets labelled by the set A, the identity being the empty set. Note that *sp*-algebras with identity are called *bimonoids* in [1].

A *zero* element in an *sp*-algebra is one which is absorbing for both product operations. If it exists, it is unique and is usually written 0.

As is usual with algebras, we define a *morphism* from an *sp*-algebra S to an *sp*-algebra T to be a mapping $\varphi: S \to T$ which preserves the two products, that is, such that if $x, y \in S$, then $\varphi(xy) = \varphi(x)\varphi(y)$ and $\varphi(x\|y) = \varphi(x)\|\varphi(y)$. A *congruence* on an *sp*-algebra S is an equivalence relation $\sim$ on S such that, if $x, y, z \in S$ and $x \sim y$, then

$xz \sim yz$, $zx \sim zy$ and $x\|z \sim y\|z$. The quotient set $S/\sim$ is then naturally equipped with the structure of an *sp*-algebra, and the projection morphism from S onto $S/\sim$ is a morphism.

1.3 *sp*-terms and *sp*-languages

The elements of the free *sp*-algebra over A, namely $SP(A)$, can be identified with A-labelled *sp*-posets according to Prop. 1.2. We abuse notation somewhat and call them *sp-terms*.

Let $x \in SP(A)$ be an *sp*-term. We say that x is a *sequential term* if it cannot be written as a parallel product $x = y\|z$ $(y, z \in SP(A))$. We say that x is a *parallel term* if it cannot be written as a sequential product $x = yz$ $(y, z \in SP(A))$. It is immediate that an *sp*-term x is sequential if and only if, as a poset, it cannot be partitioned in two subsets between which there is no order relation. The *sp*-term x is parallel if and only if it cannot be partitioned in two subsets, each element of the first one preceding each element of the second one.

Since the *sp*-algebra $SP(A)$ is freely generated by A, each *sp*-term can be written as an expression using only the letters of A and the sequential and parallel products. In particular, the only elements for which this expression is trivial are the letters of A, that is, the only *sp*-terms which are both parallel and sequential are the letters.

More precisely, the semigroup $(SP(A), \circ)$ is freely generated by the parallel terms, and the commutative semigroup $(SP(A), \|)$ is freely generated by the sequential terms. So, if $x \in SP(A)$ is an *sp*-term, then x admits a factorization of the form $x = x_1\| \cdots \|x_n$, where $n \geq 1$ and each x_i is a sequential term, and this factorization is unique up to the order of the factors. It is called the *parallel factorization* of x. The *sp*-term x also admits a unique factorization of the form $x = x_1 \circ \cdots \circ x_n$, where $n \geq 1$ and each x_i is a parallel term. That factorization is called the *sequential factorization* of x. It is clear that the parallel (resp. sequential) factorization of an *sp*-term x is trivial (that is, it has length 1) if and only if x is a parallel (resp. sequential) term.

Example 1.3 Let us follow up on the *sp*-poset of Example 1.1 above. For $n = 3$, the *sp*-term P_n can be written

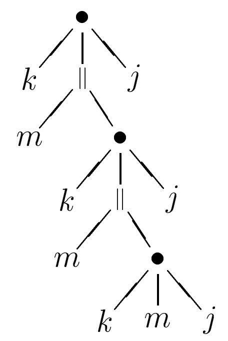
$$P_3 = k\Big(m\|\Big(k(m\|(kmj))j\Big)\Big)j.$$

For each $n \geq 1$, P_{n+1} is a sequential term, whose sequential factorization has length 3: $P_{n+1} = k \circ (m\|P_n) \circ j$. The first and last factor are letters, and the second one is a parallel term whose parallel factorization has length 2.

Note that, since $SP(A)$ is A-generated (or equivalently, for cardinality reasons), the process which consists in taking the parallel factorization of an *sp*-term x, then the sequential factorizations of the parallel factors of x, then the parallel factorizations of their sequential factors, etc, cannot go on forever: after a finite number of steps, we reach factorizations in which all the terms are letters.

This iterated factorization is conveniently represented by a tree, whose leaves are labelled by letters of A and whose other nodes are labelled either by the sequential product (the sequential nodes) or by the parallel product (the parallel nodes). In addition, no sequential (resp. parallel) node is a direct descendant of a sequential (resp. parallel) node, and the out-going arity of each node is different from 1. Such a tree representation of x is unique up to the order of the children of the parallel nodes (but including the order of the children of the sequential nodes).

Example 1.4 Let us again use the *sp*-poset of Example 1.1, with $n = 3$. The *sp*-term P_3 can be represented by the following tree.



It is easily verified that if x is a sequential term with sequential factorization $x = x_1 \cdots x_n$, then the root of the tree representing x is a sequential node, it has n children, and the subtrees hanging under the root are (in order) the trees representing $x_1, \ldots, x_n$. Similarly, if x is a parallel term with parallel factorization $x = x_1 \| \cdots \| x_n$, then the root of the tree representing x is a parallel node, it has n children, and the subtrees hanging under the root are the trees representing $x_1, \ldots, x_n$.

It follows from this discussion that, to prove properties of *sp*-terms, we can use two inductive processes. For a property $\mathcal{P}$ of *sp*-terms to be true, it suffices to show that $\mathcal{P}$ holds for the letters, and that if it holds for *sp*-terms x and y, then it holds for xy and for $x\|y$. Alternately, one can also show that $\mathcal{P}$ holds for the letters, and that if it holds for the factors in the parallel (resp. sequential) decomposition of a sequential (resp. parallel) term x, then it also holds for x.

Thus the *width* $\mathrm{wd}(x)$ of an *sp*-term $x \in SP(A)$, which counts the number of nested calls to the (binary) parallel product in the construction of x, can be inductively defined as:

$$\text{if } x \text{ is a letter, then } \mathrm{wd}(x) = 1;$$

$$\text{if } x = x_1x_2, \text{ then } \mathrm{wd}(x) = \max\{\mathrm{wd}(x_1), \mathrm{wd}(x_2)\};$$

$$\text{finally, if } x = x_1\|x_2, \text{ then } \mathrm{wd}(x) = \mathrm{wd}(x_1) + \mathrm{wd}(x_2).$$

The width of the *sp*-term P_3 in Example 1.4 is 3.

Lemma 1.5 *Let x be an element of $SP(A)$. The width of the sp-term x is the maximal cardinality of an anti-chain in the A-labelled sp-poset x.*

Proof. It is easily verified that if $\alpha(P, \leq)$ denotes the maximum cardinality of an antichain in the poset $(P, \leq)$, then $\alpha(a) = 1$ for each letter a, $\alpha(x_1 \circ x_2) = \max(\alpha(x_1), \alpha(x_2))$ and $\alpha(x_1 \| x_2) = \alpha(x_1) + \alpha(x_2)$. The lemma then follows by induction. $\square$

Remark. One can consider the width of an *sp*-term x as the number of processors needed to realize x. We could also define the width of an *sp*-term by saying that wd is the morphism from $(SP(A), \circ, \|)$ into the so-called tropical semiring $(\mathbf{N}, \max, +)$ which maps each letter to 1. Indeed, every semiring can be seen as an *sp*-algebra.

Remark. It is immediate that the free *sp*-algebra with identity is $SP(A)^1$. The above definitions can be easily extended to the identity of $SP(A)^1$, that is, the empty *sp*-poset: by convention, it is neither a sequential nor a parallel term, and it has parallel and sequential factorizations of length zero. Its tree representation is the empty tree, and its width is zero. However, in this paper, we will concentrate on properties (and subsets) of $SP(A)$.

An *sp-language* is a subset of some free *sp*-algebra $SP(A)$. (In this paper, we do not consider *sp*-languages in $SP(A)^1$, which might contain the identity element.)

For example, the free semigroup A^+ over A is an *sp*-language: it consists of those terms which do not use any parallel product. Dually, the free commutative semigroup $A^{\oplus}$ over A is contained in $SP(A)$.

We say that an *sp*-language $L \subseteq SP(A)$ has *bounded width* if there exists an integer k such that, for each element $x \in L$ we have $\mathrm{wd}(x) \leq k$. Of our two examples, A^+ is bounded width (k is 1), while $A^{\oplus}$ is not.

1.4 Subterms and substitution

Let u be an *sp*-term and let $a \in A$ be a letter. The *substitution of a by u* is the endomorphism $\sigma_{a,u}$ of $SP(A)$ which maps letter a to u, and which leaves every other letter unchanged. Since $SP(A)$ is free over A, this suffices to define $\sigma_{a,u}$. If $x \in SP(A)$, we often write $x[a, u]$ for $\sigma_{a,u}(x)$.

Let ξ be a letter. We say that an *sp*-term is a ξ-term if it contains exactly one occurrence of letter ξ. If $u, v \in SP(A)$ are *sp*-terms, we say that u is a *subterm* of x if there exists a ξ-term $c \in SP(A \cup \{\xi\})$ such that $\xi \notin A$ and such that $v = c[\xi, u]$. In that case, we say that the ξ-term c is a *context* of u in v. u may have several contexts in v.

Example 1.6 Let us again use the *sp*-posets of Example 1.1. We have seen that $P_{n+1} = k(m \| P_n)j$. So P_n is a subterm of P_{n+1}, in the context $c = k(m \| \xi)j$. In other words, $P_{n+1} = c[\xi, P_n]$. The term k is a subterm of P_n in n different ways, that is, in n different contexts. The term $(m \| (kmj))j$ also is a subterm of P_n but the term $(kmj)j$ is not.

1.5 Series-rational *sp*-languages

The series-rational expressions over the alphabet A are defined inductively as follows:

(1) if $a \in A$, then a is a series-rational expression;

(2) if e_1 and e_2 are series-rational expressions, then so are $e_1 + e_2$, $e_1 \circ e_2$ and $e_1 \| e_2$;

(3) if e is a series-rational expression, then e^+ is a series-rational expression.

We say that e^+ is the *sequential iteration* of e. Now, we associate with each series-rational expression, an *sp*-language in $SP(A)$:

(1) if $a \in A$, then $L(a) = \{a\}$;

(2) if e_1 and e_2 are series-rational expressions, then $L(e_1 + e_2) = L(e_1) \cup L(e_2)$, $L(e_1 \circ e_2) = L(e_1) \circ L(e_2)$ (that is, all the *sp*-terms of the form $t_1 \circ t_2$, with $t_1 \in L(e_1)$ and $t_2 \in L(e_2)$), and $L(e_1 \| e_2) = L(e_1) \| L(e_2)$;

(3) if e is a series-rational expression, then $L(e^+) = L(e)^+$, where for any *sp*-language U, we let $U^+ = \bigcup_{i \geq 1} U^i$, with $U^1 = U$ and $U^{i+1} = U \circ U^i$.

An *sp*-language $L \subseteq SP(A)$ is said to be *series-rational* if it is of the form $L(e)$ for some series-rational expression e over A. As is customary, we will often write a series-rational expression instead of an *sp*-language, that is, we will forget about the semantic mapping $L(\cdot)$. As usual again, the series-rational expression denoting a series-rational language is not unique.

The following result is immediate, from the definition of series-rational languages.

Lemma 1.7 *Every series-rational sp-language has bounded width.*

Of course, the converse does not hold: there are bounded-width *sp*-languages which are not series-rational. It suffices to consider non-rational languages contained in the free semigroup A^+, since such languages have width 1.

It will be convenient also to consider the parallel iteration $e^{\oplus}$. If U is an *sp*-language, we let $U^{(1)} = U$ and $U^{(i+1)} = U \| U^{(i)}$. Then $L(e^{\oplus}) = \bigcup_{i \geq 1} L(e)^{(i)}$.

Expressions using union, sequential and parallel product and the parallel iteration are called *parallel-rational*. Expressions using in addition the sequential iteration are called *series-parallel-rational expressions*. Of course these definitions extend to *sp*-languages, so we may talk of parallel-rational and of series-parallel-rational *sp*-languages.

2 Recognizable *sp*-languages

We say that an *sp*-language $L \subseteq SP(A)$ is *recognized* by a morphism φ from $SP(A)$ into an *sp*-algebra S if $L = \varphi^{-1}\varphi(L)$. In that case, we also say that L is *recognized* by the *sp*-algebra S. Of course, every *sp*-language is trivially recognized by the identity morphism of $SP(A)$. We say that an *sp*-language L is *recognizable* if it is recognized by a finite *sp*-algebra.

If $\sim$ is a congruence on $SP(A)$, we say that $\sim$ *recognizes* the *sp*-language L if L is a union of $\sim$-classes. It is immediate that, with this definition, an *sp*-language is recognizable if and only if it is recognized by a finite-index congruence.

This general treatment of recognizability in an algebra goes back to Mezei and Wright [17]. In the work of Doner [6] and Thatcher and Wright [25] (see also [10] and [2, Sections 6.3–6.6]), this is presented in terms of bottom-up deterministic tree automata and their response morphism. But these authors are concerned with recognizing languages of terms of absolutely free algebras (trees), in which no associativity or commutativity is assumed on the operations available. We will introduce automata for *sp*-languages in Section 4. In the general case of axiomatized Σ-algebras, properties of recognizable subsets are explored by Steinby [24] and Courcelle [3].

2.1 Examples of recognizable *sp*-languages

Let A be an alphabet and let $a \in A$. Then the *sp*-language $\{a\}$ is recognized by the 2-element *sp*-algebra $S = \{x, 0\}$ given by $x \circ x = x \| x = 0$. It suffices to consider the morphism from $SP(A)$ into S which maps a to x and the other letters of A to 0.

The *sp*-language $a^{\oplus}$ is recognized by the 2-element *sp*-algebra $\{x, 0\}$ given by $x \| x = x$ and $xx = 0$. One needs to consider the morphism from $SP(A)$ into S mapping a to x and the other letters to 0.

To show that a^{+} is recognizable, it suffices to exchange the roles of the sequential and the parallel products in the previous example.

These examples can easily be generalized. If $(S, \circ)$ is a semigroup, then fixing $s_0 \in S$ and letting $s \| t = s_0$ for all $s, t \in S$ makes S into an *sp*-algebra. It follows that any subset of the free semigroup A^{+} which is recognizable in the usual sense (i.e. by a morphism from A^{+} into a finite semigroup) is a recognizable *sp*-language. Conversely, if a recognizable *sp*-language L is contained in A^{+} (that is, it consists only of sequential words), and if it is recognized by a finite *sp*-algebra $(S, \circ, \|)$, then it is also recognized, as a language in the free semigroup A^{+}, by the semigroup $(S, \circ)$. Applying the same reasoning to the subsets of $A^{\oplus}$, we have the following result.

Lemma 2.1 *Let $L \subseteq A^{+}$ (resp. $L \subseteq A^{\oplus}$). The sp-language L is recognizable if and only if L is a recognizable subset of A^{+} (resp. $A^{\oplus}$).*

This provides examples of non-recognizable *sp*-languages. For instance $\{a^n b^n \mid n \geq 1\}$ is not recognizable, since it is not recognizable in A^{+}. Similarly, the *sp*-language $(a\|b)^{\oplus}$ is not recognizable since it is not recognizable in $A^{\oplus}$ (to see that, observe that its inverse image in A^{+} is the set of all words having as many a as b, which is not recognizable).

We conclude this section with a more complex example.

Example 2.2 On the 3-letter alphabet $A = \{a, b, c\}$, we consider the least *sp*-language L containing c and such that if $x \in L$, then $a\|bx \in L$. $L = \{c, a\|(bc), a\|(b(a\|bc)), \ldots\}$. Then L is recognized by the 5-element *sp*-algebra $S = \{p, q, r, s, 0\}$ where the only non-zero sequential product is $qr = s$ and the only non-zero parallel product is $p\|s = r$. One can verify that the morphism $\varphi\colon SP(A) \to S$ mapping a to p, b to q and c to r is such that $\varphi^{-1}(r) = L$. Thus L is recognizable.

Note however that L is not series-parallel-rational. Indeed, the *sp*-terms appearing in the iterated sequential and parallel factorizations of the elements of L all have either a parallel factorization of length at most 2 or a sequential factorization of length at most 2, so no sequential or parallel iteration may be used in the description of L.

2.2 Syntactic congruence

Let $L \subseteq SP(A)$ be an *sp*-language. The *syntactic congruence* of L is defined as follows. Let $u, v \in SP(A)$. Intuitively, we say that u is $\sim_L$-equivalent to v if, for each *sp*-term $t \in L$ such that u (resp. v) is a subterm of t, replacing u by v (resp. v by u) in t yields another element of L. More precisely, we let $u \sim_L v$ if, for each ξ-term c (where $\xi \notin A$), we have

$$c[\xi, u] \in L \iff c[\xi, v] \in L.$$

It is easily verified that $\sim_L$ is a congruence on $SP(A)$ which recognizes L. The quotient *sp*-algebra $\mathrm{Synt}(L) = SP(A)/\sim_L$ is called the *syntactic sp-algebra* of L and the projection morphism $\mu_L\colon SP(A) \to \mathrm{Synt}(L)$ is called the *syntactic morphism* of L.

Remark. Recall that if L is a subset of a semigroup $(S, *)$, the usual definition of the syntactic congruence of L in S is the congruence $\sim_L$ given by

$$s \sim_L t \quad \text{if and only if} \quad \forall\, x, y \in S \cup \{1\},\ x * s * y \in L \Leftrightarrow x * t * y \in L.$$

The above definition of the syntactic congruence of an *sp*-language is a generalization of this definition, adapted to the situation where there are several operations.

As could be expected, the syntactic *sp*-algebra $\mathrm{Synt}(L)$ is the "least" *sp*-algebra recognizing L in the following sense.

Proposition 2.3 *Let $L \subseteq SP(A)$ be an sp-language, and let $\varphi\colon SP(A) \to S$ be an onto morphism into an sp-algebra S. Then $L = \varphi^{-1}\varphi(L)$ if and only if, for any $u, v \in SP(A)$, $\varphi(u) = \varphi(v)$ implies $u \sim_L v$.*

In other words, φ recognizes L if and only if there exists an onto morphism $\psi\colon S \to \mathrm{Synt}(L)$ such that $\mu_L = \psi \circ \varphi$.

Proof. If $\mu_L = \psi \circ \varphi$, then φ recognizes L since μ_L recognizes L. For the converse proof, let us consider *sp*-terms $u, v \in SP(A)$ such that $\varphi(u) = \varphi(v)$. Let ξ be a letter not in A.

The morphisms $\varphi \circ \sigma_{\xi,u}$ and $\varphi \circ \sigma_{\xi,v}$, from $SP(A \cup \{\xi\})$ into $SP(A)$, map ξ to $\varphi(u) = \varphi(v)$ and map each letter $a \in A$ to $\varphi(a)$. Thus they coincide on $A \cup \{\xi\}$, and therefore they coincide on all of $SP(A \cup \{\xi\})$. In particular, if c is a ξ-term, we have $\varphi(c[\xi, u]) = \varphi(c[\xi, v])$, and hence $c[\xi, u] \in L$ if and only if $c[\xi, v] \in L$, that is, $u \sim_L v$. $\square$

Corollary 2.4 *Let L be an sp-language. Then L is recognizable if and only if the syntactic congruence $\sim_L$ of L has finite index.*

Note the following result, which will be used in the sequel.

Lemma 2.5 *Let L be an sp-language and s an sp-term which is not a subterm of any element of L. Then* $\mathrm{Synt}(L)$ *has a zero and $\mu_L(s) = 0$, where μ_L is the syntactic morphism of L.*

Proof. By definition, the *sp*-terms which are not subterms of elements of L are all syntactically equivalent, that is, they have the same image in $\mathrm{Synt}(L)$. Let s be such a term. To see that $\mu(s)$ is absorbing for both operations, it suffices to observe that for any $x \in SP(A)$, s is a subterm of xs, sx and $s\|x$, so that these three *sp*-terms too, are not subterms of any element of L. It follows that $\mu(s) = \mu(xs) = \mu(sx) = \mu(s\|x)$. $\square$

2.3 Computing the syntactic congruence

We present here an adaptation of the usual algorithm to compute the minimal automaton of a language L starting from some deterministic automaton recognizing L [13].

Let L be a *sp*-language, and let $\varphi\colon SP(A) \to S$ be an onto morphism into a *sp*-algebra S recognizing L (in particular, φ may be the identity morphism of $SP(A)$). We define a sequence $(\equiv_{S,n})_{n\geq 0}$ of equivalence relations on S as follows. The relation $\equiv_{S,0}$ has two classes: $\varphi(L)$ and $S \setminus \varphi(L)$ (it has only one class if L is empty or equal to $SP(A)$).

If $\equiv_{S,n}$ is defined for some $n \geq 0$, then we let $s \equiv_{S,n+1} t$ if and only if $s \equiv_{S,n} t$ and for each $u \in S$, we have $su \equiv_{S,n} tu$, $us \equiv_{S,n} ut$ and $s\|u \equiv_{S,n} t\|u$.

Then each $\equiv_{S,n}$ is an equivalence relation, and $\equiv_{S,n+1}$ is contained in $\equiv_{S,n}$. Let $\equiv_S$ be the intersection of the relations $\equiv_{S,n}$ $(n \geq 0)$. Then $\equiv_S$ is easily seen to be a congruence on S. In fact, one can verify that $\equiv$ is the greatest congruence on S contained in $\equiv_{S,0}$. Moreover the quotient algebra $S/{\equiv_S}$ recognizes S, since $\equiv_S$ refines the relation $\equiv_{S,0}$. Let $\pi\colon S \to S/{\equiv_S}$ be the projection morphism.

By Proposition 2.3, $\varphi^{-1}(\equiv_S)$ is contained in $\sim_L$. Conversely, it is immediate that $\sim_L$ is contained in $\varphi^{-1}(\equiv_{S,0})$, that is, $\varphi(\sim_L)$ is contained in $\equiv_{S,0}$. Therefore $\varphi(\sim_L)$ is contained in $\equiv_S$ and hence $\sim_L$ is contained in $\varphi^{-1}(\equiv_S)$. In particular, $S/{\equiv_S}$ is equal to (in fact, isomorphic to) $\mathrm{Synt}(L)$, and $\pi \circ \varphi$ is the syntactic morphism μ_L.

In the particular case of a recognizable *sp*-language L, where we are given explicitly a morphism $\varphi\colon SP(A) \to S$ into a finite *sp*-algebra S, and a subset $P \subseteq S$ such that $L = \varphi^{-1}(P)$, we can now explicitly compute the syntactic *sp*-algebra $\mathrm{Synt}(L)$, which we already know to be finite. It suffices to compute the increasing sequence $\equiv_{S,n}$. It is easily verified also that if $\equiv_{S,n}=\equiv_{S,n+1}$ for some n, then $\equiv_{S,n}=\equiv$. The finiteness of S now immediately implies that we can compute $\sim_L$ and $\mathrm{Synt}(L)$ in a finite number of steps.

2.4 Some closure properties of recognizable *sp*-languages

The main theorem of this section is the following.

Theorem 2.6 *Let A be an alphabet. The class of recognizable sp-languages over A is closed under Boolean operations, inverse image by a morphism, sequential product, parallel product and sequential iteration. It is not closed under parallel iteration or homomorphic image.*

Regarding the negative part of Theorem 2.6, let us note immediately that $a^{\oplus}$ is recognizable, whereas $(a\|b)^{\oplus}$ is not (see Section 2.1). But the latter language is both a homomorphic image of $a^{\oplus}$, and the parallel iteration of a finite, and hence recognizable language.

That recognizable languages are closed under Boolean operations and inverse morphisms holds for any algebra [24]. The next three lemmas, more specific to *sp*-languages, complete the proof of Theorem 2.6.

Lemma 2.7 *Recognizable sp-languages are closed under sequential product.*

Proof. This proof is an adaptation of the classical proof for languages in the free semigroup (see [8] for example). Let $\sim$ and $\sim'$ be the syntactic congruences of two recognizable *sp*-languages L and L' in $SP(A)$. By hypothesis, these are finite-index congruences. We define the relation $\equiv$ on $SP(A)$ by letting $u \equiv v$ if and only if

(1) $u \sim v,\ u \sim' v$,

(2) for each factorization $u = u_1u_2$ in $SP(A)$,
there exists a factorization $v = v_1v_2$ in $SP(A)$ such that $u_1 \sim v_1$ and $u_2 \sim' v_2$,

(3) for each factorization $v = v_1v_2$ in $SP(A)$,
there exists a factorization $u = u_1u_2$ in $SP(A)$ such that $u_1 \sim v_1$ and $u_2 \sim' v_2$.

The relation $\equiv$ is clearly an equivalence relation, and it has finite index, bounded above by $ii'2^{ii'}$, where i (resp. i') is the index of $\sim$ (resp. $\sim'$).

Next, we verify that LL' is a union of $\equiv$-classes. Indeed, let $u, v \in SP(A)$ be such that $u \in LL'$ and $u \equiv v$. Then there exists a factorization $u = u_1u_2$ with $u_1 \in L$ and $u_2 \in L'$. By definition of $\equiv$, there exists a factorization $v = v_1v_2$ such that $u_1 \sim v_1$ and $u_2 \sim' v_2$. Now $u_1 \in L$ implies $v_1 \in L$, and similarly $u_2 \in L'$ implies $v_2 \in L'$. Thus $v \in LL'$.

Now it suffices to check that $\equiv$ is a congruence. Let us assume that $u \equiv v$ and let $x \in SP(A)$. First we consider the parallel product. There is no factorization of $x\|u$ or $x\|v$ as a sequential product. Since $\sim$ and $\sim'$ are congruences of $SP(A)$, we have $x\|u \sim x\|v$ and $x\|u \sim' x\|v$, so $x\|u \equiv x\|v$.

Next we consider the sequential product. Since $\sim$ and $\sim'$ are congruences of $SP(A)$, we have $xu \sim xv$ and $xu \sim' xv$. Let us now consider a factorization $xu = x_1x_2$. Recall that the semigroup $(SP(A), \circ)$ is freely generated by the parallel terms. That is, each

sp-term in $SP(A)$ can be considered to be a (sequential) word over the set of parallel terms, namely its sequential decomposition. In the case of the sp-term $xu = x_1x_2$, two cases arise, according to the relation between the lengths of the words u and x_2.

- If the word x_2 is longer than the word u, then we have $x_2 = x_2'u$ and $x = x_1x_2'$ for some term $x_2' \in SP(A)$. Let $y_1 = x_1$ and $y_2 = x_2'v$. Then we get immediately $y_1y_2 = xv$ and $x_1 \sim y_1$. Moreover, $u \sim' v$ implies $x_2 = x_2'u \sim' x_2'v = y_2$.
- If the word x_2 is strictly shorter than u, then we have $x_1 = xx_1'$ and $u = x_1'x_2$ for some term $x_1' \in SP(A)$. Since $u \equiv v$, there exists a factorization $v = v_1v_2$ such that $x_1' \sim v_1$ and $x_2 \sim' v_2$. Let $y_1 = xv_1$ and $y_2 = v_2$. Then we get immediately $y_1y_2 = xv$ and $x_2 \sim y_2$. Moreover, $x_1' \sim v_1$ implies $x_1 = xx_1' \sim xv_1 = y_1$.

It follows that $xu \equiv xv$. The proof that $ux \equiv vx$ is entirely similar. Thus $\equiv$ is a congruence on $SP(A)$, it has finite index and it recognizes LL'. Therefore LL' is recognizable. $\square$

Lemma 2.8 *Recognizable sp-languages are closed under parallel product.*

Proof. We adapt the previous proof to the (commutative) parallel product. If L and L', $\sim$ and $\sim'$ are as in the previous proof, we now define the relation $\equiv$ on $SP(A)$ by letting $u \equiv v$ if and only if

(1) $u \sim v,\ u \sim' v$,

(2) for each factorization $u = u_1 \| u_2$ in $SP(A)$,
there exists a factorization $v = v_1 \| v_2$ in $SP(A)$ such that $u_1 \sim v_1$ and $u_2 \sim' v_2$,

(3) for each factorization $v = v_1 \| v_2$ in $SP(A)$,
there exists a factorization $u = u_1 \| u_2$ in $SP(A)$ such that $u_1 \sim v_1$ and $u_2 \sim' v_2$.

As above, one verifies that $\equiv$ is a finite-index equivalence relation of $SP(A)$ and that $L \| L'$ is a union of $\equiv$-classes. Next, we verify that $\equiv$ is a congruence. Let u, v be sp-terms such that $u \equiv v$, and let $x \in SP(A)$.

There is no factorization of xu or ux as a parallel product, so as above, it is immediate that $xu \equiv xv$ and $ux \equiv vx$. Let us now show that $u \| x \equiv v \| x$.

By Levi's lemma for the free commutative semigroup [5, Thm 1.3.4 or Cor. 6.2.4], if $x \| u = x_1 \| x_2$, then there exist sp-terms $p, q, r, s \in SP(A)$ such that $x = p \| q$, $u = r \| s$, $x_1 = q \| r$ and $x_2 = p \| s$. Let us first assume that $r \neq 1$ and $s \neq 1$. Since $u \equiv v$, there exists a factorization $v = r' \| s'$ such that $r \sim r'$ and $s \sim' s'$. Now let $y_1 = q \| r'$ and $y_2 = p \| s'$. Then $y_1 \| y_2 = x \| v$, $x_1 \sim y_1$ and $x_2 \sim' y_2$.

If $r = 1$, to get the same result, it suffices to take $y_1 = q$ and $y_2 = p \| v$. Dually, if $s = 1$, it suffices to take $y_1 = q \| v$ and $y_2 = p$. Therefore $u \| x \equiv v \| x$. This concludes the proof. $\square$

Lemma 2.9 *Recognizable sp-languages are closed under sequential iteration.*

Proof. The proof is an imitation of the classical construction of an automaton to recognize the iteration of a rational language [13].

Let $\varphi: SP(A) \to S$ be a morphism into a finite *sp*-algebra with identity, recognizing L. Let i be a new symbol not in S, and let S^i be the *sp*-algebra $S \cup \{i\}$ defined by letting i act as an identity for both the sequential and the parallel product (the element i is added whether S has an identity or not). We define a mapping $\Delta: S^i \times SP(A) \to \mathcal{P}(S^i)$ as follows. Let $s \in S^i$, let $t \in SP(A)$, and let $t = t_1 \cdots t_k$ ($k \geq 1$) be the sequential decomposition of t.

$$\text{If } k = 1,\ \Delta(s,t) = \begin{cases} \{s\varphi(t)\} & \text{if } s\varphi(t) \notin \varphi(L), \\ \{s\varphi(t), i\} & \text{if } s\varphi(t) \in \varphi(L); \end{cases}$$
$$\text{If } k \geq 2,\ \Delta(s,t) = \bigcup_{u \in \Delta(s, t_1 \cdots t_{k-1})} \Delta(u, t_k).$$

Next we define a relation $\equiv$ on $SP(A)$ by letting $u \equiv v$ if and only if $\varphi(u) = \varphi(v)$ and $\Delta(s,u) = \Delta(s,v)$ for each $s \in S^i$. This relation is seen to be an equivalence relation. It has finite index (bounded above by $n(2^{n+1})^{n+1}$ if $n = |S|$). Moreover one verifies that an *sp*-term x lies in L^+ if and only if $i \in \Delta(i,x)$. It follows that L^+ is a union of $\equiv$-classes.

So it suffices to check that $\equiv$ is a congruence. Let x, u, v be *sp*-terms such that $u \equiv v$. One verifies that for each $s \in S^i$, we have $\Delta(s, ux) = \bigcup_{t \in \Delta(s,u)} \Delta(t, x)$ (by induction on the length of the sequential decomposition of x). The hypothesis that $u \equiv v$ implies that $\Delta(s,u) = \Delta(s,v)$, so $\Delta(s,ux) = \Delta(s,vx)$, and hence $ux \equiv vx$.

Similarly, $\Delta(s, xu) = \bigcup_{t \in \Delta(s,x)} \Delta(t,u) = \bigcup_{t \in \Delta(s,x)} \Delta(t,v) = \Delta(s,xv)$, so $xu \equiv xv$.

Finally, $\Delta(s, x\|u)$ is equal to $\{s(\varphi(x)\|\varphi(u)), i\}$ or $\{s(\varphi(x)\|\varphi(u))\}$, depending on whether $s(\varphi(x)\|\varphi(u))$ lies in $\varphi(L)$ or not. Since $u \equiv v$, we have $\varphi(u) = \varphi(v)$, so $x\|u \equiv x\|v$. This concludes the proof. □

The proof of Theorem 2.6 is now complete. It yields the following important corollary.

Corollary 2.10 *Let L be a sp-language. If L is series-rational, then L is recognizable.*

Proof. In view of Theorem 2.6, it suffices to verify that each letter constitutes a recognizable *sp*-language. This was done in Section 2.1. □

The converse of Corollary 2.10 does not hold: there are recognizable *sp*-languages which are not series-rational. For instance, if $a \in A$ then $a^\oplus$ is not series-rational (by Lemma 1.7). However, it is recognizable, see Section 2.1.

3 Bounded-width *sp*-languages

The *sp*-language $a^\oplus$ is recognizable but not bounded width. This leads us to the conjecture that the series-rational *sp*-languages may coincide with the recognizable ones having bounded width. This turns out to be true, as we prove at the end of this section.

For this purpose, we give an algebraic characterization of the *sp*-languages having bounded width (whether recognizable or not). The homomorphic image, the union, the intersection, the sequential product, the parallel product and the sequential iteration of bounded-width *sp*-languages are again bounded-width; however, the complement obviously does not have bounded width.

3.1 The parallel nesting relation in *sp*-algebras

Let S be an *sp*-algebra. We define three relations on S, written $\prec$, $<_{\mathrm{n}}$ and $\leq_{\mathrm{n}}$. The relation $s \prec t$ essentially states that t occurs as a sequential factor of a parallel factor, in a non-trivial parallel factorization of s. More precisely, if $s, t \in S$, we let

$$s \prec t \quad \text{if and only if} \quad s = u(p\|(qtr))v \text{ for some } u, p, q, r, v \in S^1 \text{ such that } p\|(qtr) \neq qtr.$$

The relation $<_{\mathrm{n}}$ is defined to be the transitive closure of $\prec$, and it is called the *parallel nesting relation*. The relation $\leq_{\mathrm{n}}$ is defined to be the reflexive closure of $<_{\mathrm{n}}$. By construction, $\leq_{\mathrm{n}}$ is a quasi-order on S.

Example 3.1 In $SP(A)$, we have $a\|(bcd) \prec c$ whereas $(a\|b)cd \not\prec c$. If $s, t \in SP(A)$ are *sp*-terms, then $s <_{\mathrm{n}} t$ if and only if t is a subterm of s and for at least one context c of t in s (c is a ξ-term), ξ is not a sequential factor of c. In particular, $\leq_{\mathrm{n}}$ is a partial order on $SP(A)$.

We will be concerned with *sp*-algebras S in which, for each $s \in S$, there exists an integer k such that each $<_{\mathrm{n}}$-chain with least element s has length at most k. Such algebras are called *depth-graded*. Example 3.1 shows that this is the case if $S = SP(A)$, the free *sp*-algebra. Clearly, in a depth-graded *sp*-algebra, the relation $<_{\mathrm{n}}$ is irreflexive, and hence it is a strict partial order.

In addition, if S is a depth-graded *sp*-algebra, there is a notion of *parallel nesting depth*, or simply *depth*, for the elements of S:

$$\mathrm{dp}(s) = \begin{cases} 1 & \text{if } s \text{ is } \leq_{\mathrm{n}}\text{-maximal,} \\ 1 + \max\{\mathrm{dp}(t) \mid s <_{\mathrm{n}} t\} & \text{otherwise.} \end{cases}$$

That is, the depth of s is the maximum cardinality of a $<_{\mathrm{n}}$-chain with least element s. Note that the maximal $<_{\mathrm{n}}$-chains are $\prec$-chains.

If L is a subset of a depth-graded *sp*-algebra S, we let $\mathrm{dp}(L) = \sup_{x \in L} \mathrm{dp}(x)$. If $\mathrm{dp}(L) < \infty$, we say that L has *bounded depth*.

Remark. The relation $\prec$ can also be defined in terms of the semigroup-theoretic Green $\mathcal{J}$-quasi-order (see [14]). If we let $\leq_{\mathcal{J}\circ}$ (resp. $\leq_{\mathcal{J}\|}$) be the $\mathcal{J}$-quasi-order of the semigroup $(S, \circ)$ (resp. $(S, \|)$), then we have

$$s \prec t \quad \text{if and only if} \quad s \leq_{\mathcal{J}\circ} x \leq_{\mathcal{J}\|} y \leq_{\mathcal{J}\circ} t \text{ for some } x \neq y \in S.$$

It is now immediate that $\leq_{\mathcal{J}\|}$ is contained in $\leq_{\mathrm{n}}$. In particular, if $\leq_{\mathrm{n}}$ is a partial order, then so is $\leq_{\mathcal{J}\|}$, and hence $(S, \|)$ is a $\mathcal{J}$-trivial commutative semigroup.

3.2 Depth vs. width in the free algebra

In view of Example 3.1, one can verify that the free *sp*-algebra $SP(A)$ is depth-graded, but does not have bounded depth. More precisely, here is a procedure for the computation of the depth of an *sp*-term $x \in SP(A)$, whose proof is left to the reader. If $x \in A^+$ (x is a sequential word), then $\mathrm{dp}(x) = 1$. Otherwise, the tree representing x has at least one $\|$-labelled node. With each such node, associate a weight equal to the number of children of that node minus one. Then $\mathrm{dp}(x) - 1$ is the maximum value of the sum of the weights of $\|$-labelled nodes along a path from root to leaf in the tree of x.

It is easily verified that if $x \in SP(A)$ is an *sp*-term, then $\mathrm{dp}(x) \leq \mathrm{wd}(x)$. Of course, we do not necessarily have equality between the width and the depth an of *sp*-term: if $u = a(a\|a)$ and if x is the parallel product of n copies of u, then one verifies that $\mathrm{wd}(x) = 2n$ and $\mathrm{dp}(x) = n + 1$. This example can be generalised by letting $u_k = a(a\|a\|\cdots\|a)$ (with a parallel product of k factors) and letting $x_{k,n}$ be the parallel product of n copies of u_k. Then $\mathrm{wd}(x_{k,n}) = kn$ and $\mathrm{dp}(x_{k,n}) = n - 1 + k$. So the ratio $\mathrm{wd}(x)/\mathrm{dp}(x)$ is not bounded.

Yet very wide *sp*-terms must be very deep.

Proposition 3.2 *Let L be an sp-language. Then L has bounded width if and only if L has bounded depth.*

Proof. We already noticed that $\mathrm{dp}(x) \leq \mathrm{wd}(x)$ for each $x \in SP(A)$. So a bounded-width *sp*-language also has bounded depth. To prove the converse, we show that for any $x \in SP(A)$, we have a crude (factorial) upper bound: $\mathrm{wd}(x) \leq \mathrm{dp}(x)!$.

First we observe that the following formulæ hold for any $x_1, \ldots, x_n \in SP(A)$:

$$\begin{aligned}\mathrm{dp}(x_1 \cdots x_n) &= \max\{\mathrm{dp}(x_1), \ldots, \mathrm{dp}(x_n)\},\\ \mathrm{dp}(x_1\|\cdots\|x_n) &= n - 1 + \max\{\mathrm{dp}(x_1), \ldots, \mathrm{dp}(x_n)\}.\end{aligned}$$

It follows that if $\mathrm{dp}(x) = d$, then any parallel term which appears as a subterm of x has at most d factors. We now show by induction on d, that $\mathrm{wd}(x) \leq d!$.

If $\mathrm{dp}(x) = d = 1$, then x is a sequential word, that is, $x \in A^+$, and $\mathrm{wd}(x) = 1$. Let us assume that $d \geq 2$ and that if $\mathrm{dp}(x) \leq d - 1$, then $\mathrm{wd}(x) \leq (d-1)!$. Now we consider $x \in SP(A)$ such that $\mathrm{dp}(x) = d$.

If x is a parallel term, then the parallel factorization of x is $x = x_1\|\cdots\|x_n$ with $2 \leq n \leq d$, and $d = n - 1 + \max\{\mathrm{dp}(x_i) \mid 1 \leq i \leq n\}$. In particular, $\mathrm{dp}(x_i) \leq d - 1$ for each i. Moreover $\mathrm{wd}(x) = \sum_{i=1}^{n} \mathrm{wd}(x_i)$. So by induction we have $\mathrm{wd}(x) \leq d(d-1)! = d!$.

If x is a sequential term, then its sequential factorization is $x = x_1 \cdots x_n$ with $n \geq 2$, and $d = \max\{\mathrm{dp}(x_i) \mid 1 \leq i \leq n\}$. So each x_i is a parallel term of depth at most d and we already know that $\mathrm{wd}(x_i) \leq d!$. Therefore $\mathrm{wd}(x) = \max\{\mathrm{wd}(x_i) \mid 1 \leq i \leq n\} \leq d!$.

This concludes the proof that $\mathrm{wd}(x) \leq \mathrm{dp}(x)!$ for each $x \in SP(A)$. It follows immediately that a bounded-depth *sp*-language also has bounded width. $\square$

3.3 Depth-nilpotent *sp*-algebras

We now return to the general situation. The existence of a zero element is related to an *sp*-algebra having bounded depth.

Lemma 3.3 *Let S be an sp-algebra such that $<_{\mathrm{n}}$ is a strict partial order, and assume that S has a $\leq_{\mathrm{n}}$-minimal element. Then S has a zero, which is $\leq_{\mathrm{n}}$-minimum.*

Proof. Let us assume that there exists an element $z \in S$ which is $\leq_{\mathrm{n}}$-minimal in S. Let $x \in S$. If $z \| x \neq z$, then $z \| x <_{\mathrm{n}} z$, in contradiction with the minimality of z. So $z \| x = z$ for each $x \in S$, and hence $z \leq_{\mathrm{n}} x$ for each x. Thus z is $\leq_{\mathrm{n}}$-minimum. Moreover, this shows that z is absorbing for the parallel product.

Now let $x \in S$. We show that zx is $\leq_{\mathrm{n}}$-minimum, and hence that $zx = z$. Indeed, for each $s \in S \setminus \{z\}$, we have $z <_{\mathrm{n}} s$, that is, $z \prec s_1 \prec \ldots \prec s_k = s$ for some sequence $s_1, \ldots, s_k$ ($k \geq 1$) of elements of S. In particular, $z = z \| s_1 \neq s_1$. Then $zx = (z \| s_1)x$ so $zx \prec s_1$, and hence $zx <_{\mathrm{n}} s$. So zx too is $\leq_{\mathrm{n}}$-minimum, and hence $zx = z$. Similarly, one verifies that $xz = z$. □

We say that an *sp*-algebra S has *bounded depth* if there exists an integer d such that each $\prec$-chain is of length at most d. If S is a bounded-depth *sp*-algebra, then S is trivially depth-graded, $\mathrm{dp}(S) < \infty$ and any element of maximal depth is $\leq_{\mathrm{n}}$-minimal. The converse also holds: if S is depth-graded and S has a $\leq_{\mathrm{n}}$-minimal element, then that element is 0 by Lemma 3.3 and it is $\leq_{\mathrm{n}}$-minimum. In particular, every other element of S has depth at most $\mathrm{dp}(0) - 1$. That is, we have the following lemma.

Lemma 3.4 *An sp-algebra S has bounded depth if and only if it is depth-graded and it has an $\leq_{\mathrm{n}}$-minimal element.*

Recall that a semigroup S is said to be *nilpotent* if it has a zero and there exists an integer $c \geq 1$ such that any product of at least c elements of S is equal to 0. In that case, 0 is the only idempotent element in S. We say that an *sp*-algebra S is *depth-nilpotent* if it has bounded depth and 0 is the only idempotent element of S for the parallel product.

Lemma 3.5 *Let S be an sp-algebra with bounded depth. The following are equivalent:*

(1) S is depth-nilpotent.

(2) If $s, t \in S$ and $t \neq 0$, then $s \| t \neq t$.

(3) The semigroup $(S, \|)$ is nilpotent.

Proof. Let $d = \mathrm{dp}(S)$. We first assume that S satisfies (1). Let $s, t \in S$ such that $s \| t = t$, let $s_1 = s$ and for each $n \geq 1$, let $s_{n+1} = s \| s_n$. If $s_{i+1} \neq s_i$ for each $i \leq d$, then $s_{d+1} \prec s_d \prec \ldots \prec s_1$, so $\mathrm{dp}(s_{d+1}) \geq d + 1$, a contradiction. It follows that $s_{n+1} = s_n$ for some $n \leq d$, and hence $s_m = s_n$ for each $m \geq n$. In particular $s_{2n} = s_n \| s_n = s_n$. Therefore $s_n = 0$ and hence $t = s_n \| t = 0$. Thus (1) implies (2).

Now we assume that S satisfies (2). Let $s_1, \ldots, s_{d+1} \in S$. Let also $t_1 = s_1$ and $t_{i+1} = s_{i+1} \| t_i$ for each $i \geq 1$. As above, if $t_{i+1} \neq t_i$ for each $i \leq d$, then $\mathrm{dp}(t_{d+1}) \geq d+1$, a contradiction. So there exists $1 \leq i \leq d$ such that $t_{i+1} = s_{i+1} \| t_i = t_i$. It follows that $t_i = 0$ and hence $t_{d+1} = s_{d+1} \| \cdots \| s_2 \| s_1 = 0$. Thus (2) implies (3).

It is immediate that in a nilpotent semigroup, 0 is the only idempotent element, so (3) implies (1), which concludes the proof. $\square$

We will use the following property of depth-nilpotent *sp*-algebras.

Lemma 3.6 *Let $\varphi: T \to S$ be a morphism between sp-algebras, and assume that S is depth-nilpotent. If $t, t' \in T$ and $t \prec t'$, then either $\varphi(t) = 0$ or $\varphi(t) \prec \varphi(t')$.*

Proof. By definition, there exist elements $u, p, q, r, v \in T^1$ such that $t = u(p \| (qt'r))v$ and $p \| (qt'r) \neq qt'r$. Taking the φ-images, we get

$$\varphi(t) = \varphi(u)(\varphi(p) \| (\varphi(q)\varphi(t')\varphi(r))\varphi(v).$$

By Lemma 3.5, we have either $\varphi(q)\varphi(t')\varphi(r) = 0$, or $\varphi(p) \| (\varphi(q)\varphi(t')\varphi(r)) \neq \varphi(q)\varphi(t')\varphi(r)$. In the first case, $\varphi(t) = 0$, and in the second case, $\varphi(t) \prec \varphi(t')$ as required. $\square$

3.4 Bounded width and depth-nilpotency

Lemma 3.6 justifies the choice of the name "depth-nilpotent": it shows in particular that, if $\varphi: SP(A) \to S$ is a morphism into a depth-nilpotent *sp*-algebra, then the image of every *sp*-term with large enough depth (exceeding $\mathrm{dp}(S)$) is 0. We formalize this idea in the following theorem, which gives an algebraic characterization of bounded width.

Theorem 3.7 *Let $L \subseteq SP(A)$ be an sp-language. The following properties are equivalent.*

(1) *L has bounded width;*

(2) $\mathrm{Synt}(L)$ *is depth-nilpotent and $0 \notin \mu(L)$, where μ is the syntactic morphism of L;*

(3) *L is recognized by a morphism $\varphi: SP(A) \to S$ into a depth-nilpotent sp-algebra S, such that $0 \notin \varphi(L)$.*

Proof. We first assume that L is recognized by a morphism $\varphi: SP(A) \to S$ into a depth-nilpotent *sp*-algebra S, such that $0 \notin \varphi(L)$. It follows immediately from Lemma 3.6 that, if $\mathrm{dp}(0) = k+1$, and if $s \in S$ and $s \neq 0$, then the *sp*-language $\varphi^{-1}(s)$ consists only of *sp*-terms of depth at most k. Thus L has bounded depth since $0 \notin \varphi(L)$. But Proposition 3.2 shows that bounded-depth *sp*-languages also have bounded width, so (3) implies (1).

Now we assume that L has bounded width. By Proposition 3.2, L is also a bounded-depth language. Let x be an *sp*-term with width (resp. depth) greater than $\mathrm{wd}(L)$ (resp.

$\mathrm{dp}(L)$). Then x cannot be a subterm of an element of L. By Lemma 2.5, we know that $\mathrm{Synt}(L)$ has a zero and that $\mu(x) = 0$. In particular, $0 \notin \mu(L)$.

Contrapositively, if $\mu(x) \neq 0$, then $\mathrm{dp}(x) \leq \mathrm{dp}(L)$. Thus $\mathrm{Synt}(L)$ is depth-bounded. Now let $x \in SP(A)$ be such that $\mu(x) \| \mu(x) = \mu(x)$. If x_n is the n-th parallel power of x, then $\mu(x_n) = \mu(x)$ as well. But if n is large enough, then $\mathrm{wd}(x_n) > \mathrm{wd}(L)$, so $\mu(x_n) = 0$. It follows that $\mu(x) = 0$. So 0 is the only element of $\mathrm{Synt}(L)$ which is idempotent for the parallel product, and hence $\mathrm{Synt}(L)$ is depth-nilpotent. Thus (1) implies (2).

The remaining implication ((2) implies (3)) is immediate, since any language is recognized by its syntactic morphism. □

3.5 Bounded width, recognizability and series-rationality

Since recognizable *sp*-languages have finite syntactic *sp*-algebras, Theorem 3.7 implies:

Corollary 3.8 *Let $L \subseteq SP(A)$ be a sp-language. The following properties are equivalent.*

(1) *L is recognizable and it has bounded width;*

(2) $\mathrm{Synt}(L)$ *is finite, depth-nilpotent and $0 \notin \mu(L)$, where μ is the syntactic morphism of L;*

(3) *L is recognized by a morphism $\varphi\colon SP(A) \to S$ into a finite depth-nipotent sp-algebra S, such that $0 \notin \varphi(L)$.*

The main result of this section completes this statement by connecting bounded-width, recognizability and series-rationality.

Theorem 3.9 *Let L be a sp-language. Then L is series-rational if and only if L is recognizable and bounded-width.*

Proof. It follows from Lemma 1.7 and Corollary 2.10 that series-rational languages are bounded-width and recognizable.

To prove the converse, we use the algebraic characterization of bounded-width recognizable *sp*-languages established in Corollary 3.8, and we assume that L is recognized by $\varphi\colon SP(A) \to S$, a morphism into a finite depth-nilpotent *sp*-algebra.

The result is trivial if $|S| = 1$: then $L = \emptyset$ since $0 \notin \varphi(L)$, and $\emptyset$ is series-rational. Now we assume that S is non-trivial. In particular, $\mathrm{dp}(0) > 1$.

The rest of the proof is an adaptation of McNaughton-Yamada's algorithm for languages recognized by finite automata (e.g. [8, 13]). First we enumerate the elements of S, $S = \{s_1, s_2, \ldots, s_N\}$. Next, for each $s \in S^1$, for each $t \in S$ and for each $0 \leq k \leq N$, we define

$$\begin{aligned} L^k_{s,t} = \{x \in SP(A) \mid \ & \text{the sequential decomposition of } x \text{ is } x = x_1 \cdots x_n \ (n \geq 1), \\ & s\varphi(x) = s\varphi(x_1) \cdots \varphi(x_n) = t, \\ & \text{and } \forall\, 1 \leq i < n,\ \exists\, j \leq k,\ s\varphi(x_1) \cdots \varphi(x_i) = s_j\}. \end{aligned}$$

In particular, $\varphi^{-1}(t) = L^N_{1,t}$. If L is recognized by φ, then L is the union of the $L^N_{1,t}$, where t runs over $\varphi(L)$, so it suffices to show that each $L^k_{s,t}$ is series-rational, when $t \neq 0$.

There will be two nested inductions. The first (outermost) is an induction on the depth of t. The second one is on k. Before we start with the inductions, let us observe that $L^0_{s,t}$ is a set of *sp*-terms whose sequential decompositions have length 1, that is, of parallel terms. Let us also note that if $su = t$ in S, then $\mathrm{dp}(t) \geq \mathrm{dp}(s)$ and $\mathrm{dp}(t) \geq \mathrm{dp}(u)$. In particular, if $\mathrm{dp}(s) > \mathrm{dp}(t)$, then $L^k_{s,t} = \emptyset$ for each k.

Base case: $\mathrm{dp}(t) = 1$**.**

In view of Lemma 3.6, if $\mathrm{dp}(t) = 1$, then $\varphi^{-1}(t) \subseteq A^+$. So $\varphi^{-1}(t)$ is recognized by the semigroup $(S, \circ)$ in the usual sense, so it can be expressed using the letters and the operations of union, sequential product and sequential iteration. In particular, $\varphi^{-1}(t)$ is a series-rational *sp*-language.

Induction step: $\mathrm{dp}(t) > 1$**.**

Let us now assume that for each $s \in S^1$, for each $t \in S$ and for each $0 \leq k \leq N$, the *sp*-language $L^k_{s,t}$ is series-rational whenever $\mathrm{dp}(t) \leq d$.

Let $t \in S$ have depth $d+1$. We know that 0 is the only element of S of maximal depth. So if $\mathrm{dp}(0) = d+1$, then $t = 0$, and we are not concerned with $L^k_{s,t}$. Now we assume that $\mathrm{dp}(0) > d+1$ and $t \neq 0$. We have

$$L^0_{s,t} = \{x \in SP(A) \mid \ x \text{ is a parallel term, and } s\varphi(x) = t\}.$$

Let $x \in L^0_{s,t}$. The parallel decomposition of x is of the form $x = x_1 \| \cdots \| x_n$ $(n \geq 2)$, and $t = s(\varphi(x_1) \| \cdots \| \varphi(x_n))$. Then $t <_{\mathrm{n}} \varphi(x_i)$ for each i by Lemma 3.5, so $\mathrm{dp}(t) > \mathrm{dp}(\varphi(x_i))$. In particular, $\varphi^{-1}\varphi(x_i)$ is series-rational by induction hypothesis. Now we have

$$L^0_{s,t} = \bigcup_{\substack{s(t_1\|\cdots\|t_n)=t \\ n\geq 2}} \varphi^{-1}(t_1) \| \cdots \| \varphi^{-1}(t_n),$$

and this union is finite because $(S, \|)$ is nilpotent: n must be less than the class of nilpotency of $(S, \|)$. Thus $L^0_{s,t}$ is series-rational.

Suppose now that for each u of depth $d+1$ and for each $s \in S^1$, $L^k_{s,u}$ is series-rational, and let t have depth $d+1$. Again we have

$$L^{k+1}_{s,t} = L^k_{s,t} \cup L^k_{s,s_{k+1}} L^k_{s_{k+1},t} \cup L^k_{s,s_{k+1}} (L^k_{s_{k+1},s_{k+1}})^+ L^k_{s_{k+1},t}.$$

If $\mathrm{dp}(s_{k+1}) > \mathrm{dp}(t)$, then $L^k_{s_{k+1},t} = \emptyset$, so $L^{k+1}_{s,t} = L^k_{s,t}$, and hence $L^{k+1}_{s,t}$ is series-rational. If $\mathrm{dp}(s_{k+1}) \leq \mathrm{dp}(t)$, then both $L^k_{s_{k+1},s_{k+1}}$ and $L^k_{s_{k+1},t}$ are series-rational by induction hypothesis, and hence $L^{k+1}_{s,t}$ is series-rational. This concludes the proof. $\square$

In view of the developments of automata theory around the notion of pseudovariety and Eilenberg's theorem (see [8, 21]), it is interesting to note the following results. As this is not the main focus of this paper, we refer the reader to [20] for background on this topic.

Recall that a class of algebras is a *pseudovariety* if it is closed under taking subalgebras, homomorphic images and finite direct products. One easily verifies the following.

Proposition 3.10 *The class of finite depth-nilpotent (resp. bounded-depth) sp-algebras forms a pseudovariety of sp-algebras.*

We observe that for finite *sp*-algebras, the properties of being depth-graded or bounded-depth are equivalent. Our results imply the following: the *sp*-languages recognized by a finite depth-nilpotent *sp*-algebra are the series-rational *sp*-languages and their complements.

To capture the class of series-rational *sp*-languages (which we know is not closed under complement, and hence cannot be characterized using only the syntactic *sp*-algebras), we have to use the notion of *ordered syntactic sp-algebra*, see [20] and [21]. It is not difficult to show that, with the definitions in these articles, the series-rational *sp*-languages are exactly the *sp*-languages whose ordered syntactic *sp*-algebra is depth-nilpotent and is such that 0 is the maximal element (in the syntactic partial order), and that these algebras form a pseudovariety of ordered finite *sp*-algebras.

4 Branching automata

In order to provide an automaton characterization of series-rational *sp*-languages, we introduce a new kind of automata, with branch and join features.

4.1 Definition and operation

A *branching automaton* over the alphabet A is a tuple $\mathcal{A} = (Q, T_{seq}, T_{fork}, T_{join}, I, F)$ where Q is the (finite) set of states, I and F are subsets of Q, respectively the set of initial and of final states, $T_{seq} \subseteq Q \times A \times Q$ is the set of sequential transitions, $T_{fork} \subseteq Q \times \mathcal{M}_{ns}(Q)$ and $T_{join} \subseteq \mathcal{M}_{ns}(Q) \times Q$ are respectively the sets of fork and join transitions. Here $\mathcal{M}_{ns}(Q)$ ("nonempty and nonsingleton") stands for multisets over Q of cardinality at least 2.

An element $(q, a, q') \in T_{seq}$ is said to be an a-labelled transition from q to q'. As usual, we write $q \xrightarrow{a} q'$.

An element $(q, \{q_1, \ldots, q_n\})$ of T_{fork} is said to be a *fork transition of arity n*. We denote it by $q \rightarrow \{q_1, \ldots, q_n\}$. Similarly, an element $(\{q_1, \ldots, q_n\}, q)$ of T_{join}, written $\{q_1, \ldots, q_n\} \rightarrow q$, is said to be a *join transition of arity n*.

Let p, q be states of a branching automaton $\mathcal{A}$ and let t be a *sp*-term. The existence of a *run of $\mathcal{A}$ on t from p to q* is inductively defined as follows: there is a run on letter a from p to q if $p \xrightarrow{a} q$. If a term t has sequential factorization $t_1 \ldots t_n$ $(n \geq 2)$, there is a run on t from p to q if there exist states $p = p_0, p_1, \ldots p_n = q$ and there exists a run on t_m from p_{m-1} to p_m, for each $1 \leq m \leq n$.

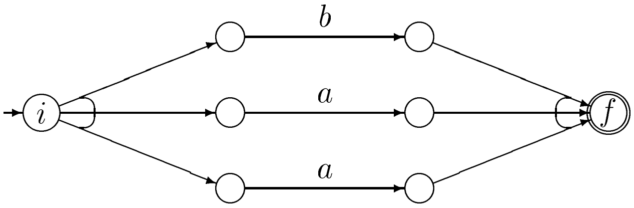
Suppose now that a term t has parallel factorization $t_1||\dots||t_n$ $(n \geq 2)$. There is a run on t from p to q if there is a fork at p, sub-runs for the factors and then a matching join ending at q. In order to handle arbitrarily long parallel products with finitely many fixed arity fork transitions, we allow the automaton to do the forking in levels, as follows:

Let $\pi = \{r_1, \dots, r_s\}$ be a subset of $\{1, \dots, n\}$. By t_π we mean the parallel term $t_{r_1}||\dots||t_{r_s}$. Now we say there is a run on t from p to q if there exists a partition $\{\pi_1, \dots, \pi_m\}$ of $\{1, \dots, n\}$ and there are states p_ℓ, q_ℓ, for $1 \leq \ell \leq m$ such that there is a fork $k = p \to \{p_1, \dots, p_m\}$, there are runs on t_{π_ℓ} from p_ℓ to q_ℓ, $1 \leq \ell \leq m$, and there is a join $j = \{q_1, \dots, q_m\} \to q$.

We say that the fork transition k and the join transition j are *matched* in this run, and that the states and the transitions forming the sub-runs on the t_{π_ℓ} are *nested within* the matched pair (k, j) in the run on t.

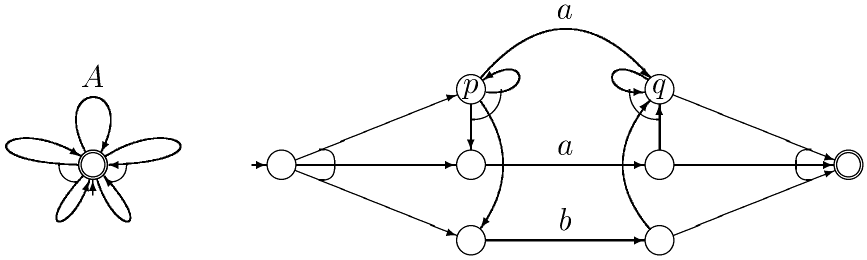
Finally, a branching automaton $\mathcal{A}$ *accepts* the *sp*-term t if there exists a run of $\mathcal{A}$ on t from an initial state to a final state. An *sp*-language L is *regular* if it is the set of *sp*-terms accepted by a finite branching automaton $\mathcal{A}$. We say that $\mathcal{A}$ *accepts* L and we write $L = L(\mathcal{A})$.

Example 4.1 The following automaton has a run on $a||a||b$ from state i to state f. The partition of $\{1, 2, 3\}$ here is simple: it just has three singletons $\{\{1\}, \{2\}, \{3\}\}$. The language accepted is $\{a||a||b\}$.



As in the case of recognizable languages (Section 2), the notion of regularity extends beyond series-rational languages. It is easy to find a branching automaton accepting the non-bounded-width language $a^\oplus$. Here are other, more complex examples of regular, non-series-rational *sp*-languages.

Example 4.2 The following automata accept respectively $SP(A)$ and $a||(a||b)^\oplus$ (parallel products of a and b having one more a than b).



4.2 Fork-acyclic automata

Note the looping fork at state p and the looping join at state q in the automaton of Example 4.2. It is the presence of a loop from inside one branch of a fork to another (or to outside the fork) that allows unbounded parallel-width terms to be accepted. To allow an automaton to accept a bounded width *sp*-language, we rule out looping forks and joins, as follows.

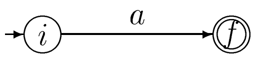
A run of $\mathcal{A}$ on t is *fork-acyclic* if, for any pair (k, j) of a fork and a join transition, (k, j) does not occur as a matched pair nested within itself. If all the successful runs of $\mathcal{A}$ (i.e. the runs from an initial state to a final state) are fork-acyclic, we call the automaton *fork-acyclic*. An *sp*-language is *fork-regular* if it is accepted by a finite fork-acyclic branching automaton. We characterize the fork-regular *sp*-languages in Section 4.4.

Example 4.3 The automaton of Example 4.1 is fork-acyclic, and those of Example 4.2 are not. In the second automaton of the latter example, the only fork-acyclic runs from the initial state to the final state are labelled $a\|a\|b$ and $a\|a\|a\|b\|b$. In all other runs, the pair consisting of the fork transition out of state p and the join transition into state q occurs nested within itself.

Any (usual) automaton on words, having no fork transitions and recognizing a set of sequential words, is clearly fork-acyclic. In particular, every regular language of A^+ is a regular *sp*-language.

Let us describe particular fork-acyclic branching automata which can be associated with each *sp*-term $x \in SP(A)$. Their construction is inductive and, for each x, the automaton $\mathcal{A}(x)$ has a single initial state and a single final state.

If $x = a$ $(a \in A)$, then $\mathcal{A}(x)$ has two states, i and f, i is initial and f is final, there is one sequential transition, namely $i \xrightarrow{a} f$, and there are no fork or join transitions.

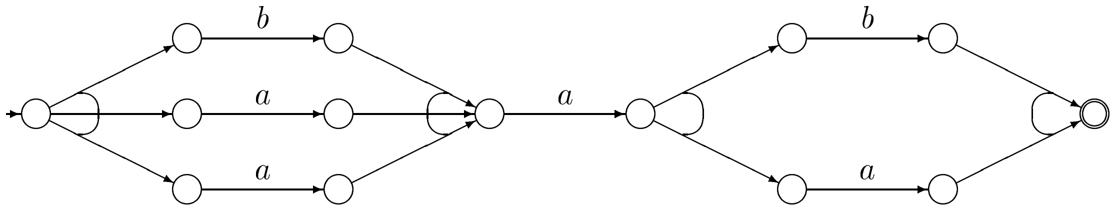


If x is a parallel term, with parallel factorization $x = x_1\|x_2\|\cdots\|x_n$ $(n \geq 2)$, then $\mathcal{A}(x)$ is obtained by taking the union of the $\mathcal{A}(x_m)$ (with initial state i_m and final state f_m), by adding two new states i and f, chosen to be respectively initial and final, and by adding a new fork transition $i \to \{i_1, \ldots, i_n\}$ and a new join transition $\{f_1, \ldots, f_n\} \to f$. For instance, the automaton in Example 4.1 is exactly $\mathcal{A}(a\|a\|b)$.

If x is a sequential term, with sequential factorization $x = x_1x_2\cdots x_n$ $(n \geq 2)$, then $\mathcal{A}(x)$ is obtained by taking the union of the $\mathcal{A}(x_m)$, and by identifying the states f_m and i_{m+1}, for each $1 \leq m < n$. The initial state of $\mathcal{A}(x)$ is chosen to be i_1 and its final state is f_n.

It is easy to see that, in this inductive construction, the initial state participates in exactly one transition, and is the start of either a fork or a sequential transition. Dually, the final state participates in exactly one transition, and is the end of either a join or a sequential transition. Every other state participates in exactly two transitions, once as a start state and once as an end state. Also one can verify that each $\mathcal{A}(x)$ is fork-acyclic.

Example 4.4 The following automaton is $\mathcal{A}((a\|a\|b)a(a\|b))$.



A key property of fork-acyclic automata is that they suffice to accept bounded width regular languages. This is shown by the next result, which proves a pumping property with respect to parallel depth. Its converse is established in Section 4.4.

Proposition 4.5 *If the sp-language accepted by a branching automaton $\mathcal{A}$ has bounded width, then $\mathcal{A}$ is fork-acyclic.*

Proof. Let L be the language accepted by $\mathcal{A}$ and let us assume that there exists an element $x \in L$ and a successful run ρ of $\mathcal{A}$ on x which is not fork-acyclic. Then there exists a pair (f, j) of matched fork and join transitions which occurs nested within itself in that run.

More precisely, there exists a parallel subterm y of x and a parallel subterm z of y such that $z \neq y$, the run ρ on x uses a sub-run ρ' on y starting with transition f and ending with transition j (an (f, j)*-run* for short), and the run ρ' itself uses an (f, j)-run ρ'' on z.

Let c and d be ξ-terms such that $y = c[\xi, z]$ and $x = d[\xi, y]$. Note that if t is any parallel term which labels an (f, j)-run, then $c[\xi, t]$ also labels an (f, j)-run and $d[\xi, t] \in L$.

Let $y_0 = z$. The sequences $(y_n)_n$ and $(x_n)_n$ are defined by letting $y_{n+1} = c[\xi, y_n]$ and $x_n = d[\xi, y_n]$. By induction, each y_n labels an (f, j)-run and each x_n lies in L. Since y is a parallel term, c is a parallel product of at least 2 factors, so $\mathrm{dp}(y_{n+1}) > \mathrm{dp}(y_n)$ and $\mathrm{dp}(x_{n+1}) > \mathrm{dp}(x_n)$. This shows that L does not have bounded-depth, a contradiction in view of Proposition 3.2. □

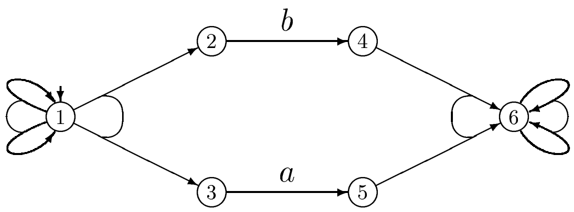
4.3 Properties of regular *sp*-languages

The main theorem of this section is the following.

Theorem 4.6 *The class of regular sp-languages is closed under union, intersection, inverse and direct image by a morphism, sequential product and iteration, and parallel product and iteration.*

Some of these closure properties can be proved by simple and classical constructions. This is the case for the operations of union, intersection, and inverse and direct image by a morphism, for which we will give no proof: the reader is refered to the analogous results on word languages, in [13].

It is tempting to continue adapting the classical (sequential) automata-theoretic constructions to prove the rest of Theorem 4.6. But consider the following automaton, which recognizes $(a\|b)^{\oplus}$.



To apply the "obvious" analogue of the classical construction for the sequential product of languages to two copies of this automaton, say $\mathcal{A}$ and $\mathcal{A}'$ with respective state sets $\{1,\ldots,6\}$ and $\{1',\ldots,6'\}$, we let 1 be initial and $6'$ be final, and we add duplicates of the fork transitions out of $1'$, originating at 6. There are now runs on $a\|b$ from 1 to 6 and from 6 to $6'$, so that there are runs on $(a\|b)(a\|b)$ from 1 to $6'$. Moreover we have a fork transition $1 \to \{1,1\}$ and a join transition $\{6',6'\} \to 6'$. Thus the parallel product of 2 copies of $(a\|b)(a\|b)$ is accepted by this automaton, whereas it does not lie in $(a\|b)^{\oplus}(a\|b)^{\oplus}$.

This leads us to the introduction of a special class of branching automata. We say that an automaton is *misbehaved* if there exists a fork transition $q \to \{q_1,\ldots,q_n\}$ such that there are runs from each q_i to some final state, or if there exists a join transition $\{q_1,\ldots,q_n\} \to q$ such that for each i there are runs from some initial state to q_i. The automaton is said to be *behaved* otherwise.

Example 4.7 All automata without fork or join transitions are trivially behaved. The automata of the form $\mathcal{A}(x)$ ($x \in SP(A)$) are behaved. The automaton accepting $(a\|b)^{\oplus}$ given above is misbehaved, since $1 \to \{1,1\}$ is a fork transition and there is a run (on $a\|b$) from 1 to 6, the final state. In the next proposition, we show that being behaved or misbehaved is a property of the automaton, not of the language, and that there exists a behaved automaton accepting $(a\|b)^{\oplus}$.

We will see that when we use behaved automata, then the classical constructions for the sequential product and iteration of sequential languages can be extended to handle *sp*-languages. First we prove that behaved automata have the same accepting power as branching automata.

Proposition 4.8 *Every regular sp-language is accepted by a behaved automaton.*

Proof. Let $\mathcal{A}$ be a branching automaton and let L be the *sp*-language accepted by $\mathcal{A}$. For each fork transition $k = q \to \{q_1,\ldots,q_r\}$, we distinguish (arbitrarily) one of the receiving end states, say, that which is denoted q_1.

We construct a new automaton $\mathcal{B}$ as follows. First we take the disjoint union of 2 copies of $\mathcal{A}$, denoted respectively $\mathcal{A}_0$ and $\mathcal{A}_1$. Next we delete from $\mathcal{A}_0$ all the fork and join transitions. For each fork transition $k = q \to \{q_1,\ldots,q_r\}$ of $\mathcal{A}$, we add to $\mathcal{B}$ a fork transition $q \to \{q_1, \ldots, q_r\}$ where $q, q_2, \ldots, q_r$ are taken in $\mathcal{A}_0$ and q_1 is taken in $\mathcal{A}_1$. Moreover, for each fork transition k of arity r and for each join transition of the same arity, say $j = \{p_1, \ldots, p_r\} \to p$, we add r join transitions $j_{k,m}$ $(1 \leq m \leq r)$ simulating j, with p_m taken in $\mathcal{A}_1$, and with p and the p_h $(h \neq m)$ taken in $\mathcal{A}_0$. Finally, the initial and final states of $\mathcal{B}$ are the same as in $\mathcal{A}$, but taken in $\mathcal{A}_0$. (It is a good exercise for the reader to draw automaton $\mathcal{B}$ if $\mathcal{A}$ is the automaton accepting $(a\|b)^{\oplus}$ given above.)

The only way out of $\mathcal{A}_1$ is through one of these join transitions, only one of whose branches originates in $\mathcal{A}_1$. Thus the runs of $\mathcal{B}$ starting in a state of $\mathcal{A}_1$, end in a state of $\mathcal{A}_1$. In particular, there is no run from a state of $\mathcal{A}_1$ to a final state. Since every fork transition has at least one of its end states in $\mathcal{A}_1$, it follows that $\mathcal{B}$ satisfies the fork condition in the definition of behavedness.

Dually, the only way into $\mathcal{A}_1$ is through a fork transition, only one of whose branches ends in $\mathcal{A}_1$. On the other hand, the joins ending in $\mathcal{A}_1$ have all their start states in $\mathcal{A}_1$. Thus the runs of $\mathcal{B}$ starting in a state of $\mathcal{A}_0$, end in a state of $\mathcal{A}_0$. Since every join transition has at least one of its start states in $\mathcal{A}_1$, $\mathcal{B}$ satisfies the join condition in the definition of behavedness, and hence $\mathcal{B}$ is behaved.

To conclude the proof, we need to show that $\mathcal{A}$ and $\mathcal{B}$ accept the same *sp*-language. By construction, if x is a letter, then there is a run of $\mathcal{A}$ on x from state p to state q if and only if there is one in $\mathcal{B}$, with p and q taken in $\mathcal{A}_0$.

Let us assume that $x, y \in SP(A)$ are such that there is a run of $\mathcal{A}$ on x (resp. y) from state p to state q if and only if there is one in $\mathcal{B}$, with p and q taken in $\mathcal{A}_0$. If there is a run of $\mathcal{A}$ on xy from p to q, then there is a run of $\mathcal{A}$ on x from p to some state r, and one on y from r to q. Therefore there is a run of $\mathcal{B}$ on x from p to r and one on y from r to q with p, r, q taken in $\mathcal{A}_0$. Thus there is a run of $\mathcal{B}$ on xy from p to q, with p and q taken in $\mathcal{A}_0$.

Conversely, if $\mathcal{B}$ has a run on x from p to r and a run on y from r to q with p and q taken in $\mathcal{A}_0$, then r must also be in $\mathcal{A}_0$ since there are no runs from $\mathcal{A}_1$ to $\mathcal{A}_0$. Thus $\mathcal{A}$ has runs on x from p to r and on y from r to q, that is, $\mathcal{A}$ has a run on xy from p to q.

Now we assume that $x_1, \ldots, x_n$ $(n \geq 1)$ are sequential terms such that for each i, there is a run of $\mathcal{A}$ on x_i from state p to state q if and only if there is one in $\mathcal{B}$, with p and q taken in $\mathcal{A}_0$. We proceed by induction on n to show that there exists a run of $\mathcal{A}$ on $x_1\|\cdots\|x_n$ from p to q, if and only if there is one in $\mathcal{B}$ from p to q with p and q taken from $\mathcal{A}_0$. The case where $n = 1$ is trivial. Now let $n \geq 2$. If there is a run of $\mathcal{A}$ on $x_1\|\cdots\|x_n$ from p to q, then there exists an integer $2 \leq r \leq n$, a fork transition $k = p \to \{p_1, \ldots, p_r\}$, a join transition $\{q_1, \ldots, q_r\} \to q$ and terms $y_1, \ldots, y_r$ with $x_1\|\cdots\|x_n = y_1\|\cdots\|y_r$, such that there exists a run of $\mathcal{A}$ on y_i from p_i to q_i. Note that each y_i is a parallel product of at most $n-1$ of the x_h. By induction, there exists a run of $\mathcal{B}$ on y_i from p_i to q_i, with p_i and q_i taken from $\mathcal{A}_0$. Now $\mathcal{B}$ has a fork transition $p \to \{p_1, \ldots, p_r\}$ with $p, p_2, \ldots, p_r$ taken from $\mathcal{A}_0$ and p_1 taken from $\mathcal{A}_1$. It also has the join transition $\{q_1, \ldots, q_r\} \to q$ with $q, q_2, \ldots, q_r$ taken from $\mathcal{A}_0$ and q_1 taken from $\mathcal{A}_1$. Finally, since $\mathcal{A}_1$ is a copy of $\mathcal{A}$, it has a run on y_1 from p_1 to q_1. It follows that $\mathcal{B}$ has a run on $x_1\|\cdots\|x_n$ from p to q, with p and q taken from $\mathcal{A}_0$.

Let us assume conversely that $\mathcal{B}$ has a run on $x_1\|\cdots\|x_n$ from p to q, with p and q taken from $\mathcal{A}_0$. In view of the definition of $\mathcal{B}$, there exist a fork transition $k = p \to \{p_1, \ldots, p_r\}$ and a join transition $\{q_1, \ldots, q_r\} \to q$ in $\mathcal{A}$, and there exist terms $y_1, \ldots, y_r$ with $x_1\|\cdots\|x_n = y_1\|\cdots\|y_r$ such that $r \geq 2$ and there exist runs of $\mathcal{B}$ on y_i $(i > 2)$ from p_i to q_i with p_i and q_i taken from $\mathcal{A}_0$, and there exists a run of $\mathcal{B}$ on y_1 from p_1 to q_1. By induction, there exist runs of $\mathcal{A}$ on y_i $(i > 2)$ from p_i to q_i. Moreover we observed that, since p_1 and q_1 are both in $\mathcal{A}_1$, the run of $\mathcal{B}$ from p_1 to q_1 is a run of $\mathcal{A}_1$. Since $\mathcal{A}_1$ is a copy of $\mathcal{A}$, there also exists a run of $\mathcal{A}$ on y_1 from p_1 to q_1. Thus there exists a run of $\mathcal{A}$ on $x_1\|\cdots\|x_n$ from p to q.

So $\mathcal{A}$ and $\mathcal{B}$ accept the same *sp*-language, which concludes the proof. □

We now continue the proof of Theorem 4.6. The next closure property we consider is an extension of the substitutions considered in Section 1.4. Let K, L be *sp*-languages, and let $a \in A$. We denote by $L[a, K]$ the set of *sp*-terms obtained by replacing in an element of L each occurrence of letter a by an *sp*-term in K (non-uniformly).

Proposition 4.9 *Let K and L be regular sp-languages over the alphabet A and let $a \in A$. Then $L[a, K]$ is regular.*

Proof. Let $\mathcal{A}$ be an automaton accepting L and let $\mathcal{B}$ be a behaved automaton accepting K. We construct a new automaton $\mathcal{C}$ as follows. For each a-labelled sequential transition $p \xrightarrow{a} q$ in $\mathcal{A}$, we construct a new copy of $\mathcal{B}$, denoted by $\mathcal{B}_{p,q}$, and we consider the disjoint union of $\mathcal{A}$ and the $\mathcal{B}_{p,q}$. For each transition $p \xrightarrow{a} q$ in $\mathcal{A}$, we delete this transition in $\mathcal{A}$ and we add duplicates of all the sequential and fork transitions of $\mathcal{B}_{p,q}$ originating in an initial state, with p substituted for that initial state on the left-hand side. Dually, we add duplicates of all the sequential or join transitions of $\mathcal{B}_{p,q}$ ending in a final state, with q substituted for that final state on the right-hand side. In addition, if there is a letter b such that there exists a b-labelled transition from an initial state of $\mathcal{B}$ to a final state, then we add a b-labelled transition from p to q. Finally, we keep as initial (resp. final) states the initial (resp. final) states of $\mathcal{A}$.

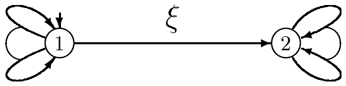
Because $\mathcal{B}$ is behaved, we are certain that the runs of $\mathcal{C}$ which start in state p with a transition into $\mathcal{B}_{p,q}$ either are entirely contained within $\mathcal{B}_{p,q}$, or visit state q and their label is of the form xy where $x \in K$. It is now immediate that $\mathcal{C}$ accepts exactly $L[a, K]$. □

Remark. Let L be a regular *sp*-language, let $\mathcal{A}$ be a behaved automaton accepting L, and let ξ be a letter not in A. Then $\{\xi\}[\xi, L] = L$. The construction in the proof of Proposition 4.9 provides a new behaved automaton accepting L, with a single initial state which is a source, and with a single final state which is a sink.

By digressing to prove Propositions 4.8 and 4.9, we now have easy proofs of the remaining closure properties required for Theorem 4.6.

Proposition 4.10 *The class of regular sp-languages is closed under sequential product, sequential iteration, parallel product and parallel iteration.*

Proof. Let L, L' be regular *sp*-languages in $SP(A)$, and let $\zeta, \xi \notin A$. The *sp*-languages $\{\zeta\xi\}$ and ξ^+ are regular, like all regular sequential languages (regular subsets of the free semigroup). The *sp*-language $\{\zeta\|\xi\}$ is regular too, it is accepted by $\mathcal{A}(\zeta\|\xi)$. Finally, the *sp*-language $\xi^\oplus$ is regular, it is accepted by the following (misbehaved) automaton:



Now we have

$$L^+ = \xi^+[\xi, L],\ \ L^\oplus = \xi^\oplus[\xi, L],\ \ LL' = (\{\zeta\xi\}[\xi, L'])[\zeta, L]\ \text{ and }\ L\|L' = (\{\zeta\|\xi\}[\xi, L'])[\zeta, L].$$

It follows from Proposition 4.9 that L^+, $L^\oplus$, LL' and $L\|L'$ are regular. □

Corollary 4.11 *The series-parallel-rational sp-languages are regular.*

Proof. This follows from the above closure properties and from the fact that the *sp*-languages of the form $\{a\}$ ($a \in A$) are regular. □

4.4 Series-rationality and fork-regularity

We now return to fork-regular *sp*-languages, and give a new characterization of the series-rational languages which complements the results of Section 3.5.

Theorem 4.12 *Let $L \subseteq SP(A)$ be a sp-language. The following properties are equivalent.*

(1) *L is regular and bounded-width.*

(2) *L is fork-regular.*

(3) *L is series-rational.*

Proof. For each $x \in SP(A)$, the automaton $\mathcal{A}(x)$ defined in Section 4.2 and accepting $\{x\}$ is easily seen to be fork-acyclic and behaved. Next one verifies easily that the automata constructions given in the previous sections for sequential and parallel products, and sequential iterations, preserve fork-acyclicity and behavedness. The same is true for the classical construction of an automaton accepting the union $L \cup L'$, namely taking the disjoint union of automata accepting respectively L and L'. Thus series-rational *sp*-languages are accepted by fork-acyclic automata and (3) implies (2). It follows that (3) also implies (1).

Proposition 4.5 showed that (1) implies (2). The last implication, (2) implies (3), is obtained using a nontrivial extension of the McNaughton-Yamada construction for finite automata.

Let $\mathcal{A}$ be a fork-acyclic branching automaton and let L be the *sp*-language accepted by $\mathcal{A}$. Let x be a *sp*-term and let $x = x_1 \cdots x_k$ be its sequential factorization. Let ρ be a run of $\mathcal{A}$ on x. We say that a pair (f, j) of a fork transition f and a join transition j of $\mathcal{A}$ is used *at the upper level* by ρ if the sub-run of ρ on some x_i starts with transition f and ends with transition j. If $D, E \subseteq T_{\text{fork}} \times T_{\text{join}}$ are sets of pairs of a fork and a join transition in $\mathcal{A}$, such that $D \subseteq E$, and if $p, q \in Q$, we let $L_{p,q}^{D,E}$ be the set of labels of runs of $\mathcal{A}$ from p to q which use only matched pairs in E, and only matched pairs in D at the upper level.

We say that the pair of states (p, q) is *admissible* if there exists a successful run of $\mathcal{A}$ which contains a sub-run from p to q. If that is the case, then every run of $\mathcal{A}$ from p to q is a sub-run of some successful run. We build up inductively series-rational expressions for the $L_{p,q}^{D,E}$ when (p, q) is an admissible pair. The induction is on $|E|$.

First let us note that if the fork transition f and the join transition j do not have the same arity, then (f, j) cannot occur as a matched pair in a run. Similarly, if r is the start state of f, if s is the end state of j and if (r, s) is not an admissible pair of states, then (f, j) cannot occur as a matched pair within a successful run, and hence it cannot occur as a matched pair within a run from p to q whenever (p, q) is admissible. So we can remove from the sets D and E all pairs (f, j) of transitions with different arity, or such that the pair of the initial state of f and the final state of j is not admissible.

When $|E| = 0$, no forks are used, so $L_{p,q}^{\emptyset,\emptyset}$ is the usual McNaughton-Yamada expression, that is, an expression using only unions, sequential products and sequential iterations.

We now consider $L_{p,q}^{D,E}$ for some non-empty set E. We proceed by induction on $|D|$. If $|D| = 0$, again no forks are used so $L_{p,q}^{\emptyset,E}$ is a rational language of sequential words.

For the induction step, let $(f, j) \in D$ with $f = r \to \{r_1, \ldots, r_n\}$ and $j = \{s_1, \ldots, s_n\} \to s$. Let $L(E, f, j)$ be the set of labels of runs of $\mathcal{A}$ on parallel terms which use only matched pairs in E, which start with transition f and end with transition j. In particular, (f, j) is the only matched pair occurring at the upper level in these runs, so $L(E, f, j)$ is contained in $L_{r,s}^{D,E}$.

The runs from p to q using only matched pairs in E, and only matched pairs in D at the upper level, either do not use (f, j) at the upper level, or use it a finite, linearly ordered number of times. All but the last sub-run on a parallel term from transition f to transition j (i.e. sub-runs in $L(E, f, j)$) are followed (sequentially) by a run from s to state r using only matched pairs in E, and only matched pairs in $D - \{(f, j)\}$ at the upper level, and the last sub-run in $L(E, f, j)$ is followed by such a run from s to q. Finally, each sub-run in $L(E, f, j)$ is on a parallel term, and decomposes into a parallel product of runs using only matched pairs in $E - \{(f, j)\}$ (by fork-acyclicity), from one of the end states of f to one of the start states of j. This is expressed as follows.

$$L(E, f, j) \;=\; \bigcup_{\sigma \in S_n} \Big(L_{r_1, s_{\sigma(1)}}^{E-\{(f,j)\}, E-\{(f,j)\}} \| \cdots \| L_{r_n, s_{\sigma(n)}}^{E-\{(f,j)\}, E-\{(f,j)\}} \Big).$$

Here S_n denotes the permutation group over $\{1, \ldots, n\}$ elements. Next we have

$$L_{p,q}^{D,E} \;=\; L_{p,q}^{D-\{(f,j)\},E} \;\cup\; L_{p,r}^{D-\{(f,j)\},E} L(E, f, j) \Big(L_{s,r}^{D-\{(f,j)\},E} L(E, f, j) \Big)^* L_{s,q}^{D-\{(f,j)\},E}.$$

Note that if $\sigma \in S_n$, if all the $L^{E-\{(f,j)\},E-\{(f,j)\}}_{r_m,s_{\sigma(m)}}$ are non-empty, then the pairs $(r_m, s_{\sigma(m)})$ are admissible, since (r,s) is admissible. Similarly, if $L_{p,r}^{D-\{(f,j)\},E}$, $L(E,f,j)$ and $L_{s,q}^{D-\{(f,j)\},E}$ are non-empty, then (p,r) and (s,q) are admissible pairs. If in addition $L_{s,r}^{D-\{(f,j)\},E}$ is non-empty, then (s,r) is admissible.

Thus $L_{p,q}^{D,E}$ is series-rational by induction. Finally L is the union of the $E_{i,k}^{T_{fork} \times T_{join}}$, where (i,k) runs over all admissible pairs of an initial and a final state, so L is series-rational. $\square$

Say that an automaton is *strongly fork-acyclic* if all its runs are fork-acyclic. The first part of the proof of Theorem 4.12 above actually implies the following.

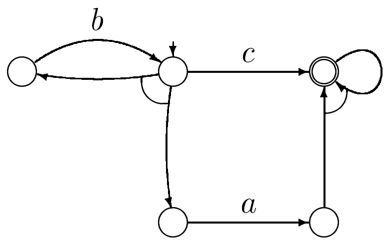
Corollary 4.13 *Let $L \subseteq SP(A)$ be an sp-language. Then L is series-rational if and only if L is accepted by a strongly fork-acyclic automaton.*

5 Unbounded width languages

We have proved that the three formalisms of algebraic recognizability, rational expressions and automata recognizability coincide in the context of bounded width *sp*-languages. Some properties of the classes of recognizable and regular *sp*-languages were also discussed in the paper.

When the bounded width restriction is dropped, these formalisms do not coincide any more. The class of series-parallel-rational *sp*-languages is incomparable with the recognizable languages, as the following example shows.

The language of Example 2.2 (the least language L containing letter c and such that if $x \in L$, then $a||(bx) \in L$) was shown there to be recognizable, yet not series-parallel-rational. However L is regular, as it is accepted by the following automaton.



But Corollary 4.11 showed that the series-parallel-rational languages are regular. In a forthcoming paper [15], working in a more general setting, we show that the class of recognizable languages is again included in the regular languages. That the inclusion is strict is witnessed by the regular language $(a||b)^{\oplus}$, which is not recognizable.

It remains open to characterize each of the smaller classes inside the larger ones. More precisely, it would be interesting to have necessary and sufficient conditions on a branching automaton $\mathcal{A}$ for $L(\mathcal{A})$ to be recognizable, or series-parallel-rational. Finally, we would be very interested in finding logical characterizations of the different classes of *sp*-languages which we considered.

Acknowledgements. The authors thank R. Ramanujam for discussion, and the anonymous referees for their comments. In particular, one referee suggested simplifications of the proofs of Lemma 2.7 and Proposition 4.8, which we incorporated.

References

[1] S. Bloom and Z. Ésik. Free shuffle algebras in language varieties, *Theoret. Comp. Sci.* **163** (1996), 55–98.

[2] J.R. Büchi. *Finite automata, their algebras and grammars: Towards a theory of formal expressions* (D. Siefkes, ed.), Springer (1989).

[3] B. Courcelle. On recognizable sets and tree automata, in *Resolution of equations in algebraic structures* (M. Nivat, H. Aït-Kaci eds.), Academic Press (1989) 93–126.

[4] B. Courcelle. Graph rewriting: an algebraic and logical approach, in *Handbook of Theoretical Computer Science* **B** (J. van Leeuwen, ed.), Elsevier (1990) 193–242.

[5] V. Diekert and G. Rozenberg. *The book of traces*, World Scientific (1995).

[6] J. Doner. Tree acceptors and some of their applications, *J. Comp. Syst. Sci.* **4** (1970) 406–451.

[7] M. Droste and P. Gastin. Asynchronous cellular automata for pomsets without autoconcurrency, in *CONCUR*, Pisa (U. Montanari, V. Sassone, eds.), *Lect. Notes Comp. Sci.* **1119**, Springer (1996) 627–638.

[8] S. Eilenberg. *Automata, languages and machines* **A**, Academic Press (1974).

[9] V.K. Garg and M.T. Ragunath. Concurrent regular expressions and their relationship to Petri nets, *Theoret. Comp. Sci.* **96** (1992) 285–304.

[10] F. Gécseg and M. Steinby. *Tree automata*, Akadémiai Kiadó, Budapest (1984).

[11] J.L. Gischer. The equational theory of pomsets, *Theoret. Comp. Sci.* **61** (1988) 199–224.

[12] J. Grabowski. On partial languages, *Fund. Inform.* **IV** (1981) 427–498.

[13] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computation*, Addison Wesley (1979).

[14] G. Lallement. *Semigroups and combinatorial applications*, Wiley, New York (1979).

[15] K. Lodaya and P. Weil. Rationality in algebras with a series operation, IMSc Report **99/11/39** (1999). `http://www.imsc.ernet.in/~kamal/ssig.ps.gz`

[16] A. Mazurkiewicz. Concurrent program schemes and their interpretations, DAIMI Rep. **PB 78**, Aarhus University (1977).

[17] J. Mezei and J. Wright. Algebraic automata and context-free sets, *Inf. Contr.* **11** (1967) 3–29.

[18] M. Nielsen, G.D. Plotkin and G. Winskel. Petri nets, event structures and domains, Part I, *Theoret. Comp. Sci.* **13** (1981) 85–108.

[19] E. Ochmański. Regular behaviour of concurrent systems, *Bull. EATCS* **27** (1985) 56–67.

[20] J.-E. Pin, Syntactic semigroups, in *Handbook of Formal Language Theory* **I** (G. Rozenberg and A. Salomaa, eds.), Springer (1997) 679–746.

[21] J.-E. Pin and P. Weil. A Reiterman theorem for pseudovarieties of finite first-order structures, *Algebra Universalis* **35** (1996) 577–595.

[22] V. Pratt. Modelling concurrency with partial orders, *Int. J. Parallel Prog.* **15**(1) (1986) 33–71.

[23] W. Reisig and G. Rozenberg (eds). *Lectures on Petri nets* (2 vols), *Lect. Notes Comp. Sci.* **1491–2**, Springer (1998).

[24] M. Steinby. Some algebraic aspects of recognizability and rationality, in *Fundamentals of Computation Theory*, Szeged (F. Gécseg ed.), *Lect. Notes Comp. Sci.* **117**, Springer (1981) 360–372.

[25] J.W. Thatcher and J.B. Wright. Generalized finite automata with an application to a decision problem of second order logic, *Math. Syst. Theory* **2** (1968) 57–82.

[26] J. Valdes, R.E. Tarjan and E.L. Lawler. The recognition of series-parallel digraphs, *SIAM J. Comput.* **11** (1981) 298–313.

[27] J. Winkowski. Algebras of partial sequences – a tool to deal with concurrency, in *Fundamentals of Computation Theory*, Poznań-Kórnik (M. Karpiński, ed.), *Lect. Notes Comp. Sci.* **56** (1977) 187–196.

[28] W. Zielonka, Notes on finite asynchronous automata, *RAIRO Inf. Théorique et Appl.* **21** (1987) 99–135.