A Kleene iteration for parallelism [1]

Kamal Lodaya
Institute of Mathematical Sciences
CIT Campus
Chennai 600 113 – India.
kamal@imsc.ernet.in

Pascal Weil
LaBRI
Université Bordeaux 1 & CNRS
351 cours de la Libération
33405 Talence Cedex – France.
Pascal.Weil@labri.u-bordeaux.fr

Abstract. This paper extends automata-theoretic techniques to unbounded parallel behaviour, as seen for instance in Petri nets. Languages are defined to be sets of (labelled) series-parallel posets — or, equivalently, sets of terms in an algebra with two product operations: sequential and parallel. In an earlier paper, we restricted ourselves to languages of posets having *bounded width* and introduced a notion of *branching* automaton. In this paper, we drop the restriction to bounded width. We define *rational expressions*, a natural generalization of the usual ones over words, and prove a Kleene theorem connecting them to *regular* languages (accepted by finite branching automata). We also show that *recognizable* languages (inverse images by a morphism into a finite algebra) are strictly weaker.

Introduction

The success of automata as a computational model has led to their being generalized in many different ways. Early on, Thatcher and Wright followed an idea of Büchi (as cited in [16]) and constructed automata operating on terms in free algebras, generalizing from strings, which are terms of the free monoid. This led to the development of tree automata and an alternate approach to context-free languages [8].

More recently, Courcelle applied the notion of recognizability to graphs [4]. He showed that for languages of graphs, definability in monadic second-order (MSO) logic implies recognizability. For special classes of graph languages such as context-free graphs, the two notions coincide [5]. Recognizability is defined algebraically and not via automata.

We go back to a very early generalization of automata, that of Petri nets [13]. Although nets are well-researched, the focus has been on examining causality, conflict and concurrency in their behaviour (see the book [15], for example.) There have been a few attempts to study whether results such as the Kleene or Myhill-Nerode theorems extend to Petri nets. Garg and Ragunath [7] obtained a Kleene theorem using *concurrent* rational expressions for all Petri net languages. Grabowski [10] also has a Kleene theorem using languages of (labelled) posets, which provide more information about concurrency. Boudol described "rational parallel place machines" and provided translations into process algebra [2].

[1] Part of this work was done while the second author was visiting the Institute of Mathematical Sciences, in Chennai.

Our idea is to combine the algebraic notion of recognizability with that of regularity. As in a previous paper [11], we work with *sp-languages* (languages of series-parallel posets). Interaction between rationality, recognizability and regularity was investigated for languages of *traces* (see [6] for a recent survey, especially Ochmański's and Zielonka's theorems.) In [11], we obtained a Myhill-Nerode and a Kleene theorem for those languages which have a uniform bound on the "parallel width" of their terms.

Theorem [11] *A bounded width sp-language is series-rational iff recognizable iff regular.*

The *series-rational* languages over the alphabet A are defined by extending the usual rational expressions to include parallel product: every letter $a \in A$ is a series-rational expression; and if e_1 and e_2 are series-rational expressions, then so are $e_1 + e_2$, $e_1 \circ e_2$, $e_1 \| e_2$ and e_1^+. *Recognizability* means the *sp*-language is described by a morphism into a finite algebra. By *regularity*, we mean acceptance by a finite "branching" automaton, which has fork/join transitions in addition to the usual transitions on a letter of the alphabet. This is a restricted form of Petri net.

Bounded width is the key property in proving these theorems. In terms of nets, this translates to the technical notion of 1-safe behaviour.

Differently expressed, the syntax does not allow a "parallel iteration." For instance, the *sp*-language $\{a||bc, a||b(a||bc)c, a||b(a||b(a||bc)c)c, \ldots\}$, which we will denote $(a||bc)\circ_\xi(a||b\xi c)^{*\xi}$, is regular but not bounded width and hence not series-rational. $(a||b\xi c)^{*\xi}$ is an example of a *rational iteration*, while $\circ_\xi$ is a *substitution*. These are borrowed from the "generalized" rational expressions of Thatcher and Wright [16]. In our syntax, the iterated variable is restricted to appear only within a parallel term (see Section 2 for details). A syntax like $\mu\xi.(a||bc + a||b\xi c)$ would be used in process algebra, but that also includes infinite behaviour, which we are not considering.

What happens to recognizability and regularity for unbounded width languages? By using techniques from Büchi [3], we establish one direction of a Myhill-Nerode theorem for the unrestricted languages.

Theorem 1 *Recognizable sp-languages are rational.*

The converse is not true. The *sp*-language $\{a||b, a||a||b||b, \ldots\}$ (equal number of a's and b's), denoted $(a||b)\circ_\xi(a||b||\xi)^{*\xi}$, is rational but not recognizable.

We turn our attention to acceptance by branching automata. The results are more pleasing: we get both directions of a Kleene theorem.

Theorem 2 *An sp-language is rational iff it is regular.*

All three proofs (of Theorem 1 and the two directions of Theorem 2) are nontrivial. Together they verify a conjecture made in [11], that recognizable *sp*-languages are regular. We can also nicely round off our earlier theorem.

Theorem 3 *A bounded width sp-language is series-rational iff recognizable iff regular iff rational.*

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

***sp*-posets.** Let A be an alphabet (a finite non-empty set). An *A-labelled poset* is a poset $(P, \leq)$ equipped with a labelling function $\lambda: P \to A$. 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* [14]), so unless otherwise specified, the underlying sets of distinct posets are always assumed to be pairwise disjoint. All our posets are finite and non-empty.

Let $\mathcal{P} = (P, \leq_P, \lambda_P)$ and $\mathcal{Q} = (Q, \leq_Q, \lambda_Q)$ be A-labelled posets. The *parallel product* $\mathcal{P}||\mathcal{Q}$ of these posets is $(P \cup Q, \leq_P \cup \leq_Q, \lambda_P \cup \lambda_Q)$, and their *sequential product* $\mathcal{P} \circ \mathcal{Q}$ is $(P \cup Q, \leq_P \cup \leq_Q \cup (P \times Q), \lambda_P \cup \lambda_Q)$. It is easily verified that both products are associative, and that the parallel product is commutative.

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.

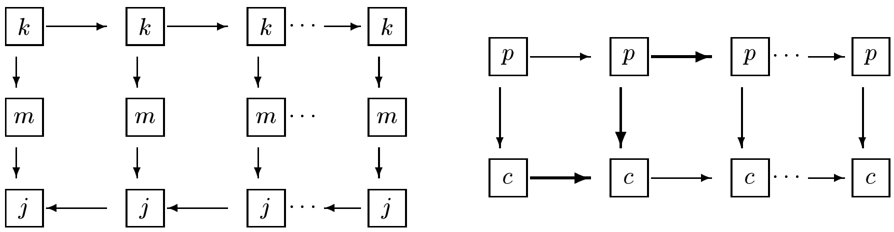


Fig. 1. Posets: series-parallel and not

Example 1.1 Figure 1 shows two posets. Let P_n be the left hand poset having n occurrences of k. Then $P_1 = \{k\} \circ \{m\} \circ \{j\}$ and $P_{n+1} = \{k\} \circ \big(\{m\} || P_n\big) \circ \{j\}$.

The right hand poset 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 using thick lines. A poset is *N-free* if it does not contain N as a subposet. 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 [10, 17].

***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. Note that no distributivity is assumed. The discussion in the previous section shows that the class of (non-empty) *sp*-posets labelled by the alphabet A, forms an *sp*-algebra.

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 (see also [9]).

Proposition [1] *$SP(A)$ is isomorphic to the sp-algebra of non-empty A-labelled sp-posets.*

***sp*-terms.** We mildly abuse notation and call the elements of $SP(A)$ *sp-terms*. (We will never have occasion to refer to elements of the free $(2,2)$-algebra, which would have been the traditional terms.) Thus $x \circ (y \circ z)$ and $(x \circ y) \circ z$ are the *same* *sp*-term, conventionally written xyz.

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 $x = y \| z$ $(y, z \in SP(A))$, and that x is a *parallel term* if it cannot be written as a sequential product $x = y \circ z$ $(y, z \in SP(A))$. The only *sp*-terms which are both parallel and sequential are the letters.

More precisely, 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.

Morphisms and substitution. As is usual with algebras, we define a *morphism* from an *sp*-algebra S to an *sp*-algebra T to be a mapping $\varphi\colon S \to T$ which preserves the two products, that is, such that if $x, y \in S$, then $\varphi(x \circ y) = \varphi(x) \circ \varphi(y)$ and $\varphi(x \| y) = \varphi(x) \| \varphi(y)$.

The endomorphism of $SP(A)$ which maps the letter a to the term t, and leaves every other letter unchanged is *the substitution of a by t*. We write $[t/a]x$ for the substitution of a by t in the term x.

Let $\xi \notin A$. We say that t is a *subterm* of x if there exists a $c \in SP(A \cup \{\xi\})$ with exactly one occurrence of ξ such that $x = [t/\xi]c$. c is called a *context* of t in x. t may have several contexts in x.

Example 1.2 In Example 1.1, $P_{n+1} = k(m \| P_n)j$. So P_n is a subterm of P_{n+1}, in the context $c = k(m \| \xi)j$. In other words, $P_{n+1} = [P_n/\xi]c$. The term k is a subterm of P_n in n different contexts. The term $(m \| (kmj))j$ is also a subterm of P_n but the term $(kmj)j$ is not.

We define the *width* of an *sp*-term as the morphism from $(SP(A), \circ, \|)$ into the "tropical" semiring $(\mathbf{N}, \max, +)$ which maps each letter to 1. Alternatively, it is the maximal cardinality of an anti-chain in the corresponding *sp*-poset.

2 Rational *sp*-languages

An *sp-language* is a subset of some free *sp*-algebra $SP(A)$. One of the results of this paper characterizes the "finite state" *sp*-languages in terms of *rational expressions*, as in the case of languages over words.

Let B be an alphabet containing A. (The elements of $B \setminus A$ will be used as *variables*.) We now define rational expressions over B inductively. The treatment follows Thatcher and Wright [16] (see also [8] and [3]).

(1) Every $b \in B$ is a rational expression;

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

(3) if e_1 and e_2 are rational expressions, and if $\xi \in B$, then $e_1 \circ_\xi e_2$ is a rational expression;

(4) if e is a rational expression, and if $\xi \in B$, then $e^{*\xi}$ is a rational expression, *provided* every occurrence of ξ or ${}^{*\xi}$ within e is inside a parallel subexpression $e_1 || \dots || e_n$.

The last two operations (3) and (4) are called *substitution* and *rational iteration* (or *ξ-exponentiation*), respectively. The proviso in (4) is a departure from [16], but a key element in preserving regularity (Section 4).

Now, we associate with each rational expression an *sp*-language in $SP(B)$:

(1) $L(b) = \{b\}$ for $b \in B$;

(2) $L(e_1 + e_2) = L(e_1) \cup L(e_2)$, $L(e_1 \circ e_2) = L(e_1) \circ L(e_2)$, $L(e_1 \| e_2) = L(e_1) \| L(e_2)$ and $L(e_1^+) = L(e_1)^+$ (Kleene iteration);

(3) $L(e_1 \circ_\xi e_2) = [L(e_1)/\xi]L(e_2)$, that is, the set of *sp*-terms obtained from the elements of $L(e_2)$ by replacing each occurrence of ξ by an element of $L(e_1)$ *non-uniformly* (this is an extension of the substitution notation from the previous section);

(4) $L(e^{*\xi}) = L(e)^{*\xi}$, where for any *sp*-language U, we let $U^{*\xi} = \bigcup_{i \geq 0} U_i^\xi$, with $U_0^\xi = \{\xi\}$ and $U_{i+1}^\xi = (U_0^\xi \cup \cdots \cup U_i^\xi) \circ_\xi U$. This amounts to taking the least *sp*-language L containing ξ such that $L \supseteq [U/\xi]L$.

Finally, an *sp*-language over A is said to be *rational* if it is of the form $L(e)$ for some rational expression e over an alphabet B containing A. For instance, the *sp*-posets of Figure 1 form the rational language $(kmj) \circ_\xi (k(m || \xi)j)^{*\xi}$.

Substitution and rational iteration play a role similar to concatenation and Kleene star. The operation $\circ_\xi$ is associative, and its iteration yields ${}^{*\xi}$. One can define the *parallel iteration* $L^\oplus$ of L as $L^\oplus = L \circ_\xi (\xi \| L)^{*\xi}$ for $\xi \notin A$. Clearly parallel iteration is a form of rational iteration, but the converse is not true: $SP(A) = A^+ \circ_\xi ((A^*(\xi || \xi))^* A^*)^{*\xi}$ cannot be defined using parallel iteration.

3 Recognizable *sp*-languages

An *sp*-language $L \subseteq SP(A)$ is *recognized* by an *sp*-algebra S if there is a morphism $\varphi : SP(A) \to S$ such that $L = \varphi^{-1}\varphi(L)$. An *sp*-language L is *recognizable* if it is recognized by a *finite* *sp*-algebra.

Example 3.1 Let $a \in A$. Then $a^\oplus$ is recognized by the 2-element *sp*-algebra $\{x, 0\}$ given by $x \| x = x$ and $x \circ x = 0$, mapping letter a to x and every other letter in A to 0. $a^\oplus$ is recognizable but not bounded width.

The rational expressions we have defined serve to describe recognizable languages, which is the content of our next theorem. The proof is a nontrivial extension of the McNaughton-Yamada construction. We follow Büchi's ideas [3, Section 6.6, Exercise 4].

Theorem 1 *Recognizable sp-languages are rational.*

Proof. Let S be a finite *sp*-algebra recognizing L by the morphism $\varphi : SP(A) \to S$, with $L = \bigcup_{f \in \varphi(L)} \varphi^{-1}(f)$. The rational expressions we will construct will be over the alphabet $B = A \cup \{\xi_q \mid q \in S\}$, with extra variables for each element of the algebra. The morphism φ is extended to $SP(B)$ by setting $\varphi(\xi_q) = q$.

Let $L(Z, R, s)$, for Z a set of extra variables, $R \subseteq S$, $s \in S$, be the set of terms t in $SP(A \cup Z)$ such that $\varphi(t) = s$, and for every parallel subterm $t_1||t_2$ of t, $\varphi(t_1||t_2)$ is in R.

Clearly, $\varphi^{-1}(f) = L(\emptyset, S, f)$, so it suffices to construct rational expressions for $L(Z, R, s)$, which is done by induction on $|R|$.

For the base case, when $|R| = 0$, all terms in $L(Z, R, s)$ are sequential and we can use the usual McNaughton-Yamada construction to get a rational expression.

For the induction step, we have to define $L(Z, R \cup \{r\}, s)$, with $r \in S \setminus R$ in terms of expressions involving languages of the form $L(Y, R, q)$. The proof is a bit heavy on notation. We introduce some abbreviations.

We use $\widehat{Z}$ for $Z \cup \{\xi_r\}$ and $\widetilde{R}$ for $R \cup \{r\}$. This notation is extended to languages: $\widehat{L}_q$ stands for $L(\widehat{Z}, R, q)$, $\widetilde{L}_q$ for $L(Z, \widetilde{R}, q)$ and $\widetilde{\widehat{L}}_q$ for $L(\widehat{Z}, \widetilde{R}, q)$. We use L_q for $L(Z, R, q)$.

The following equation allows the construction of a rational expression from the rational expressions for the languages on the right hand side, which exist by the induction hypothesis. (This is just an extension of the expression used in the usual McNaughton-Yamada proof.) It suffices to verify the equation to complete the proof of the theorem.

$$\widetilde{L}_s = L_r \circ_{\xi_r} \Big(\bigcup_{p||q=r} \widehat{L}_p || \widehat{L}_q \Big)^{*\xi_r} \circ_{\xi_r} \widehat{L}_s. \qquad (1)$$

Since there is only one variable under consideration, we write $\circ_r$ and ^{*r} instead of $\circ_{\xi_r}$ and $^{*\xi_r}$. We also abbreviate the cumbersome $\bigcup_{p||q=r} \widehat{L}_p || \widehat{L}_q$ by $\widehat{L}_{par}$.

We first show that the right hand side of (1) is included in the left. Let us first look at the language $\widetilde{\widehat{L}}_r = L(\widehat{Z}, \widetilde{R}, r)$. This is the set of terms t using the variable ξ_r such that $\varphi(t) = r$, all whose parallel subterms map to $R \cup \{r\}$. Clearly $\widetilde{\widehat{L}}_r \circ_r \widetilde{\widehat{L}}_r \subseteq \widetilde{\widehat{L}}_r$, and $\xi_r \in \widetilde{\widehat{L}}_r$. Hence $\widetilde{\widehat{L}}_r{}^{*r} = \widetilde{\widehat{L}}_r$.

Now suppose $p||q = r$. Since $r \notin R$, $\widehat{L}_p || \widehat{L}_q \subseteq \widetilde{\widehat{L}}_r$. Since p, q were arbitrary, we get $\widehat{L}_{par} \subseteq \widetilde{\widehat{L}}_r$. Exponentiating both sides, $\widehat{L}_{par}{}^{*r} \subseteq \widetilde{\widehat{L}}_r{}^{*r} = \widetilde{\widehat{L}}_r$.
So $L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r \widehat{L}_s \subseteq L_r \circ_r \widetilde{\widehat{L}}_r \circ_r \widehat{L}_s \subseteq L_r \circ_r \widetilde{\widehat{L}}_r \circ_r \widetilde{\widehat{L}}_s \subseteq L_r \circ_r \widetilde{\widehat{L}}_s \subseteq \widetilde{L}_s$.

Now we turn to the inclusion from left to right. Consider $t \in \widetilde{L}_s$. By induction on the number n of subterms $t_1||t_2$ in t such that $\varphi(t_1||t_2) = r$, we prove $t \in L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r \widehat{L}_s$.

If $n = 0$, then t is actually in $L_r \circ_r \widehat{L}_s$. $\widehat{L}_s$ can be written as $\{\xi_r\} \circ_{\xi_r} \widehat{L}_s$, which is a subset of any term of the form $e^{*r} \circ_r \widehat{L}_s$. Using $\widehat{L}_{par}$ as e, we get the RHS of (1).

For $n > 0$, consider the outermost parallel subterms $t_1||t_2$ in t with $\varphi(t_1) = p$, $\varphi(t_2) = q$ for some p, q and $p||q = r$. Then $t_1 \in \widetilde{L}_p$ and $t_2 \in \widetilde{L}_q$ have at most $n-1$ parallel subterms mapping to r. By the induction hypothesis, $t_1 \in L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r \widehat{L}_p$ and $t_2 \in L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r \widehat{L}_q$.

So $t_1||t_2 \in L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r (\widehat{L}_p||\widehat{L}_q)$.

Hence each such $t_1||t_2$ is in $L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r \widehat{L}_{par} \subseteq L_r \circ_r \widehat{L}_{par}{}^{*r}$.

All these parallel subterms can be substituted for variable ξ_r in a subterm $u \in \widehat{L}_s$ to yield t. Hence $t \in L_r \circ_r \widehat{L}_{par}{}^{*r} \circ_r \widehat{L}_s$. $\square$

As stated in the introduction, the converse to Theorem 1 does not hold.

4 Branching automata

To seek a machine characterization of rational *sp*-languages, we now turn to the branching automata defined in our earlier paper [11].

The tuple $\mathcal{A} = (Q, T_{seq}, T_{fork}, T_{join}, I, F)$ is a *branching automaton* over the alphabet A, 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, $T_{fork} \subseteq Q \times \mathcal{P}_{ns}(Q)$ and $T_{join} \subseteq \mathcal{P}_{ns}(Q) \times Q$ are respectively the sets of fork and join transitions. Here $\mathcal{P}_{ns}(Q)$ ("nonempty and nonsingleton") stands for subsets of 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 $q \to \{q_1, \ldots, q_n\}$. Similarly, an element $(\{q_1, \ldots, q_n\}, q)$ of T_{join}, written $\{q_1, \ldots, q_n\} \to q$, is said to be a *join transition of arity* n.

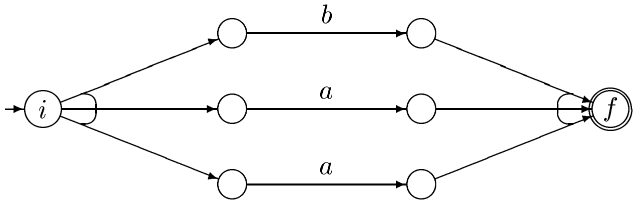
Let p, q be states of a branching automaton $\mathcal{A}$ and let t be a *sp*-term. The existence of a *run of* $\mathcal{A}$ *on* t *from* p *to* q is inductively defined as follows: There is a run on letter a from p to q if $p \xrightarrow{a} q$. If a term t has sequential factorization $t_1 \ldots t_n$ ($n \geq 2$), 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 $t_1|| \ldots ||t_n$ ($n \geq 2$). 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:

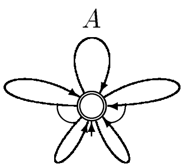
Let $\pi = \{r_1, \ldots, r_s\}$ be a subset of $\{1, \ldots, n\}$. By t_π we mean the parallel term $t_{r_1}|| \ldots ||t_{r_s}$. Now we say there is a run on t from p to q if there exists a partition $\{\pi_1, \ldots, \pi_m\}$ of $\{1, \ldots, n\}$ and there are states p_ℓ, q_ℓ, for $1 \leq \ell \leq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

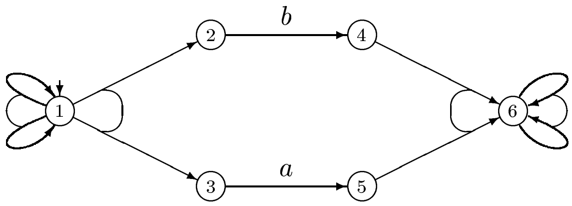
Finally, a branching automaton $\mathcal{A}$ *accepts* the *sp*-term t if there exists a run of $\mathcal{A}$ on t from an initial state to a final state. An *sp*-language L is *regular* if it is the set of *sp*-terms accepted by a finite branching automaton.

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 $\{a||a||b\}$.



Example 4.2 The following automata accept respectively $SP(A)$ and $(a\|b)^{\oplus}$.





Note the looping forks and looping joins in the two automata. It is these that allow unbounded width terms to be accepted. However, these two automata mix up the states "inside" and "outside" the fork, in the sense defined below. This "misbehaviour" creates technical difficulties in proofs and constructions and must be eliminated, as we argue in [12].

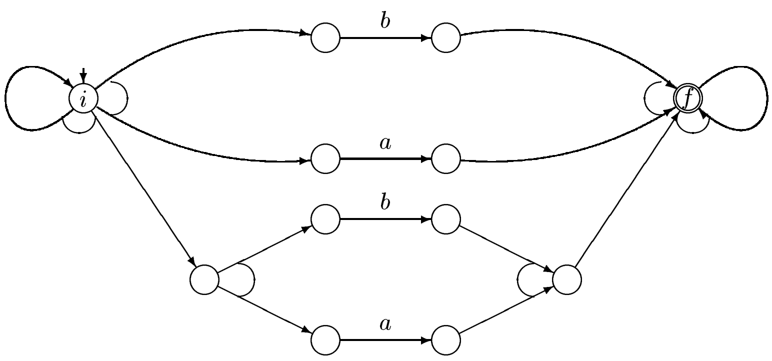
We say that an automaton is *misbehaved* if there exists a fork transition $q \to \{q_1, \ldots, q_n\}$ such that there are runs from each q_i to some final state, or if there is a join transition $\{q_1, \ldots, q_n\} \to q$ such that there are runs from some initial state to each q_i. An automaton is *behaved* if it is not misbehaved.

The two automata in Example 4.2 are misbehaved. In the next proposition, we show that being behaved or misbehaved is a property of the automaton, not of the language. As an example, we give below a behaved automaton for $(a||b)^{\oplus}$.

Proposition 4.3 *Any 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_r.

We construct a new automaton $\mathcal{B}$ as follows. First we take the disjoint union of $|T_{fork}| + 1$ copies of $\mathcal{A}$, denoted respectively $\mathcal{A}_0$ and $\mathcal{A}_k$ $(k \in T_{fork})$. 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 $q, q_1, \ldots, q_{r-1}$ are taken in $\mathcal{A}_0$ and q_r is taken in $\mathcal{A}_k$. 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 $\mathcal{A}_k$, 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 $\mathcal{A}_0$. We leave the reader to verify that $\mathcal{A}$ and $\mathcal{B}$ accept the same *sp*-language. □

4.1 Closure properties of regular *sp*-languages

The main theorem of this section is the following.

Theorem 4 *The class of regular sp-languages is closed under union, intersection, inverse and direct image by a morphism, substitution, sequential product and iteration, parallel product and rational iteration.*

For the operations of union, intersection, and inverse and direct image by a morphism, the proof is by simple constructions adapted from the classical theory of automata. It is the behaved automata which enable us to prove closure under substitution.

Lemma 4.4 *Let K and L be regular sp-languages over the alphabet A and let $a \in A$. Then $[K/a]L$ 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 $[K/a]L$. □

We can easily derive the closure under sequential and parallel product and iteration, since we can come up with automata for $\{\zeta \circ \xi\}$, $\{\zeta \| \xi\}$, ξ^+ and $\xi^\oplus$. These variables can be substituted to prove closure. We need a final lemma to prove closure under rational iteration.

Lemma 4.5 *Let L be a language over $A \cup \{\xi\}$, such that for each term in L, the letter ξ appears, if at all, only inside a parallel subterm. Then $L^{*\xi}$ is regular.*

Proof. We give a sketch of the proof. Let $\mathcal{B}$ be a behaved automaton accepting L and T_ξ its set of ξ-labelled transitions. We construct a new automaton $\mathcal{C}$ from $|T_\xi| + 1$ disjoint copies of $\mathcal{B}$, denoted $\mathcal{B}_0$ and $\mathcal{B}_{p,q}$, for each $p \xrightarrow{\xi} q$ in T_ξ, with the following modifications:

For each sequential and fork transition out of an initial state of $\mathcal{B}$, we add a transition from the state p in each copy to the same destinations in $\mathcal{B}_{p,q}$. For each sequential and join transition into a final state of $\mathcal{B}$, we add a transition from the same sources in $\mathcal{B}_{p,q}$ to q in each copy.

If there is a sequential transition from an initial to a final state of $\mathcal{B}$, we add a similar transition from p to q in each copy. Hence a run of $\mathcal{B}$ can be simulated by a subrun from p to q in any copy of $\mathcal{C}$. Since this also includes the p and q in $\mathcal{B}_{p,q}$, runs for a term obtained by repeated substitution can also be constructed.

The initial and final states of $\mathcal{C}$ are those of $\mathcal{B}$, taken in $\mathcal{B}_0$, along with a new initial and final state with a ξ transition between them. This makes $\mathcal{C}$ accept ξ.

Hence we can argue that any term $t \in L^{*\xi}$ is accepted by $\mathcal{C}$.

For the converse direction, consider a run of $\mathcal{C}$ accepting t. If t does not make use of any new transitions, it is accepted by $\mathcal{B}_0$, and $t \in L = \{\xi\} \circ_\xi L \subseteq L^{*\xi}$.

Otherwise notice that the condition on occurrence of ξ's and the behavedness of $\mathcal{B}$ make sure that the initial and final states of $\mathcal{C}$ cannot be used inside a subrun for a parallel subterm of t. Hence a run ρ of t on $\mathcal{C}$ using a new transition must begin and end in $\mathcal{B}_0$ and go through the copies $\mathcal{B}_{p,q}$. The automaton construction makes sure that the subrun for each substitution of ξ begins and ends in the same copy. This leads to a proof that t is in $L^{*\xi}$. □

This completes the proof of Theorem 4. In particular, the regular languages are closed under the rational operations, so:

Corollary 5 *Rational sp-languages are regular.*

4.2 Extending McNaughton-Yamada further

Theorem 6 *Every regular sp-language is rational.*

Proof. We use a McNaughton-Yamada construction. Let $\mathcal{A}$ be a branching automaton with state set Q. For each pair of states (p,q), let $L_{p,q}$ be the set of all *sp*-terms which label a run of $\mathcal{A}$ from p to q, and let $\xi_{p,q}$ be a new symbol.

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 fork transition f 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. If $D \subseteq T_{\text{fork}}$ is a set of fork transitions of $\mathcal{A}$ and if $p,q \in Q$, we let $L^D_{p,q}$ be the set of labels of runs of $\mathcal{A}$ which use only fork transitions in D at the upper level. Finally, if f is a fork transition in D, we let $L(D,f,q)$ be the set of labels of runs of $\mathcal{A}$ which start with transition f, end at state q and use f only once at the upper level.

First we consider the case where $D = \emptyset$. Then the elements of $L^{\emptyset}_{p,q}$ are (sequential) words, no fork transitions are used, so the usual McNaughton-Yamada algorithm shows that $L^{\emptyset}_{p,q}$ is rational. Next we assume that $D \neq \emptyset$, and we consider a fork transition $f = r \to \{r_1, \dots, r_n\} \in D$. Then we have

$$L(D,f,q) = \bigcup_{g=\{s_1,\dots,s_n\}\to s} \ \bigcup_{\sigma \in S_n} \Big(L_{r_1,s_{\sigma(1)}} \| \cdots \| L_{r_n,s_{\sigma(n)}} \Big) L^{D-\{f\}}_{s,q}$$

where S_n is the group of permutations of $\{1,\dots,n\}$, and

$$L^D_{p,q} \ = \ L^{D-\{f\}}_{p,q} \ \cup \ L^{D-\{f\}}_{p,r} L(D,f,r)^* L(D,f,q).$$

Of course, $L_{p,q} = L^{T_{\text{fork}}}_{p,q}$. Thus, using the above formulæ, for each pair of states $p,q \in Q$, we get a rational expression for $L_{p,q}$, where the languages $L_{r,s}$, $r,s \in Q$, only appear within parallel products. In other words, the *sp*-languages $L_{p,q}$ are the solution of a system of equations

$$\xi_{p,q} = \mathrm{expr}_{p,q}(\xi_{r,s};\ r,s \in Q) \qquad p,q \in Q.$$

Solving is done in standard fashion, using the ${}^{*\xi}$ operation to eliminate each variable. Eventually, we get a simple equation without any argument, that is, a rational expression. Plugging this information back, we iterate these substitutions to find that all the $L_{p,q}$ are rational. □

Corollary 5 and Theorem 6 give us at last our Kleene theorem. We can also quickly dispose of our third theorem by a rather long-winded proof.

Theorem 2 *An sp-language is rational iff it is regular.*

Theorem 3 *A bounded width sp-language is rational iff it is series-rational.*

Proof. The reverse direction is clear. Using Corollary 5, bounded width rational languages are bounded width regular. [11] shows that bounded width regular languages are series-rational. □

Conclusion

In [11], we proved that the three formalisms of algebraic recognizability, rational expressions and automata acceptance coincide in the context of bounded width *sp*-languages. The connection between the latter two formalisms continues to hold in a more liberal setting, where parallelism can be unbounded. But we could not generalize the notion of recognizability to match rational expressions and automata acceptance. This is not surprising. Iteration over a commutative, associative operation is very powerful (recall Parikh's theorem).

We have been unable to define a good analogue of deterministic automata, and to prove (or disprove) that regular *sp*-languages are closed under complement. An obvious lacuna of our approach is the complete absence of communication primitives. That is, our work has more to do with "nets of processes" than with synchronization. Hopefully our theorems are clean enough to provide ideas for extending this approach to a syntax with communication.

References

1. S. Bloom and Z. Ésik. Free shuffle algebras in language varieties, *TCS* **163** (1996) 55–98.
2. G. Boudol. Notes on algebraic calculi of processes, in *Logics and models of concurrent systems* (K.R. Apt, ed.), NATO ASI Series F13 (1985) 261–305.
3. J.R. Büchi. *Finite automata, their algebras and grammars: Towards a theory of formal expressions* (D. Siefkes, ed.), Springer (1989).
4. B. Courcelle. Graph rewriting: an algebraic and logical approach, in *Handbook of Theoretical Computer Science* **B** (J. van Leeuwen, ed.), Elsevier (1990).
5. B. Courcelle. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability, *Theoret. Comp. Sci.* **80** (1991) 153–202.
6. V. Diekert and G. Rozenberg. *The book of traces*, World Scientific (1995).
7. V.K. Garg and M.T. Ragunath. Concurrent regular expressions and their relationship to Petri nets, *Theoret. Comp. Sci.* **96** (1992) 285–304.
8. F. Gécseg and M. Steinby. *Tree automata*, Akadémiai Kiadó, Budapest (1984).
9. J.L. Gischer. The equational theory of pomsets, *TCS* **61** (1988) 199–224.
10. J. Grabowski. On partial languages, *Fund. Inform.* **IV** (1981) 427–498.
11. K. Lodaya and P. Weil. Series-parallel posets: algebra, automata and languages, in *Proc. STACS* (Paris '98), *LNCS* **1373** (M. Morvan, C. Meinel, D. Krob, eds.) (1998) 555–565.
12. K. Lodaya and P. Weil. Series-parallel languages and the bounded width property, *IMSc Tech Rep* **98/07/36** (1998). `http://www.imsc.ernet.in/~kamal/splbwp.ps.gz`.
13. C.A. Petri. Fundamentals of a theory of asynchronous information flow, *Proc. IFIP* (Amsterdam '62), North-Holland (1963) 386-390.
14. V. Pratt. Modelling concurrency with partial orders, *IJPP* **15**(1) (1986) 33–71.
15. W. Reisig. *Petri nets, an introduction*, Springer (1985).
16. 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.
17. J. Valdes, R.E. Tarjan and E.L. Lawler. The recognition of series-parallel digraphs, *SIAM J. Comput.* **11** (1981) 298–313.