Branching Automata and Pomset Automata

Nicolas Bedon
LITIS (EA 4108), University of Rouen, France

Abstract

We compare, in terms of expressive power, two notions of automata recognizing finite N-free pomsets: *branching automata* by Lodaya and Weil [7, 8, 9, 10] and *pomset automata* by Kappé, Brunet, Luttik, Silva and Zanasi [5]. In the general case, they are equivalent. We also consider sub-classes of both kind of automata that we prove equivalent.

2012 ACM Subject Classification Theory of computation → Regular languages

Keywords and phrases Finite N-free Pomsets, Finite Series-Parallel Pomsets, Branching Automata, Pomset Automata, Series-Parallel Rational Languages

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.37

Acknowledgements The author would like to thank the anonymous referees of this paper, whose comments helped in improving its quality.

1 Introduction

Automata are among the main tools in theoretical computer science. They are at the center of a large number of theoretical results and practical applications. Among them, let us cite as examples pattern matching, lexical analysis in compilers, and model-checking. In the latter, automata are used both for modeling sequential processes and to represent logical specifications. A state of an automaton represents a state of the system that is modeled, and transitions are used to change from states to states when an event occurs or an instruction is executed.

Inputs of automata as they were originally defined by Kleene [6] are finite words, that naturally model finite totally ordered sequences of events. Mainly motivated by the use of automata as a key argument in decidability algorithms in formal logic and circuits modeling, automata have quickly been extended to more complex inputs, such as for example infinite (ω) and transfinite words, terms, finite and infinite trees.

In this paper we focus on automata for languages of finite series-parallel pomsets. Informally speaking, a pomset is a word in which the total ordering of elements is not required. When A is an alphabet, finite words over A are the elements freely generated by A in the variety of monoids, and finite series-parallel pomsets over A are the elements freely generated by A in the variety of algebras $(X, \cdot, \|)$, with $(X, \cdot)$ a monoid and $(X, \|)$ a commutative monoid. Series-parallel pomsets have natural applications in computer science: when words are though of as traces of sequential executions of programs, series-parallel pomsets are traces of concurrent programs in which concurrency relies of the fork/join principle: a process forks into several concurrent parallel processes, waits for all of them to end their executions, and then continues its run. The class of series-parallel pomsets have an interesting characterisation in terms of sub-ordering: it coincides with that of N-free pomsets [11, 12].

In [7, 8, 9, 10], Lodaya and Weil introduced a class of automata on finite N-free pomsets, named *branching automata*, that extends Kleene automata with two kinds of unlabeled transitions: the *fork* and *join* transitions. A fork transition splits a path into several paths that run in parallel. When they are all finished, those parallel paths are grouped together with a join transition that goes into a single state. This join transition can be any of the join transitions: it does not depend on the definition of the branching automata, but must be chosen consistently with the definition of a path; in particular it may not be unique, and may not exist. Lodaya and Weil defined rational expressions for this class, and studied the algebraic counterpart of branching automata.

Kappé, Brunet, Luttik, Silva and Zanasi [5] introduced another class of automata on finite N-free pomsets, named *pomset automata*. Their approach is an extension of Kleene automata by an additional kind of transitions that split a path into several paths that run in parallel, and define also the destination state to reach when all those parallel paths terminate. Whereas in the definition of branching automata the fork transition that starts parallel paths and the join transition that ends are not linked, both the start and the end are defined by the same transition in pomset automata. In the general case, languages of pomset automata are those of context-free grammars. Assuming a restriction on the definition of pomset automata, their languages are precisely the series-parallel rational languages, which are defined similarly to the usual rational languages of finite words with additional parallel product and a parallel iteration.

In this paper we compare branching and pomset automata. We first slightly generalize the original definition of branching automata by allowing the empty pomset in parallel parts of paths, and show that this corresponds to remove a condition on the rational expressions of the Kleene-like theorem of Lodaya and Weil. Under this generalisation, languages of branching automata are exactly the languages of context-free grammars. As a consequence, they are also exactly the languages of pomset automata. We finally characterize the sub-class of branching automata corresponding to series-parallel rational languages. All results are effective.

2 Notation and basic definitions

Let E be a set. We denote by $\mathcal{P}(E)$, $\mathcal{P}^+(E)$ and $\mathcal{M}^{>1}(E)$ respectively the set of subsets of E, the set of non-empty subsets of E and the set of multi-subsets of E with at least two elements. For any integer n, the set $\{1, \dots, n\}$ is denoted $[n]$ and the group of permutations of $[n]$ by S_n. The cardinality of E is denoted by $|E|$. When $c = (x, y)$ is a pair, we denote by $\pi_1(c) = x$ and $\pi_2(c) = y$.

An alphabet is a set A whose elements are named *letters*. Since in this paper all alphabets are finite and non-empty we will omit to mention it. *Pomsets* (*partially ordered multi-sets*) are a generalization of words [2, 3, 13]. A *labeled poset* $(P, <_P, \rho_P)$ over an alphabet A consists of a set P, a partial ordering $<_P$ of the elements of P and a labeling map $\rho_P \colon P \to A$. For simplicity we often denote $(P, <_P, \rho_P)$ by P. Two labeled posets $(P, <_P, \rho_P)$ and $(Q, <_Q, \rho_Q)$ are *isomorphic* if there is a bijection from P to Q that preserves and reflects both labeling and ordering. A *pomset* P over A is (a representative of) an isomorphism class of labeled posets over A. The *width* of P is the maximal size of an antichain of P. Observe that the finite pomsets of width 1 correspond precisely to the usual finite words: finite totally ordered sequences of letters. The unique empty pomset is denoted by ϵ, and the unique pomset consisting of only one element labeled by $a \in A$ is simply denoted a. Since in this paper all labeled posets and pomsets are finite we omit to say it by now.

Let $(P, <_P, \rho_P)$ and $(Q, <_Q, \rho_Q)$ be two disjoint pomsets over respectively A and A'. The *parallel product* of P and Q, denoted $P \parallel Q$, is the pomset $(P \cup Q, <_P \cup <_Q, \rho_P \cup \rho_Q)$ over $A \cup A'$. The *sequential product* of P and Q, denoted by $P \cdot Q$ or PQ for simplicity, is the pomset $(P \cup Q, <_P \cup <_Q \cup P \times Q, \rho_P \cup \rho_Q)$ over $A \cup A'$. Observe that the parallel product is an associative and commutative operation on pomsets, whereas the sequential product does not commute (but is associative). The parallel and sequential products can be generalized to finite sequences of pomsets. Let $(P_i)_{i\leq n}$ be a finite sequence of pomsets. We denote by $\prod_{i\leq n} P_i = P_0 \cdot \cdots \cdot P_n$ and $\|_{i\leq n} P_i = P_0 \parallel \cdots \parallel P_n$.

The class of *series-parallel* pomsets over A, denoted $SP(A)$, is defined as the smallest class containing ϵ and a for all $a \in A$, and closed under finite parallel and finite sequential product. It is well known that this class corresponds precisely to the class of N-free pomsets [11, 12] over A, in which the exact ordering relation between any four elements x_1, x_2, x_3, x_4 cannot be $x_1 < x_2$, $x_3 < x_2$ and $x_3 < x_4$. We write $SP^+(A)$ for $SP(A) - \{\epsilon\}$. Note that for every pomset P of $SP(A)$ exactly one of the following is true: (i) $P = \epsilon$, (ii) $P = a \in A$, (iii) $P = RS$ or (iv) $P = R \parallel S$ for some non-empty pomsets R, S.

A *language* of $SP(A)$ is a sub-class of $SP(A)$. Sequential and parallel products are extended from pomsets to languages of pomsets in the usual way: when L and L' are languages of pomsets and *op* is either the sequential or the parallel product, then $L\ op\ L' = \{P\ op\ P' : P \in L, P' \in L'\}$.

Let A and B be two alphabets, $P \in SP(A)$, $L \subseteq SP(B)$ and $\xi \in A$. The language of $SP(A \setminus \{\xi\} \cup B)$ consisting of the pomset P in which each element labeled by the letter ξ is non-uniformly replaced by a pomset of L is denoted by $L \circ_\xi P$. By *non-uniformly* we mean that the elements labeled by ξ may be replaced by different elements of L. This substitution $L \circ_\xi$ is the homomorphism from $(SP(A), \|, \prod)$ into the power-set algebra $(\mathcal{P}(SP(A \cup B)), \|, \prod)$ with $\xi \mapsto L$ and $a \mapsto a$ for all $a \in A \setminus \{\xi\}$. Formally:

$$\begin{aligned}
L \circ_\xi \epsilon &= \{\epsilon\} \\
L \circ_\xi a &= \begin{cases} \{a\} & \text{if } a \in A \setminus \{\xi\} \\ L & \text{if } a = \xi \end{cases} \\
L \circ_\xi (P_1 \cdot P_2) &= (L \circ_\xi P_1) \cdot (L \circ_\xi P_2) \\
L \circ_\xi (P_1 \parallel P_2) &= (L \circ_\xi P_1) \parallel (L \circ_\xi P_2)
\end{aligned}$$

This operation can again be extended from pomsets to languages of pomsets by $L' \circ_\xi L = \cup_{P \in L} L' \circ_\xi P$.

▶ **Example 1.** Let $B = \{a, b\}$, $A = B \cup \{\xi\}$, $P = b \parallel (\xi \cdot \xi) \in SP(A)$ and $L = \{a \parallel b, b \cdot a\} \subseteq SP(B)$. Then $L \circ_\xi P = \{b \parallel ((a \parallel b) \cdot (a \parallel b)), b \parallel ((b \cdot a) \cdot (b \cdot a)), b \parallel ((a \parallel b) \cdot (b \cdot a)), b \parallel ((b \cdot a) \cdot (a \parallel b))\}$.

We also set

$$L^{*\xi} = \bigcup_{i\in\mathbb{N}} L^{i\xi} \text{ with } L^{0\xi} = \{\xi\} \text{ and } L^{(i+1)\xi} = (\bigcup_{j\leq i} L^{j\xi}) \circ_\xi L$$

$$L^* = \{\prod_{i<n} P_i : n \in \mathbb{N}, P_i \in L \text{ for each } i < n\} \qquad L^+ = L^* \setminus \{\epsilon\}$$

Assuming ξ is not used in L, we use the following abbreviation:

$$L^{\circledast} = \{\epsilon\} \circ_\xi (L \parallel \xi)^{*\xi} = \{\|_{i<n} P_i : n \in \mathbb{N}, P_i \in L \text{ for each } i < n\} \tag{1}$$

and $L^{\oplus} = L^{\circledast} \setminus \{\epsilon\}$. L^* and L^+ are the sequential iterations of L whereas $L^{\circledast}$ and $L^{\oplus}$ are its parallel iterations.

3 Branching Automata

Branching automata are a generalization of usual Kleene automata. They were introduced by Lodaya and Weil [7, 8, 9, 10]. A *branching automaton* is a tuple $\mathcal{A} = (Q, A, E, I, F)$ where Q is a finite set of states, A is an alphabet, $I \subseteq Q$ is the set of *initial states*, $F \subseteq Q$ the set of *final states*, and E is the finite set of *transitions* of $\mathcal{A}$. The set of transitions of E is partitioned into $E = (E_{\text{seq}}, E_{\text{fork}}, E_{\text{join}})$:

- $E_{\text{seq}} \subseteq Q \times A \times Q$ contains the *sequential* transitions, which are usual transitions of Kleene automata;
- $E_{\text{fork}} \subseteq Q \times \mathcal{M}^{>1}(Q)$ and $E_{\text{join}} \subseteq \mathcal{M}^{>1}(Q) \times Q$ are respectively the sets of *fork* and *join* transitions.

Sequential transitions $(p, a, q) \in Q \times A \times Q$ are sometimes denoted by $p \xrightarrow{a} q$. The *arity* of a fork (resp. join) transition $(p, R) \in Q \times \mathcal{M}^{>1}(Q)$ (resp. $(R, q) \in \mathcal{M}^{>1}(Q) \times Q$) is $|R|$. Here the *source* of a sequential or fork transition is p and the *destination* of a sequential or join transition is q.

We now turn to the definition of paths in $\mathcal{A}$. We give two definitions, namely *b-paths* and *b*-paths*, which are not equivalent: paths labeled by ϵ are allowed in the latter but not in the former. As we will see, considering ϵ as a possible label for paths changes the expressive power of branching automata.

3.1 b-regular and b-rational languages

We recall in this section the original definitions and basic results from Lodaya and Weil.

We define the relation $\underset{\mathcal{A}}{\rightarrow} \subseteq Q \times SP^{+}(A) \times Q$ as the smallest relation satisfying:

1. $p \xrightarrow[\mathcal{A}]{a} q$ if and only if $(p, a, q) \in E$;
2. if $p \xrightarrow[\mathcal{A}]{P} q$ and $q \xrightarrow[\mathcal{A}]{Q} r$ then $p \xrightarrow[\mathcal{A}]{PQ} r$;
3. for all integer $n > 1$, if $p_i \xrightarrow[\mathcal{A}]{P_i} q_i$ for all $i \in [n]$, $(p, \{p_1, \dots, p_n\}) \in E_{\text{fork}}$, $(\{q_1, \dots, q_n\}, q) \in E_{\text{join}}$ then $p \xrightarrow[\mathcal{A}]{\|_{i \in [n]} P_i} q$.

If $p \xrightarrow[\mathcal{A}]{P} q$ we say that there is a *b-path* from p to q labeled by P in $\mathcal{A}$.

A *b-path* is an equivalence class of (finite) terms over $X = \{p \xrightarrow[\mathcal{A}]{a} q : (p, a, q) \in E\}$ using (2) and (3) in the definition above as composition rules, in which terms are equivalent up to the associativity of (2) and to the ordering of the multi-sets $\{p_1, \dots, p_n\}$ and $\{q_1, \dots, q_n\}$ in the fork and join transitions of (3). Thus, the signature of terms is $X \cup \{\cdot\} \cup_{n>1} E_{\text{fork,n}} \times E_{\text{join,n}}$, where elements of X are symbols of arity 0, $\cdot$ has arity 2, and the elements of $E_{\text{fork,n}} \times E_{\text{join,n}}$, pairs of a fork and a join transition of same arity n, have arity n. When Rule (3) is used to form a term t from n terms using fork and join transitions f and j we say that t is a *parallel term rooted* by (f, j). When Rule 2 is used to form t from two terms then t is *sequential.* Each term or b-path t naturally evaluates into a unique $p \xrightarrow[\mathcal{A}]{P} q$ (in this case t is from p to q labeled by P in $\mathcal{A}$). Reciprocally, each element of $\underset{\mathcal{A}}{\rightarrow}$ is the evaluation of at least one term (or b-path). We denote by $t' \preceq t$ (resp. $t' \prec t$) when t' is a (resp. strict) sub-term of the term t. A term t *uses* a transition u if u is a sequential transition used to form t or u is a fork or a join transition and there is some $t' \preceq t$ rooted by some (f, j) with either $f = u$ or $j = u$. It uses a state q if q appear in a transition used in t. Let f and j be respectively a fork and a join transition. A term t uses (f, j) *at the upper level* if there is some $t' \preceq t$ rooted by (f,j) and if $t' \preceq t'' \preceq t$ for some t'' rooted by some (f',j') then t'' is t'. It uses (f,j) *at a sequential level* if there are some $t'' \preceq t' \preceq t$ with t'' rooted by (f,j) and t' a sequential term. Observe that two terms of the same b-path p use exactly the same transitions and states, which allows us to say that p *uses* a transition u (or a state r) if there is some term (or equivalently, if all terms) of p that uses u (or r). The same remark applies for pairs of fork and join transitions used at the upper level or at a sequential level, and can be used to naturally qualify a b-path to be *parallel* or *sequential*.

Set $L_{p,q} = \{P \in SP^+(A) : p \xrightarrow[\mathcal{A}]{P} q\}$. The *language* of $\mathcal{A}$ is $L(\mathcal{A}) = \cup_{(i,f)\in I\times F} L_{i,f}$. We call *b-automaton* a branching automaton equipped with the notion of b-path above. A language $L \subseteq SP^+(A)$ is *b-regular* if L is the language of some b-automaton.

The class of *b-rational* languages of $SP^+(A)$ is the smallest containing $\emptyset$, $\{a\}$ for all $a \in A$, and closed under the operations of $S_b = \{\cup, \cdot, ^+, \|, \circ_\xi, ^{*\xi}\}$, provided that

▶ Condition 1. In $L^{*\xi}$ any element labeled by ξ in some $P \in L$ is incomparable with another element of P.

In particular, Condition 1 excludes from the b-rational languages those of the form $(a\xi b)^{*\xi} = \{a^n\xi b^n : n \in \mathbb{N}\}$, for example.

Let S be a set of functions of arity > 0 on languages. A *S-rational expression* e is a well-formed term of signature $\{\emptyset\}\cup A \cup S$ denoting a language $L(e)$. The *b-rational expressions* are the S_b-rational expressions (verifying Condition 1).

▶ **Theorem 2** ([7]). *A language of $SP^+(A)$ is b-regular if and only if it is b-rational.*

Condition 1 is mandatory in the proof of Theorem 2. That ϵ is forbidden in labels for the parallel composition of b-paths (in Item 3 of the definition of $\xrightarrow[\mathcal{A}]{}$ each P_i is different from ϵ) is also mandatory.

3.2 b*-regular and b*-rational languages

In this section we slightly modify the definition of a b-path by allowing ϵ as a label, in particular in parallel parts.

We define the relation $\xrightarrow[\mathcal{A}]{} \subseteq Q \times SP(A) \times Q$ as the smallest relation satisfying:

1. $p \xrightarrow[\mathcal{A}]{\epsilon} p$ for all $p \in Q$;
2. $p \xrightarrow[\mathcal{A}]{a} q$ if and only if $(p,a,q) \in E$;
3. if $p \xrightarrow[\mathcal{A}]{P} q$ and $q \xrightarrow[\mathcal{A}]{Q} r$ then $p \xrightarrow[\mathcal{A}]{PQ} r$;
4. for all integer $n > 1$, if $p_i \xrightarrow[\mathcal{A}]{P_i} q_i$ for all $i \in [n]$, $(p, \{p_1,\dots,p_n\}) \in E_{\text{fork}}$, $(\{q_1,\dots,q_n\},q) \in E_{\text{join}}$ then $p \xrightarrow[\mathcal{A}]{\|_{i\in[n]}P_i} q$.

The notions of b*-path, b*-automaton, b*-regularity, etc. are defined similarly as in Section 3.1 by a replacement of the relation $\xrightarrow[\mathcal{A}]{}$ with the definition above. As in b-automata, there is no ϵ-transition in a b*-automaton. However, $p \xrightarrow[\mathcal{A}]{\epsilon} q$ with $p \neq q$ is possible using Item 4 with $p_i \xrightarrow[\mathcal{A}]{\epsilon} q_i$ and $p_i = q_i$ for all $i \in [n]$. In a b*-automaton, we have $L_{p,q} = \{P \in SP(A) : p \xrightarrow[\mathcal{A}]{P} q\}$ and $L(\mathcal{A}) = \cup_{(i,f)\in I\times F} L_{i,f}$. Note that because of Item 1 in the definition of the relation $\xrightarrow[\mathcal{A}]{}$ above, there are b*-paths of the form $p \xrightarrow[\mathcal{A}]{\epsilon} p$ that do not use any transition. Such b*-paths are named *trivial*. A b*-path t uses a pair (f,j) of a fork and a join transition at a sequential level if there are some $t'' \preceq t' \preceq t$ with t'' rooted by (f,j) and t' of the form $t' = t'_1 \cdot t'_2$ with t'_1, t'_2 *both non-trivial*.

▶ **Example 3.** Let $A = \{a, b, c\}$, $L = \{a^n c b^n : n \geq 0\}$, and $\mathcal{A}$ be the b*-automaton pictured in Figure 1. Then $L(\mathcal{A}) = L$. Note that L is not b-regular, thus the class of b-regular languages is strictly included into the class of b*-regular ones.

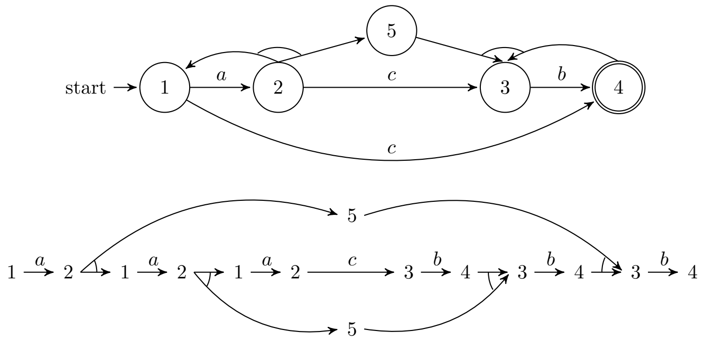


Figure 1 On the top, a b*-automaton $\mathcal{A}$ with $L(\mathcal{A}) = \{a^n c b^n : n \geq 0\}$. The only fork transition is $(2, \{1, 5\})$, the only join transition $(\{4, 5\}, 3)$, the only initial state is 1 and the only final state 4. At the bottom, a representation of a b*-path labeled by *aaacbbb*.

A *(pomset) context-free grammar*, or CFG for short, $G = (T, N, S, R)$ is given by finite sets T of *terminals*, N of *non-terminals*, R of *rules* (or *productions*) and an *axiom* $S \in N$. Rules are of the form $X \to u$ with $X \in N$ and u a finite term built from $N \cup T \cup \{\epsilon\}$ with the sequential and parallel products as operations. The *language* $L(G)$ of G is defined with the axiom S as a start symbol as usual.

▶ **Theorem 4.** *A language of $SP(A)$ is context-free if and only if it is b*-regular.*

Proof. First consider a language $L \subseteq SP(A)$ with $L = L(G)$ for some context-free grammar $G = (T, N, S, R)$. Up to usual transformations, we may assume that there there is at most one production whose right member is ϵ, if there is such a production it is $S \to \epsilon$, and that the axiom S does not appear in any of the right member of the productions in R. For each rule $X \to u \in R$, $X \in N$, build a b*-automaton $\mathcal{A}_{X \to u}$ on the alphabet $T \cup N$ such that $L(\mathcal{A}_{X \to u}) = \{u\}$. For each $X \in N$, build a b*-automaton $\mathcal{A}_X$ from the disjoint union of all $\mathcal{A}_{X \to u}$, $X \to u \in R$. Now build a b*-automaton $\mathcal{A}_G$ such that $L = L(\mathcal{A}_G)$ as follows. For all $X \in N \setminus \{S\}$, take 2 copies $\mathcal{A}_{X,1}$ and $\mathcal{A}_{X,2}$ of $\mathcal{A}_X$. Consider the disjoint union $\mathcal{A}_G$ of $\mathcal{A}_S$ and of all b*-automata $\mathcal{A}_{X,i}$, $x \in N$, $i \in [2]$. For each transition $t = (p, X, q)$, $X \in N$, in $\mathcal{A}_S$ or $\mathcal{A}_{Y,i}$, $Y \neq X$, $i \in [2]$, add a new state t, a fork transition $(p, \{s, t\})$ for each initial state s of $\mathcal{A}_{X,1}$, and a join transition $(\{s', t\}, q)$ for each final state s' of $\mathcal{A}_{X,1}$. Remove the transition t. For each transition $t = (p, X, q)$, $X \in N$, in $\mathcal{A}_{X,i}$, $i \in [2]$, add a new state t, a fork transition $(p, \{s, t\})$ for each initial state s of $\mathcal{A}_{X,j}$, $j \neq i$, and a join transition $(\{s', t\}, q)$ for each final state s' of $\mathcal{A}_{X,j}$. Remove the transition t. The initial and final states of $\mathcal{A}_G$ are taken from $\mathcal{A}_S$. The accepting b*-paths of $\mathcal{A}_G$ are precisely those of $\mathcal{A}_S$ is which each use of a transition (p, X, q), $X \in N$, is replaced by an accepting path of $\mathcal{A}_X$. Immediately, we get $L(\mathcal{A}_G) = L(G)$.

Let us turn to the other direction. Consider a b*-automaton $\mathcal{A}$, and for each pair (p, q) of its states consider the language $L_{p,q}$ of the labels of b*-paths from p to q. Following a McNaughton-Yamada like construction, we build a finite system S of equalities where each $L_{p,q}$ is expressed as a term depending of the $L_{r,s}$, the letters of the alphabet, union, parallel and sequential composition. We refer to [7, Proof of Theorem 6] for the construction of such S. The system S can be easily transformed into a CFG G with $L(G) = L(\mathcal{A})$. ◀

As a consequence, the class of b*-regular languages of $SP(A)$ is not closed under boolean operations, whereas the class of b-regular languages of $SP^+(A)$ is [1].

The class of *b*-rational* languages of $SP(A)$ is the smallest containing $\emptyset$, $\{a\}$ for all $a \in A$, and closed under $S_{b*} = \{\cup, \cdot, {}^*, \|, \circ_\xi, {}^{*\xi}\}$. This definition is the same as b-rational languages, except that sequential iteration $^+$ has been replaced by * and thus ϵ is taken into consideration, and that the restriction expressed by Condition 1 has been removed.

▶ **Example 5.** The language L of Example 3 is given by the b*-rational expression $L = c \circ_\xi (a\xi b)^{*\xi}$.

Observe that the usual Kleene rational languages of A^* are a particular case of the b*-rational languages of $SP(A)$, in which the operators $\|$, $\circ_\xi$ and $^{*\xi}$ are not allowed. The class of *commutative rational* languages of $A^{\circledast}$ (or *over* A), which is the smallest containing $\emptyset$, $\{a\}$ for all $a \in A$, and closed under $\cup$, $\|$ and $^{\circledast}$, is also a particular case of the b*-rational languages of $SP(A)$ (recall Equalities (1)).

▶ **Theorem 6.** *A language of $SP(A)$ is b*-regular if and only if it is b*-rational.*

Proof. First we build a b*-automaton $\mathcal{A}_e$ from a b*-rational expression e such that $L(\mathcal{A}_e) = L(e)$. Using Theorem 4 it suffices to build a CFG G from e, such that $L(G) = L(e)$. This is done by induction over e. For the cases where e has one of the form $e = \emptyset$, $e = \{\epsilon\}$, $e = \{a\}$ with $a \in A$, $e = e_1 \cup e_2$, $e = e_1 \cdot e_2$, $e = e_1 \| e_2$, $e = f^*$, the CFG is directly obtained using the induction hypothesis and usual techniques, so we focus on $e = e_1 \circ_\xi e_2$ and $e = f^{*\xi}$. First assume $e = e_1 \circ_\xi e_2$ and that by induction hypothesis we have two CFG $G_i = (A, N_i, S_i, R_i)$ with $L(G_i) = L(e_i)$, $i \in [2]$. Build $G = (A, N_1 \cup N_2 \cup \{X_\xi\}, S_2, R_1 \cup R \cup \{X_\xi \to \xi, X_\xi \to S_1\})$ in which $X_\xi \notin N_1 \cup N_2$ is a new non-terminal and R is R_2 is which every occurrence of the terminal ξ has been replaced by X_ξ. Then $L(G) = L(e_1 \circ_\xi e_2)$. Assume now $e = f^{*\xi}$ for some b*-rational expression f and let $G_f = (A, N_f, S_f, R_f)$ be the CFG build from f by induction hypothesis. Let $G = (A, N_f, S_f, R \cup \{S_f \to \xi\})$ where R is R_f is which every occurrence of the terminal ξ has been replaced by S_f. Then $L(G) = L(f^{*\xi})$.

Now let $\mathcal{A}$ be a b*-automaton. The proof that $L(\mathcal{A})$ is b*-rational uses exactly the same arguments as those of the direction from left to right of Theorem 2. ◀

We will need later the following particular form of branching automata, adapted from [7] to our case. A b*-automaton $\mathcal{A}$ is *misbehaved* if it has a fork transition $(p, \{p_1, \ldots, p_n\})$ such that $p_j \xrightarrow[\mathcal{A}]{P} f$ for some $j \in [n]$, P and final state f, or if it has a join transition $(\{p_1, \ldots, p_n\}, p)$ such that $i \xrightarrow[\mathcal{A}]{P} p_j$ for some $j \in [n]$, P and initial state i. If $\mathcal{A}$ is not misbehaved then it is *behaved.*

▶ **Proposition 7.** *For every b*-automaton $\mathcal{A}$ there is a behaved b*-automaton $\mathcal{B}$ such that $L(\mathcal{A}) = L(\mathcal{B})$.*

Proof. For each fork transition $f = (p, \{p_1, \ldots, p_n\})$ we take n copies $(\mathcal{A}_{f,i})_{i \in [n]}$ of $\mathcal{A}$. The b*-automaton $\mathcal{B}$ is the disjoint union of these copies with another copy $\mathcal{A}_0$. Delete all the fork and join transitions from $\mathcal{A}_0$. For each fork transition $f = (p, \{p_1, \ldots, p_n\})$ of $\mathcal{A}$, we add to $\mathcal{B}$ a fork transition $(p, \{p_1, \ldots, p_n\})$ where p is taken in $\mathcal{A}_0$ and for all $i \in [n]$, p_i is taken in $\mathcal{A}_{f,i}$. For each join transition $j = (\{q_1, \ldots, q_n\}, q)$, we add all the possible join transitions simulating j where q is taken in $\mathcal{A}_0$ and all the q_i are taken in the different copies $(\mathcal{A}_{j,i})_{i \in [n]}$. It can be verified that if the initial and final states of $\mathcal{B}$ are those of $\mathcal{A}$ taken in $\mathcal{A}_0$, then $\mathcal{B}$ is behaved and that $L(\mathcal{B}) = L(\mathcal{A})$. ◀

The definitions and results about behaveness also trivially apply to b-automata.

4 Pomset Automata

Pomset automata are also a generalization of usual Kleene automata, introduced by Kappé, Brunet, Luttik, Silva and Zanasi [5]. A *pomset automaton* is a tuple $\mathcal{A} = (Q, A, E, \{i\}, F)$ where Q is a finite set of states, A is an alphabet, i is the *initial state*, $F \subseteq Q$ the set of *final states*, and E forms the *transitions* of $\mathcal{A}$. The transitions E consists in two functions:

- $E_{\text{seq}} : Q \times A \to Q$ is the *sequential* transition function, as for usual transitions in complete deterministic Kleene automata;
- $E_{\text{par}} : Q \times Q \times Q \to Q$ is the *parallel transition function.*

We define the relation $\xrightarrow[\mathcal{A}]{} \subseteq Q \times SP(A) \times Q$ as the smallest relation satisfying:

1. $p \xrightarrow[\mathcal{A}]{\epsilon} p$;
2. $p \xrightarrow[\mathcal{A}]{a} E_{\text{seq}}(p, a)$;
3. if $p \xrightarrow[\mathcal{A}]{P} q$ and $q \xrightarrow[\mathcal{A}]{Q} r$ then $p \xrightarrow[\mathcal{A}]{PQ} r$;
4. if $p \xrightarrow[\mathcal{A}]{P} q \in F$ and $r \xrightarrow[\mathcal{A}]{Q} s \in F$ then $t \xrightarrow[\mathcal{A}]{P\|Q} E_{\text{par}}(t, p, r)$.

When presenting a pomset automaton $\mathcal{A}$, we may define the transition function only partially and implicitely assume the existence of an additional sink state $\bot$ (if $\bot \xrightarrow[\mathcal{A}]{P} q$ for some P then $q = \bot$) and a final state $\top$ such that all transitions from $\top$ go to $\bot$.

If $p \xrightarrow[\mathcal{A}]{P} q$ we say that there is a p-path from p to q labeled by P in $\mathcal{A}$. We call *p-automaton* a pomset automaton equipped with the notion of p-path defined as in Section 3.1 but with the relation $\xrightarrow[\mathcal{A}]{}$ above. The *language* of $\mathcal{A}$ is $L(\mathcal{A}) = \{P \in SP(A) : i \xrightarrow[\mathcal{A}]{P} q$ for some $q \in F\}$. A language $L \subseteq SP(A)$ is *p-regular* if L is the language of some p-automaton.

▶ **Example 8.** A p-automaton with same language as the b*-automaton of Example 3 is pictured in Figure 2. For simplicity we do not consider the states $\bot$ and $\top$.

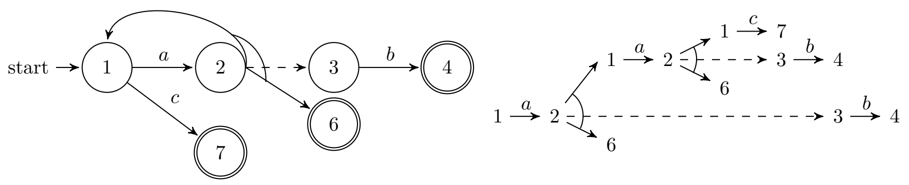


Figure 2 On the left, a p-automaton $\mathcal{A}$ with $L(\mathcal{A}) = \{a^n cb^n : n \geq 0\}$. The parallel transition function is $E_{\text{par}} : (2, 1, 6) \to 3$, the only initial state is 1 and the final states are 4,6,7. On the right, a representation of a p-path labeled by $aacbb$.

It is to notice that the transitions in branching automata are in the definition given by relations, whereas the transitions in pomset automata are functions. However, this does not mean that $p \xrightarrow[\mathcal{A}]{P} r$ and $p \xrightarrow[\mathcal{A}]{P} s$ implies $r = s$ in a p-automaton $\mathcal{A}$: consider for example that $\mathcal{A}$ may have different p-paths starting from a state q and labeled with $P = a \parallel b \parallel c$: one composing p-paths $p_1 \xrightarrow[\mathcal{A}]{a} p_2$ and $p_3 \xrightarrow[\mathcal{A}]{b\|c} p_4$ using a transition $(p, p_1, p_3) \to r$, and another composing some $q_1 \xrightarrow[\mathcal{A}]{a\|b} q_2$ and $q_3 \xrightarrow[\mathcal{A}]{c} q_4$ using a transition $(p, q_1, q_3) \to s$.

▶ **Theorem 9** ([5]). *A language of $SP(A)$ is context-free if and only if it is p-regular.*

As an immediate corollary of Theorems 4 and 9, b*-automata and p-automata have the same expressive power:

▶ **Corollary 10.** *A language of $SP(A)$ is b*-regular if and only if it is p-regular.*

5 Series-parallel rational languages

The class of *series-parallel rational* languages of $SP(A)$ is the smallest containing $\emptyset$, $\{a\}$ for all $a \in A$, and closed under $S_{sp} = \{\cup, \cdot, ^*, \|, ^\circledast\}$. As a consequence of Equalities (1), any series-parallel rational language of $SP(A)$ is also b*-rational, and any series-parallel rational language of $SP^+(A)$ (ϵ not considered) is also b-rational. As noticed in the conclusion of [8], the inclusion is strict, since for example $a \circ_\xi (a \| (a\xi))^{*\xi}$ is b-rational but not series-parallel rational.

In the conclusion of [9] the authors left open the question of a necessary and sufficient condition on a b-automaton $\mathcal{A}$ for $L(\mathcal{A})$ to be series-parallel rational. We answer this question in this section with b*-automata. The result also applies to b-automata, provided that ϵ is not taken into consideration on both automata and rational expressions sides (for example $^\circledast$ and * have to be replaced by respectively $^\oplus$ and $^+$ in the definition of series-parallel rationality above).

A language is *series-parallel regular* if it is the language of some b*-automaton $\mathcal{A}$ verifying Condition 2 below.

▶ Condition 2. There is no b*-path p rooted by some pair (f, j) of a fork and a join transition, such that p uses (f, j) at a sequential level.

Whether a b*-automaton verifies Condition 2 or not is decidable using methods similar to those developed in [10].

▶ **Theorem 11.** *A language L of $SP(A)$ is series-parallel regular if and only if it is series-parallel rational.*

Proof. From right to left we proceed by induction over a series-parallel rational expression e with $L = L(e)$. Since the construction given in the proof of Proposition 7 preserves Condition 2 we may assume that b*-automata constructed at induction steps are behaved. The cases where e has an elementary form, or $e = e_1 \cup e_2$ for some e_1, e_2 are as usual in automata theory. Assume $e = e_1 \cdot e_2$; by induction hypothesis we have behaved b*-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ for respectively e_1 and e_2. Consider the disjoint union $\mathcal{A}$ of $\mathcal{A}_1$ and $\mathcal{A}_2$. For each final state f of $\mathcal{A}_1$, initial state i of $\mathcal{A}_2$ and sequential or fork transition t of source i, duplicate t by replacing the source i with f. The initial states of the resulting b*-automaton are those of $\mathcal{A}_1$. The final states are those of $\mathcal{A}_2$ and in addition the final states of $\mathcal{A}_1$ when $\epsilon \in L(\mathcal{A}_2)$. Assume now $e = e'^*$ and let $\mathcal{A}'$ be a behaved b*-automaton for e'. For each final state f, initial state i, sequential and fork transition t of source i, duplicate t by replacing the source i with f. The initial states are those of $\mathcal{A}'$, the final states are those of $\mathcal{A}'$ plus the initial states. When $e = e_1 \| e_2$ the construction is the disjoint union of $\mathcal{A}_1$ and $\mathcal{A}_2$ with a unique initial new state i, a unique final new state f, for each initial states i_1 and i_2 of respectively $\mathcal{A}_1$ and $\mathcal{A}_2$ a new normal fork transition $(i, \{i_1, i_2\})$, for each final states f_1 and f_2 of respectively $\mathcal{A}_1$ and $\mathcal{A}_2$ a new join transition $(\{f_1, f_2\}, f)$. Assume finally $e = e'^\circledast$. We build from $\mathcal{A}'$ a b*-automaton $\mathcal{A}$ for e as follows. Let T be the set of all sequential or join transitions whose destination is a final state of $\mathcal{A}'$. Let $\mathcal{A}'_0$ and $\mathcal{A}'_t$, $t \in T$, be copies of $\mathcal{A}'$. We build $\mathcal{A}$ from the disjoint union of these copies. Add two new states i and f. For each initial state i' (resp. final state f') of $\mathcal{A}'_0$ add a fork transition $(i, \{i, i'\})$ (resp. a join transition $(\{f', f\}, f)$). For each $t \in T$ duplicate each sequential and fork transition of source i' taken in $\mathcal{A}'_t$ by replacing the source i' with i. Duplicate $t \in T$ in $\mathcal{A}_t$ by replacing the destination f' with f. For each sequential transition (i', a, f') add (i, a, f). The unique initial state of $\mathcal{A}$ is i, and its final states are i and f.

We now prove that the language of some b*-automaton $\mathcal{A} = (Q, A, E, I, F)$ verifying Condition 2 is series-parallel rational. We adapt the McNaughton-Yamada construction of a rational expression from an automaton (see [7, Section 4.2] for the case of b-automata). When $p, q \in Q$, $D, D' \subseteq E_{\text{fork}} \times E_{\text{join}}$, $(f, j) \in E_{\text{fork}} \times E_{\text{join}}$, denote by:

- $L_{p,q} = \{P \in SP(A) : p \xrightarrow[\mathcal{A}]{P} q\}$;
- $L_{p,q}^{D,D'}$ is the set of labels of b*-paths from p to q that can use only pairs of fork and join transitions from D at the upper level and from D' at a sequential level;
- $L(D', f, j)$ is the set of label of b*-paths rooted by (f, j) that can use only pairs of fork and join transitions from D' at a sequential level.

A rational expression for $L_{p,q}^{\emptyset,D'}$ is found as for automata on words since no fork and join transitions are allowed. Otherwise, $D \neq \emptyset$ and since $\mathcal{A}$ verifies Condition 2:

$$\begin{aligned}
L_{p,q} &= L_{p,q}^{E_{\text{fork}} \times E_{\text{join}}, E_{\text{fork}} \times E_{\text{join}}} \\
L_{p,q}^{D,D'} &= \bigcup_{(f,j) \in D \cap D'} \left(L_{p,q}^{D \setminus \{(f,j)\},D'} \cup L_{p,\pi_1(f)}^{D \setminus \{(f,j)\},D'} (L(D', f, j) L_{\pi_2(j),\pi_1(f)}^{D \setminus \{(f,j)\},D'})^* L_{\pi_1(f),q}^{D \setminus \{(f,j)\},D'}\right) \\
&\quad \bigcup_{\substack{(f,j) \in D \\ \pi_1(f) = p \\ \pi_2(j) = q}} \left(L_{p,q}^{D \setminus \{(f,j)\},D'} \cup L(D', f, j)\right) \\
L(D', f, j) &= \bigcup_{\sigma \in S_k} \mathop{\Big\|}_{i \in [k]} L_{r_i, s_{\sigma_i}}^{E_{\text{fork}} \times E_{\text{join}}, D' \setminus \{(f,j)\}}
\end{aligned}$$

where f and j have the form $f = (r, \{r_1, \ldots, r_k\})$ and $j = (\{s_1, \ldots, s_k\}, s)$ in the last equality. The above equalities form a system of equations where the unknowns are the $L_{p,q}^{D,D'}$, $D \neq \emptyset$, and the $L_{p,q}^{\emptyset,D'}$ are the constants. First observe that when $D \neq \emptyset$, $L_{p,q}^{D,\emptyset}$ depends only of the $L_{p',q'}^{D',\emptyset}$, $D' \subset D$ or $D' = E_{\text{fork}} \times E_{\text{join}}$. The only operations involved in the equalities for the $L_{p,q}^{D,\emptyset}$ are $\cup$ and $\|$. The system of equations is solved as usual using substitutions and $^{\circledast}$ is used to resolve circular substitutions. ◀

A similar result holds for p-automata [5] $\mathcal{A}$. In $\mathcal{A}$, define $\preceq$ as the smallest preorder on states such that $E_{\text{seq}}(q, a) \preceq q$, $E_{\text{par}}(q, r, s) \preceq q$, and if $E_{\text{par}}(q, r, s) \neq \bot$ then $r, s \preceq q$. Set also $p \prec q$ if and only if $p \preceq q$ and $q \not\preceq p$. Say that $\mathcal{A}$ is *well-nested* if each state q verifies *exactly* one of the following properties:

1. $r, s \prec q$ for all states r, s with $E_{\text{par}}(q, r, s) \neq \bot$;
2. $q \in F$, $E_{\text{seq}}(q, a) = \bot$ for all a, and if $E_{\text{par}}(q, r, s) \neq \bot$ then $E_{\text{par}}(q, r, s) = \top$, $s = q$ and $r \prec q$.

▶ **Theorem 12** ([5]). *A language L of $SP(A)$ is series-parallel rational if and only if there is a well-nested p-automaton $\mathcal{A}$ with $L = L(\mathcal{A})$.*

We now show that a very similar characterisation also exists for b*-automata to have them correspond to series-parallel rational languages. For every b*-automaton there is a b*-automaton with the same language and with all its fork and join transitions of arity 2. We assume here that all fork and join transitions of b*-automata are of arity 2. In a b*-automaton $\mathcal{A}$, define $\preceq_f$ as the smallest preorder on states such that

1. if $p \xrightarrow[\mathcal{A}]{} q$ then $q \preceq_f p$;
2. if $(p, \{p_1, p_2\}) \in E_{\text{fork}}$ then $p_1, p_2 \preceq_f p$;
3. if $(\{q_1, q_2\}, q) \in E_{\text{join}}$ then $q_1, q_2 \preceq_f q$.

Set $p \prec_f q$ if and only if $p \preceq_f q$ and $q \not\preceq_f p$. Define $\preceq_j$ and $\prec_j$ similarly, by replacing Item 1 above by: if $p \underset{\mathcal{A}}{\rightarrow} q$ then $p \preceq_f q$. Observe that if there is a b*-path $p \underset{\mathcal{A}}{\rightarrow} q$ then $r \preceq_f p$ and $r \preceq_j q$ for all states r used in the b*-path.

A fork transition $(p, \{p_1, p_2\})$ (resp. join transition $(\{p_1, p_2\}, p)$) is *normal* if $p_1, p_2 \neq p$. It is *recursive* otherwise. A state p is *normal* when all the fork transitions $(p, \{p_1, p_2\})$ and join transitions $(\{p_1, p_2\}, p)$ verify $p_1, p_2 \prec_f p$ and $p_1, p_2 \prec_j p$. It is *fork recursive* when all the conditions below are verified:

- there is some recursive fork transition $(p, \{p, p'\})$, and in this case $p' \prec_f p$ and $p' \prec_j p$ for all such transitions;
- for all normal fork transition $(p, \{p_1, p_2\})$ then $p_1, p_2 \prec_f p$ and $p_1, p_2 \prec_j p$;
- for all normal fork transition $(p, \{p_1, p_2\})$ and join transition $(\{q_1, q_2\}, q)$, when $p_1 \underset{\mathcal{A}}{\rightarrow} q_1$ and $p_2 \underset{\mathcal{A}}{\rightarrow} q_2$ then $q \prec_f p$;
- for all sequential transition (p, a, q) then $q \prec_f p$;
- there is no transition of the form (q, a, p) or $(\{p_1, p_2\}, p)$.

It is *join recursive* when all the conditions below are verified:

- there is some recursive join transition $(\{p, p'\}, p)$, and in this case $p' \prec_f p$ and $p' \prec_j p$ for all such transitions;
- for all normal join transition $(\{p_1, p_2\}, p)$ then $p_1, p_2 \prec_f p$ and $p_1, p_2 \prec_j p$;
- for all normal fork transition $(q, \{q_1, q_2\})$ and join transition $(\{p_1, p_2\}, p)$, when $q_1 \underset{\mathcal{A}}{\rightarrow} p_1$ and $q_2 \underset{\mathcal{A}}{\rightarrow} p_2$ then $q \prec_j p$;
- for all sequential transition (q, a, p) then $q \prec_j p$;
- there is no transition of the form (p, a, q) or $(p, \{p_1, p_2\})$.

Then $\mathcal{A}$ is *well-nested* if each state as a unique classification as normal, fork recursive or join recursive, and, when $x = (p, \{p_1, p_2\})$ and $y = (\{q_1, q_2\}, q)$ are a fork and a join transition and $p_i \underset{\mathcal{A}}{\rightarrow} q_i$, $i \in [2]$, then x, y are both recursive, or both normal.

▶ **Proposition 13.** *For every well-nested b*-automaton $\mathcal{A}$ there is a behaved and well-nested b*-automaton $\mathcal{B}$ such that $L(\mathcal{A}) = L(\mathcal{B})$, and the initial and final states of $\mathcal{B}$ are normal.*

Proof. It suffices to check that the construction of the proof of Proposition 7 preserves well-nestedness, and that in the copy $\mathcal{A}_0$ all states are normal. ◀

▶ **Theorem 14.** *A language L of $SP(A)$ is series-parallel rational if and only if there is a well-nested b*-automaton $\mathcal{A}$ with $L = L(\mathcal{A})$.*

Proof. The implication from left to right is by induction over a series-parallel rational expression e for L. The steps are the same as in the proof of Theorem 11 (it suffices to check that the constructions preserve well-nestedness).

For the implication from right to left we show that a well-nested b*-automaton $\mathcal{A}$ verifies Condition 2, and the conclusion follows by Theorem 11. Assume by contradiction that $\mathcal{A}$ does not verify Condition 2, ie. it has a b*-path p rooted by some pair (f, j) of a pair and a join transition such that p uses (f, j) at a sequential level. Let $f = (r, \{r_1, r_2\})$ and $j = (\{s_1, s_2\}, s)$. Take a term t which is a representative of p: there are some $t'' \preceq t' \preceq t$ with t'' rooted by (f, j) and t' of the form $t' = t'_1 \cdot t'_2$ with t'_1, t'_2 both non-trivial. Consider t as a tree. In this tree, consider the path α from the root node n of t to the root node n' of the sub-tree t''. This path goes through the root node n'' of the sub-tree t', that we may consider the first one along α such that $t' = t'_1 \cdot t'_2$ for some non-trivial t'_1, t'_2. The term t has the form $(f, j)(t_1, t_2)$, with t_i a b*-path $r_i \underset{\mathcal{A}}{\rightarrow} s_i$, $i \in [2]$. Either the root node of t_1 or of t_2 belongs to

α, say wlog. t_1. If $r_1 \prec_f r$ or $s_1 \prec_j s$ we get a contradiction since $x \preceq_f r_1$ and $x \preceq_j s_1$ for all states x used in t_1, and thus for $x = r$. Thus $r_1 = r$ and $s_1 = s$. This reasoning is true for all nodes of α between n and n'. Thus t'_1 and t'_2 are respectively non-trivial b*-paths $r \xrightarrow[\mathcal{A}]{} x$ and $x \xrightarrow[\mathcal{A}]{} s$ for some x, that can not use recursive fork or join transitions at the upper level, since there is no fork recursive state y and state y' such that $y' \xrightarrow[\mathcal{A}]{} y$, and there is no join recursive state y and state y' such that $y \xrightarrow[\mathcal{A}]{} y'$. Thus $x \prec_f r$ and $x \prec_j s$. For all states y used in t'_1 it holds $y \preceq_f r$, and for all states y used in t'_2 we have $y \preceq_j s$. Assume first t'' is a sub-term of t'_2. Then we have $z \preceq_f x$ for all states z appearing in t'_2, in particular $r \preceq_f x$, which is a contradiction. Thus t'' is a sub-term of t'_1. We have $z \preceq_j x$ for all states z appearing in t'_1, in particular $s \preceq_j x$, which is again a contradiction. ◀

▶ **Example 15.** Figure 3 represents a well-nested b*-automaton $\mathcal{A}$ obtained by induction on the series-parallel rational expression $e = ((a \parallel b)^*)^{\circledast}$ following the steps of the proof of Theorem 14. Note that it is misbehaved since for example 1 is initial, $1 \xrightarrow[\mathcal{A}]{a\|b} 8$ and because of the join transition $(\{2', 8\}, 8)$. The only fork recursive state is 1, the only join recursive state is 8, and all other states are normal.

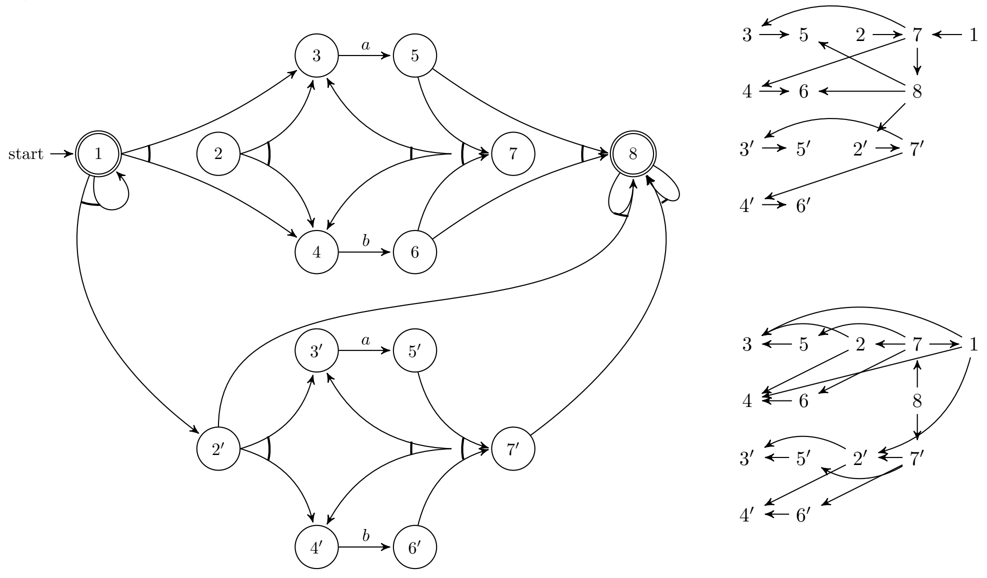
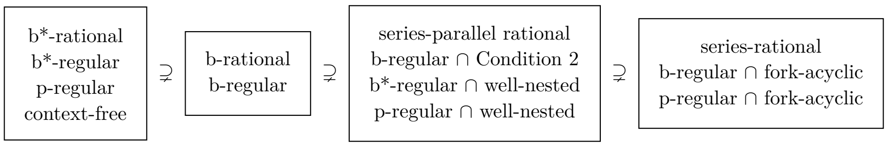


Figure 3 On the left, a well-nested b*-automaton $\mathcal{A}$ with $L(\mathcal{A}) = ((a \parallel b)^*)^{\circledast}$. It has 6 fork transitions $f_1 = (1, \{3, 4\})$, $f_2 = (1, \{1, 2'\})$, $f_3 = (2, \{3, 4\})$, $f_4 = (7, \{3, 4\})$, $f_5 = (2', \{3', 4'\})$, $f_6 = (7', \{3', 4'\})$, 4 join transitions $j_1 = (\{5, 6\}, 8)$, $j_2 = (\{7', 8\}, 8)$, $j_3 = (\{5, 6\}, 7)$, $j_4 = (\{2', 8\}, 8)$, $j_5 = (\{5', 6'\}, 7')$. On the right top, a diagram of the preorder $\preceq_f$ over the states of $\mathcal{A}$. On the right bottom, a diagram of $\preceq_j$.

6 Conclusion

We have compared branching versus p-automata. When they are defined in the most general manner, they have the same expressive power which corresponds to that of context-free grammars, or equivalently, to series-parallel rational expressions with additional $L \circ_\xi L'$ and $L^{*\xi}$ operations that enable respectively substitutions and iterated substitutions. As consequences of the equivalence between b*-automata and context-free grammars questions such as "Is a b*-regular language b-regular, or series-parallel rational?" are undecidable.

We also gave characterizations on branching automata to have them exactly as expressive as series-parallel rational expressions, answering of question of [9]; a similar restriction (*well-nestedness*) was already known for p-automata [5]. All the results are effective. *Series-rational* languages are defined similarly to series-parallel languages without the ability to iterate parallelism (ie. series-rational expressions are series-parallel expressions without $L^{\circledast}$). They have been investigated in [9] for branching automata and in [4] for p-automata. Corresponding automata have the fork-acyclicity property, ie. they can not use a transition that splits an execution flow into several parallel flows $f_1, \ldots, f_n$ into the f_i's again. The following diagram sums-up those results:



References

1 Nicolas Bedon. Logic and branching automata. *Logical Methods in Computer Sciences*, 11(4:2):1–38, October 2015.

2 Jay L. Gischer. The equational theory of pomsets. *Theoret. Comput. Sci.*, 61(2):199–224, 1988. doi:10.1016/0304-3975(88)90124-7.

3 Jan Grabowski. On partial languages. *Fundam. Inform.*, 4(1):427–498, 1981.

4 Tobias Kappé, Paul Brunet, Bas Luttik, Alexandra Silva, and Fabio Zanasi. Brzozowski goes concurrent - A Kleene theorem for pomset languages. In *Proc. CONCUR 2017: 28th International Conference on Concurrency Theory*, volume 28, pages 21:1–21:15, January 2017.

5 Tobias Kappé, Paul Brunet, Bas Luttik, Alexandra Silva, and Fabio Zanasi. On series-parallel pomset languages: Rationality, context-freeness and automata. *Journal of Logical and Algebraic Methods in Programming*, 103, December 2018. doi:10.1016/j.jlamp.2018.12.001.

6 Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Shannon and McCarthy, editors, *Automata studies*, pages 3–41, Princeton, New Jersey, 1956. Princeton University Press.

7 Kamal Lodaya and Pascal Weil. A Kleene iteration for parallelism. In V. Arvind and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1530 of *Lect. Notes in Comput. Sci.*, pages 355–367. Springer-Verlag, 1998.

8 Kamal Lodaya and Pascal Weil. Series-parallel posets: algebra, automata and languages. In M. Morvan, Ch. Meinel, and D. Krob, editors, *STACS'98*, volume 1373 of *Lect. Notes in Comput. Sci.*, pages 555–565. Springer-Verlag, 1998.

9 Kamal Lodaya and Pascal Weil. Series-parallel languages and the bounded-width property. *Theoret. Comput. Sci.*, 237(1–2):347–380, 2000.

10 Kamal Lodaya and Pascal Weil. Rationality in algebras with a series operation. *Inform. Comput.*, 171:269–293, 2001.

11 Jacobo Valdes. Parsing flowcharts and series-parallel graphs. Technical Report STAN-CS-78-682, Computer science departement of the Stanford University, Standford, Ca., 1978.

12 Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler. The recognition of series parallel digraphs. *SIAM J. Comput.*, 11:298–313, 1982. doi:10.1137/0211023.

13 Józef Winkowski. An algebraic approach to concurrence. In *MFCS'79*, volume 74 of *Lect. Notes in Comput. Sci.*, pages 523–532. Springer Verlag, 1979.