Hierarchical structure of 1-safe Petri nets *

Kamal Lodaya[1], D. Ranganayakulu[2], and K. Rangarajan[3]

[1] Institute of Mathematical Sciences, CIT Campus, Chennai 600 113
`http://www.imsc.res.in/~kamal`
[2] SIVET College, Gowrivakkam, Chennai 601 302
`nayakulu@imsc.res.in`
[3] Madras Christian College, Tambaram, Chennai 600 059
`rraguram@md5.vsnl.net.in`

Abstract. The hierarchical process structure of Petri nets can be modelled by languages of series-parallel posets. We show how to extract this structure from a 1-safe Petri net. The technique also applies to represent 1-safe S-systems [11] and communication-free systems [5] in terms of structured programs with `cobegin-coend`. We also define *SR-systems*, a class of 1-safe Petri nets which exactly represents programs of this kind.

Let A be a finite nonempty alphabet. A finite-state automaton over A is an abstract version of flowcharts—think of the letters of the alphabet as representing assignments and tests. Rational (or regular) expressions are an abstract version of the control mechanisms of a "structured" programming language. Kleene's theorem thus ties together flowchart and program construct representations of a sequential programming language. (By using the power of assignments, the Böhm-Jacopini theorem shows that one loop is sufficient.)

1-safe Petri nets [16] can be thought of as representing the flowchart structure of a concurrent program. The state of the program is distributed into a set of places and the transitions can be *forking* (with more than one target place) or *joining* (with more than one source place) or both. These can be used to model the hierarchical structure of processes as well as their interaction with each other. One can seek therefore a Kleene theorem which represents a Petri net as a program in an abstract concurrent programming language.

Earlier work. Several authors have defined expressions for nets. The earliest work we know of is by Grabowski [10] who defined expressions for poset languages of 1-safe net systems. (In this paper, we only deal with posets which are finite and labelled—these are also known as pomsets [17].) Garg and Ragunath [6] defined "concurrent regular expressions" to describe word languages of Petri nets. The Petri Box Calculus (PBC) [1] is a rich formalism for describing concurrent and branching behaviour of Petri nets, following much work in process algebra [9, 8, 20, 3, 15].

* The first author acknowledges partial support from the Indo-French project IFC-PAR/CEFIPRA 2102-1 for presenting a first version of the ideas in this paper in the Workshop on Logic and Algebra in Concurrency, Dresden, 2000.

A disadvantage of Grabowski's "Kleene-like" theorem (as also that of Garg and Ragunath) is that the "expressions" allow renaming and hiding operations. Hiding is renaming to an empty poset, and can be represented as manipulation of a local variable in a programming language. For example, the process algebra community usually models the sequential composition of two programs `E1;E2` by using a shared variable (called `tick` below) and synchronization so that the poset corresponding to a run of `E1` precedes that corresponding the run of `E2`.

```
new shared var tick in
cobegin E1 (* has assignments to tick to signal termination *)
||      E2 (* has tests of tick to signal starting *)
coend
```

This introduces parallelism in program structure where there is none in the semantics. In our view, shared variables are acceptable to represent inter-process communication, but they should not be introduced where no process structure is required. Similarly, iteration (looping) is a natural programming structure and simulating it by prefixing and recursion is not acceptable.

Ochmański's co-rational expressions [14] provide syntax for Mazurkiewicz trace languages [4], which are another popular way of describing poset behaviour of 1-safe Petri nets. Again, Ochmański's co-iteration does not correspond to a natural programming language construct.

Our approach. Whether the full class of 1-safe net behaviours can be represented in terms of programming constructs remains to be seen. In this paper we model the forking and joining transitions which are used only for hierarchical ordering of processes by *series-rational expressions* [13], which can be thought of as `while` programs with a `cobegin-coend` construct. We do not deal with the representation of communication between processes.

More precisely, we show that the series-parallel posets accepted by 1-safe nets form exactly the series-rational languages, and we define *SR-systems*, a class of 1-safe Petri nets such that the posets they accept are exactly these languages.

Other abstract ways of describing "finite-state" languages are by morphisms from the language of terms into a finite algebra, or by using a run as a model over which logical formulas can be evaluated. We use earlier work to show equivalence to this kind of algebraic recognizability and logical definability.

Thus we have a Kleene-Myhill-Nerode-Büchi description. Renaming and especially hiding are not easy operations to model in the algebraic and logical frameworks and this is a technical reason we avoid using them.

One of the nice features about Petri nets is the structural characterizations that are available for some significant subclasses. Prominent among these are S-systems, T-systems and free choice (FC) systems [11].

We characterize two subclasses of nets which fall within the ambit of sp-languages: S-systems and communication-free (CF) systems [5]. We define two subclasses of sp-languages, the right and outermost sp-languages, and show that the rsp- and osp-languages accepted by 1-safe nets can be accepted by 1-safe CF and S-systems respectively. They are also characterized in terms of algebra, logic and syntactic expressions to obtain a Kleene-Myhill-Nerode-Büchi description.

1 Preliminaries: sp-terms and their runs on nets

A (labelled) net is a tuple (P, T, F, ℓ) where P is a set of places, T a (disjoint) set of transitions, $\ell : T \to A$ a function labelling the transitions and $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation.

For a place or transition x, its pre-set $F^{-1}(x)$ is conventionally denoted ${}^\bullet x$ and its post-set $F(x)$ is denoted $x^\bullet$, and their union is called the neighbourhood of x. A net is said to be unbranched if for all its places p, $|{}^\bullet p| \le 1$ and $|p^\bullet| \le 1$.

We will assume that F satisfies the condition that for each transition t, ${}^\bullet t$ and $t^\bullet$ are nonempty, and for each place p, either ${}^\bullet p$ or $p^\bullet$ is nonempty. A net is called acyclic if the relation F^* is antisymmetric.

A (labelled) net system is a tuple $N = (P, T, F, \ell, M_0)$, where $M_0 : P \to \mathbf{N}$ is an initial marking. The run of a net system is described by a "token game" leading from the initial marking to other markings [16]. Formally, a marking is a function $M : P \to \mathbf{N}$.

A net system is said to be 1-safe if for all markings M reachable from the initial marking and for all places p, we have $M(p) \le 1$. Now markings can be interpreted as 0/1-vectors, or equivalently as subsets of places. We will therefore use $M \subseteq P$ instead of $M : P \to \mathbf{N}$ to describe markings of 1-safe net systems.

Our definition of a "run" of a net system is known in net theory as a "non-sequential process". To "accept" finite runs of net systems as in automata theory, we define an *accepting* 1-safe net system to be a tuple $\mathcal{N} = (N, \mathcal{F})$, where N is a 1-safe net system and $\mathcal{F} \subseteq \wp P$ is a set of final markings.

Let $\mathcal{N} = (P, T, F, \ell, M_0, \mathcal{F})$ be an accepting 1-safe net system and $O = (P', T', F', \ell')$ an unbranched, acyclic 1-safe net. O is a (terminal) process of $\mathcal{N}$ if it can be mapped onto the accepting system—formally, we require two functions $\pi : P' \to P$ and $\pi : T' \to T$ (the context will make clear which function is meant) satisfying the following properties:

- for all t in T', $\pi({}^\bullet t) = {}^\bullet\pi(t)$, $\pi(t^\bullet) = \pi(t)^\bullet$ and $\ell(\pi(t)) = \ell'(t)$ (transition neighbourhood and labelling respected);
- $\pi(\{p \in P' \mid {}^\bullet p = \emptyset\}) = M_0$ and $\pi(\{p \in P' \mid p^\bullet = \emptyset\}) \in \mathcal{F}$ (initial and final markings respected).

The derived tuple $(T', (F')^* \cap (T' \times T'), \ell' \lceil T')$ is a poset, and we will say that the net system $\mathcal{N}$ *accepts* this poset. The set of all posets $\mathcal{N}$ accepts is called $PL(\mathcal{N})$, the poset language of $\mathcal{N}$.

1.1 The sp-runs of a net system

We now restrict attention to the subclass of labelled N-free posets, that is, those nonempty posets which satisfy

$$\forall w, x, y, z (w \le y,\ w \le z,\ x \le z,\ w \operatorname{co} x,\ y \operatorname{co} z \text{ implies } x \le y) \qquad (1)$$

Here $a \,\mathrm{co}\, b$ says that a and b are unordered. It has been shown [10, 7] that these posets are described inductively using a set of series-parallel terms $SP(A)$ over the alphabet A, given by the syntax $u ::= a \in A \mid u_1 u_2 \mid u_1 || u_2$, which we call *sp-terms*. (Note that there is no way of describing the empty poset.)

We use this result to inductively describe those runs of a net system which correspond to N-free posets. The $M_1 \, [u\rangle \, M_2$ notation standard in net theory is used. Of these, the ones from the initial marking to a final one form $PL(\mathcal{N}) \cap SP(A)$.

We say $M_1 \, [a\rangle \, M_2$ holds if for some a-labelled transition t such that ${}^\bullet t \subseteq M_1$, $(M_1 \setminus {}^\bullet t) \cup t^\bullet = M_2$.

The term $u||v$ gives the parallel composition (disjoint union) of the posets u and v, leaving their elements unordered with respect to each other. $M_1 \, [u||v\rangle \, M_2$ holds if each M_i can be split into disjoint components M_i^u and M_i^v such that $M_1^u \, [u\rangle \, M_2^u$ and $M_1^v \, [v\rangle \, M_2^v$.

The term uv stands for the poset formed by sequencing the posets u and v, that is, ordering all elements of u before those of v. We say that a marking M is *connected* if it cannot be decomposed into two markings M_1 and M_2 which can independently enable transitions. $M_1 \, [uv\rangle \, M_2$ holds if there is an intermediate connected marking M such that $M_1 \, [u\rangle \, M$ and $M \, [v\rangle \, M_2$.

Without the connectedness condition on intermediate markings, a run on $ab||cd$ could also be seen as a run on $(a||c)(b||d)$, and these two are distinct poset behaviours.

We have implicitly used the fact that sequencing and parallel composition are associative and that parallel composition is also commutative.

We do not know of any structural definition which will restrict nets to exactly those which have N-free behaviour.

1.2 sp-algebras and sp-languages

More generally, an *sp-algebra* is a set S equipped with the binary associative operations of sequential product and parallel composition, such that parallel composition is also commutative. $SP(A)$ is the free sp-algebra over generators A, built from the letters of A by these operations. In this paper, our interest will be restricted to those poset languages which can be seen as *sp-languages*, subsets of $SP(A)$ [13].

We define the width of a sp-term to be the maximum number of nested parallel operations occurring in it plus one. For instance, the term $a(b||c(a||b))$ has width three. Intuitively, width provides an upper bound on the number of processors required for a maximally parallel implementation of the poset corresponding to an sp-term. The width of an sp-language is the supremum of the widths of its terms.

We will be interested in two subclasses of $SP(A)$.

The outermost parallel sp-terms $OSP(A)$ are those where no parallel composition is nested inside a sequential product. $OSP(A)$ characterizes finite posets which are disjoint unions of chains.

The right parallel sp-terms $RSP(A)$ are those where a subterm of the form $(u_1||u_2)u_3$ is disallowed, that is, only the right multiplicand of a sequential product is allowed to be a parallel composition. $RSP(A)$ characterizes finite join-free posets, that is, those posets where there do not exist distinct elements x, y, z such that $x \leq z$, $y \leq z$ and x, y are unordered.

Clearly $A^+ \subset OSP(A) \subset RSP(A) \subset SP(A)$. If an sp-language is a subset of $RSP(A)$, we call it an rsp-language; osp-languages are similarly defined.

A zero element in an sp-algebra absorbs both operations. If it exists, it is unique and is written 0. We say an sp-algebra is *right zero* if it has a zero and for any parallel term $x_1||x_2$, right multiplying it by any x_3 yields zero, that is, $(x_1||x_2)x_3 = 0$. An sp-algebra is *outermost zero* if it has a zero and multiplying any parallel term on the left or the right yields zero.

As usual, a morphism between sp-algebras is a map preserving the two operations. An sp-language L is recognized by an sp-algebra S if there is a morphism $\phi : SP(A) \to S$ such that $L = \phi^{-1}\phi(L)$. If S is finite, we say L is recognizable.

Proposition 1 *Let L be a recognizable sp-language. Then L is an rsp-language (osp-language) if and only if it is recognized by an sp-algebra S such that S is right zero (outermost zero, respectively) and 0 is not in the image of L.*

1.3 Monadic second-order logic

We now turn to a different view of labelled posets, seeing them as models of a logical language $MSO[<, A]$. The atomic formulas of this logic are $\ell(x) = a$ (for $a \in A$), $x < y$ and $x = y$. Apart from the usual first-order variables (such as x, y above) interpreted as elements of the poset under consideration, we work with monadic second-order variables, interpreted over subsets of poset elements. The formulas are closed under boolean operations, first-order and monadic second-order quantification. The definition of when the poset u satisfies the formula α ($u \models \alpha$) is standard.

Any (MSO) formula α defines the language of all the posets which satisfy it, $PL(\alpha) = \{u \mid u \models \alpha\}$. Call such a language (MSO-)definable.

For instance, the description (1) can be written as a (first-order) formula (along with a formula specifying a nonempty poset), defining the poset language $SP(A)$ of all N-free posets. Similarly, the languages $RSP(A)$ and $OSP(A)$ are definable using the formulas

$$\delta_{rsp} = \forall x, y, z(x \leq z \text{ and } y \leq z \text{ implies } x \leq y \text{ or } y \leq x)$$

$$\delta_{osp} = \delta_{rsp} \text{ and } \forall x, y, z(x \leq y \text{ and } x \leq z \text{ implies } y \leq z \text{ or } z \leq y)$$

Hence, by conjuncting a formula α with that for (say) $SP(A)$, we get the definable sp-languages. MSO-definability of sp-languages was studied by Kuske [12], who showed that the recognizable sp-languages coincide with the MSO-definable ones.

Proposition 2 *The recognizable rsp-languages (osp-languages) of bounded width equal the MSO-definable rsp-languages (osp-languages) of bounded width.*

2 Operations corresponding to programming constructs

Our basic syntax will be the *series-rational expressions*, which can be seen as an abstract representation of structured programs with `cobegin-coend` statements. These expressions will be interpreted as sp-languages.

- Every $a \in A$ is a rational expression. If r_1 and r_2 are rational expressions, then so are $r_1 \cup r_2$, $r_1 r_2$, r_1^+.
- Every rational expression is a series-rational expression. If e_1 and e_2 are series-rational expressions, then so are $e_1 \cup e_2$, $e_1 e_2$, e_1^+ and $e_1 || e_2$.

The rational expressions are exactly the same as the (regular) expressions for words, except that they are interpreted over posets. Adding parallel composition (`cobegin-coend`) gives the series-rational expressions.

We again identify two subclasses of interest. An expression with parallel composition only at the outermost level is said to be *outermost series-rational.* This corresponds to one single `cobegin-coend` inside which we have a `while` program for each process.

An expression where parallel composition is only allowed on the right hand side of a sequential product is said to be *right series-rational.* (More precisely, the right series-rational expresssions are defined by the syntax below. The outermost series-rational expressions can be similarly defined.) In terms of programming languages, these are structured programs with `cobegin-coend` such that all the `coend`'s occur at the end of the program. Processes can be "forked" to make the process structure grow, but only the end of the program terminates all the processes.

$$r ::= a \mid r_1 \cup r_2 \mid r_1 r_2 \mid r_1^+, \quad e ::= r \mid r e_1 \mid e_1 || e_2$$

Now, we associate with each expression an sp-language. $L(a)$ is the singleton set consisting of the term a, $L(e_1 \cup e_2) = L(e_1) \cup L(e_2)$, $L(e_1 e_2) = L(e_1)L(e_2)$, $L(e_1^+) = L(e_1)^+$ (Kleene iteration) and $L(e_1 || e_2) = L(e_1) || L(e_2)$. The operations on the right hand side are performed pointwise on sp-terms. L^+ is defined as usual inductively, to give the concatenation of L any number of times.

Finally, an sp-language over A is said to be series-rational if it is of the form $L(e)$ for some series-rational expression e over A. We similarly define the outermost and right series-rational languages. The paper [13] shows that the series-rational languages are the recognizable sp-languages of bounded width.

From the definitions, it follows that the outermost series-rational languages are subsets of $OSP(A)$ and the right series-rational languages are subsets of $RSP(A)$. Conversely, if a series-rational language is a subset of $OSP(A)/RSP(A)$, we can find an outermost/right series-rational expression generating it.

Proposition 3 *The right (outermost) series-rational languages are the recognizable rsp- (osp-)languages of bounded width.*

Putting Proposition 3 and Proposition 2 together gives us algebraic and logical characterizations for our expressions. In the rest of the paper, we tie these up with Petri nets.

3 Net constructions

In this section, we will give constructions on nets which correspond to the expression operations above. Many of these are based on earlier work [9, 8, 20, 3, 15], but ours are the first direct constructions we have seen for net systems with final markings.

We first recall a construction which does not affect the poset language accepted by a net. The place complement of a net adds a copy P^{comp} of the set of places P of $\mathcal{N}$ in such a way that a place contains a token iff the complementary places are empty [19]. The initial marking of the new net is $(P \setminus M_0)^{comp} \cup M_0$ and the final markings are $(P \setminus M_f)^{comp} \cup M_f$ for every $M_f \in \mathcal{F}$.

We need a technical notion: the final width $fwd(u)$ of an sp-term u is the width of its last sequential component.

We call a net system $\mathcal{N} = (P, T, F, \ell, M_0, \mathcal{F})$ *behaved* if it satisfies the following properties:

- ${}^\bullet M_0 = \emptyset$; ${}^\bullet(M_0{}^\bullet) = M_0$ (initial marking a source)
- For every M_f in $\mathcal{F}$, $M_f{}^\bullet = \emptyset$; $({}^\bullet M_f)^\bullet = M_f$ (final marking a sink)
- For every M_f in $\mathcal{F}$ such that $M_0\, [u\rangle\, M_f$, $|M_f| = fwd(u)$ (final markings separated by parallel width)
- Distinct M_f^1, M_f^2 in $\mathcal{F}$ are disjoint from each other and from M_0
- For every $M_1\, [u||v\rangle\, M_2$, $\mathcal{N}$ can be divided into disjoint parts $\mathcal{N}_u, \mathcal{N}_v$ such that $M_1\lceil\mathcal{N}_u\, [u\rangle\, M_2\lceil\mathcal{N}_u$ and $M_1\lceil\mathcal{N}_v\, [v\rangle\, M_2\lceil\mathcal{N}_v$ (parallel sub-runs disjoint).

Now we start constructing nets for expressions. The construction of a behaved net system for the expression a is trivial, we call it an *atomic* net system.

We give constructions for the other operations. Note that we use $+$ to indicate disjoint union.

3.1 Union

Let $\mathcal{N}_i = (P_i, T_i, F_i, M_0^i, \mathcal{F}^i, \ell_i), i = 1, 2$ be 1-safe net systems. Then their sum is the net system $\mathcal{N} = (P, T, F, M_0, \mathcal{F}, \ell)$ where

- $P = P_1 + P_2 + P_{new}$; $P_{new} = (M_0^1 \times M_0^2)$
- $T = T_1 + T_2 + T_{new}$; $T_{new} = (M_0^1)^\bullet \cup (M_0^2)^\bullet$
- $\ell = \ell_1 \cup \ell_2 \cup \{(t \to \ell_i(t)) \mid t \in T_{new} \cap T_i, i = 1, 2\}$
- $F = F_1 \cup F_2 \cup \{((p_1, p_2), t) \in P_{new} \times T_{new} \mid (p_1, t) \in F_1 \text{ or } (p_2, t) \in F_2\}$ $\cup \{(t, p) \in T_{new} \times (P_1 \cup P_2) \mid (t, p) \in F_1 \text{ or } (t, p) \in F_2\}$
- $M_0 = P_{new}$; $\mathcal{F} = \mathcal{F}^1 \cup \mathcal{F}^2$

The new system adds to a copy of both $\mathcal{N}_i$ a fresh set of places for the marking $M_0^1 \times M_0^2$ (which is the new initial marking, and a final marking if either M_0^i was a final marking) and a fresh copy of transitions enabled by each M_0^i. Figure 1 shows an example.

From the construction, we have for $\mathcal{N}$ that its initial marking is a source. If both the $\mathcal{N}_i$ are behaved, then the final markings of $\mathcal{N}$ are sinks, and are disjoint from each other. The parallel final width and disjoint sub-run conditions also follow from the behavedness of the $\mathcal{N}_i$.

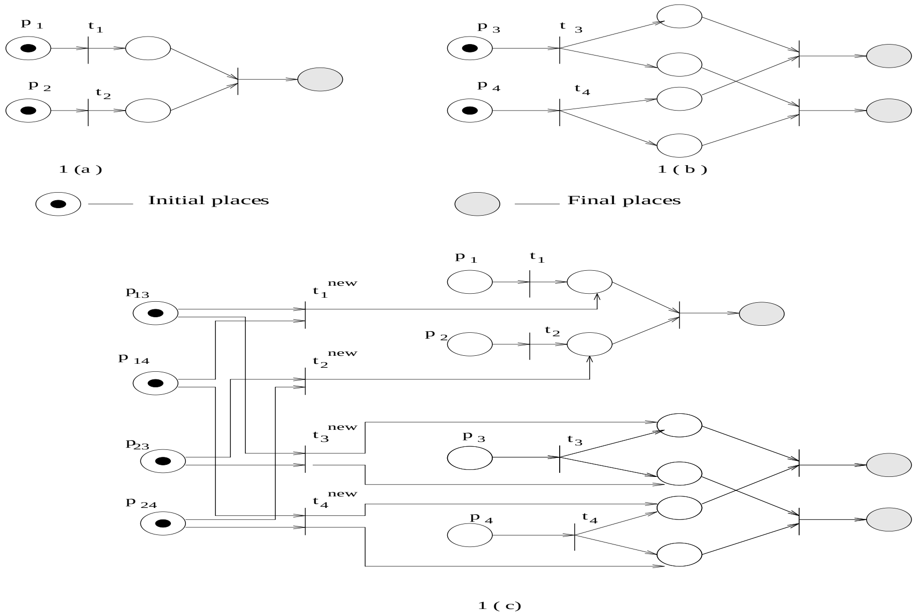

Fig. 1. $PL(1c) = PL(1a) \cup PL(1b)$

Proposition 4 $PL(\mathcal{N}) = PL(\mathcal{N}_1) \cup PL(\mathcal{N}_2)$.

3.2 Sequencing

Let $\mathcal{N}_i = (P_i, T_i, F_i, M_0^i, \mathcal{F}^i, \ell_i), i = 1, 2$ be 1-safe net systems. Then their sequential composition is the net system $\mathcal{N} = (P, T, F, M_0, \mathcal{F}, \ell)$ where

- $P = P_1 + P_2 + P_{new};\ P_{new} = \bigcup_{M_f^1 \in \mathcal{F}^1} (M_f^1 \times M_0^2)$
- $T = T_1 + T_2 + T_{new};\ T_{new} = \bigcup_{M_f^1 \in \mathcal{F}^1} ({}^\bullet(M_f^1) \cup (M_0^2)^\bullet)$
- $\ell = \ell_1 \cup \ell_2 \cup \{(t \to \ell_i(t)) \mid t \in T_{new} \cap T_i, i = 1, 2\}$
- $F = F_1 \cup F_2 \cup \{(p, t) \in P_1 \times T_{new} \mid (p, t) \in F_1\}$
  $\cup \{(t, (p_1, p_2)) \in T_{new} \times P_{new} \mid (t, p_1) \in F_1\}$
  $\cup \{((p_1, p_2), t) \in P_{new} \times T_{new} \mid (p_2, t) \in F_2\}$
  $\cup \{(t, p) \in T_{new} \times P_2 \mid (t, p) \in F_2\}$
- $M_0 = M_0^1;\ \mathcal{F} = \mathcal{F}^2$

The new system uses a product of $M_f^1 \times M_0^2$ for each $M_f^1 \in \mathcal{F}^1$, making copies of transitions coming in to M_f^1 as well as leaving M_0^2. M_0^1 is the initial marking, and $\mathcal{F}^2$ the final markings. This is illustrated in Figure 2.

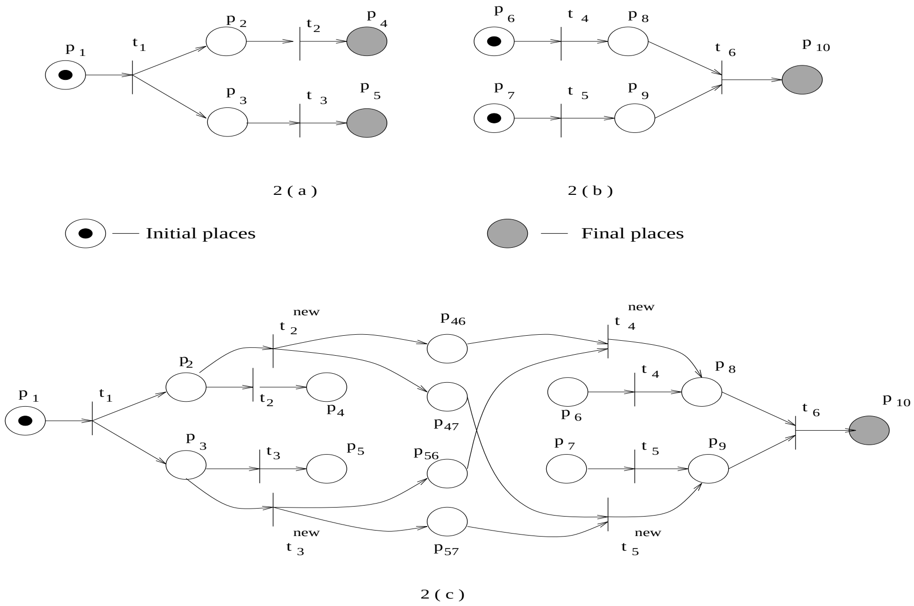

Fig. 2. $PL(2c) = PL(2a)PL(2b)$

If both nets $\mathcal{N}_i$ are behaved, since the construction does not affect the initial marking of $\mathcal{N}_1$ or the final markings of $\mathcal{N}_2$, the source, sink and final width properties are maintained. The parallel sub-run property is also maintained by the argument presented next.

Proposition 5 *If $\mathcal{N}_1$ is place complemented and both $\mathcal{N}_i$ are behaved, $PL(\mathcal{N}) = PL(\mathcal{N}_1)PL(\mathcal{N}_2)$.*

3.3 Iteration

We follow the same idea for iteration as for sequencing. Given a 1-safe net system $\mathcal{N}_1$, take *two* copies of it, call them $\mathcal{N}_1$ and $\mathcal{N}_2$ and perform the "product" construction above, this time linking $\mathcal{F}^1$ to M_0^2 *as well as* $\mathcal{F}^2$ to M_0^1. Let M_0^1 be the initial marking, and the set of final markings is $\mathcal{F}^1 \cup \mathcal{F}^2$. Figure 3 shows an example. Here is a detailed construction.

Let $\mathcal{N}_i = (P_i, T_i, \ell_i, F_i, M_0^i, \mathcal{F}^i)$ be two copies of an accepting 1-safe net system $\mathcal{N}_1$. Then the net for its iteration is $\mathcal{N} = (P, T, \ell, F, M_0, \mathcal{F})$ where

– $P = P_1 + P_2 + P_{new}^{12} + P_{new}^{21}$;
$P_{new}^{12} = \bigcup_{M_f^1 \in \mathcal{F}^1} (M_f^1 \times M_0^2)$; $P_{new}^{21} = \bigcup_{M_f^2 \in \mathcal{F}^2} (M_f^2 \times M_0^1)$

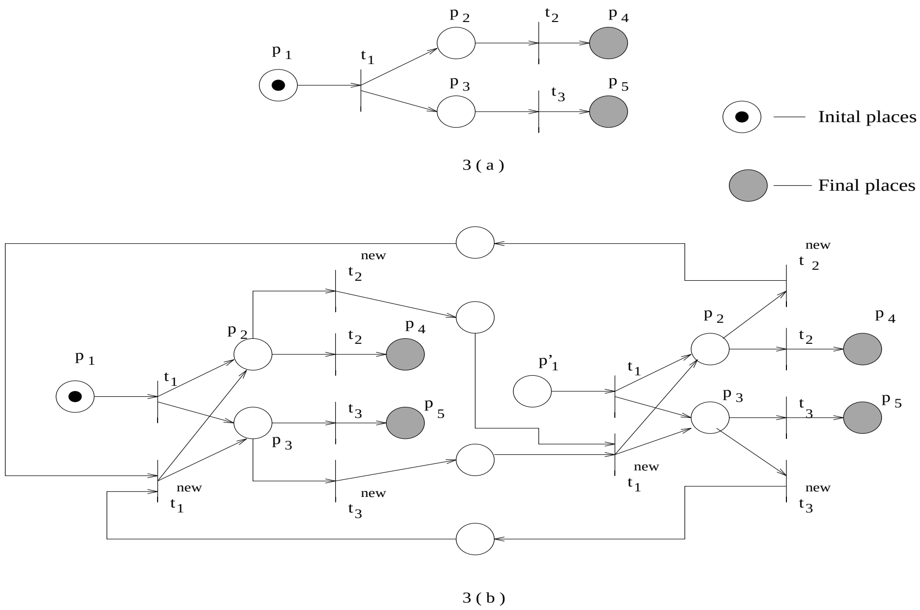

Fig. 3. $PL(3b) = PL(3a)^{+}$

- $T = T_1 + T_2 + T^{12}_{new} + T^{21}_{new}$;
  $T^{12}_{new} = \bigcup_{M^1_f \in \mathcal{F}^1} ({}^{\bullet}(M^1_f) \cup (M^2_0)^{\bullet})$; $T^{21}_{new} = \bigcup_{M^2_f \in \mathcal{F}^2} ({}^{\bullet}(M^2_f) \cup (M^1_0)^{\bullet})$
- $\ell = \ell_1 \cup \ell_2 \cup \{(t \rightarrow \ell_i(t)) \mid t \in (T^{12}_{new} \cup T^{21}_{new}) \cap T_i, i = 1, 2\}$
- $F = F_1 \cup F_2 \cup F^{12}_{new} \cup F^{21}_{new}$;
  $F^{12}_{new} = \{(p,t) \in P_1 \times T^{12}_{new} \mid (p,t) \in F_1\}$
  $\cup \{(t,(p_1,p_2)) \in T^{12}_{new} \times P^{12}_{new} \mid (t,p_1) \in F_1\}$
  $\cup \{((p_1,p_2),t) \in P^{12}_{new} \times T^{12}_{new} \mid (p_2,t) \in F_2\}$
  $\cup \{(t,p) \in T^{12}_{new} \times P_2 \mid (t,p) \in F_2\}$;
  $F^{21}_{new} = \{(p,t) \in P_2 \times T^{21}_{new} \mid (p,t) \in F_2\}$
  $\cup \{(t,(p_1,p_2)) \in T^{21}_{new} \times P^{21}_{new} \mid (t,p_1) \in F_2\}$
  $\cup \{((p_1,p_2),t) \in P^{21}_{new} \times T^{21}_{new} \mid (p_2,t) \in F_1\}$
  $\cup \{(t,p) \in T^{21}_{new} \times P_1 \mid (t,p) \in F_1\}$
- $M_0 = M^1_0$; $\mathcal{F} = \mathcal{F}^1 \cup \mathcal{F}^2$

As for sequencing, assume that $\mathcal{N}_1$ is behaved. Using the two copies maintains the source, sink and final width properties, and the argument for parallel sub-runs follows the one used above for sequencing.

Proposition 6 *If $\mathcal{N}_1$ is place complemented and behaved, then $PL(\mathcal{N}) = PL(\mathcal{N}_1)^{+}$.*

3.4 Parallel composition

The parallel composition of two 1-safe net systems $\mathcal{N}_i = (P_i, T_i, F_i, M_0^i, \mathcal{F}^i, \ell_i), i = 1, 2$ is the simple disjoint union $\mathcal{N} = (P, T, F, M_0, \mathcal{F}, \ell)$ where

- $P = P_1 + P_2$; $T = T_1 + T_2$; $\ell = \ell_1 \cup \ell_2$; $F = F_1 \cup F_2$
- $M_0 = M_0^1 \cup M_0^2$; $\mathcal{F} = \{M_f^1 \cup M_f^2 \mid M_f^1 \in \mathcal{F}^1,\ M_f^2 \in \mathcal{F}^2\}$

Clearly if both nets $\mathcal{N}_i$ are behaved, so is $\mathcal{N}$, the construction being designed to preserve the parallel sub-run and final width properties.

Proposition 7 $PL(\mathcal{N}) = PL(\mathcal{N}_1)||PL(\mathcal{N}_2)$.

3.5 Net subclasses

Definition. *The subclass of net systems built up from the atomic systems using the constructions for union, sequencing, iteration and parallel composition is called* (1-safe) SR-systems.

Theorem 8 *Given a series-rational expression e, one can construct an SR-system $\mathcal{N}$ such that $PL(\mathcal{N}) = L(e) \subseteq SP(A)$.*

Proof. By induction, using the Proposition corresponding to each operation.

All posets accepted by an SR-system are N-free. By constructing the reachability graph of a net system and running through all four-tuples of transitions, the N-free property can be algorithmically checked (using polynomial space). We conjecture that there is a polynomial time algorithm to check that a 1-safe net system is SR.

For net systems corresponding to the outermost/right series-rational expressions, we look at some well-known subclasses of net systems. (These are, in fact, the motivation for defining these expressions.)

An *S-system* [11] is a net system where neither forking ($|t^\bullet| > 1$) nor joining ($|{}^\bullet t| > 1$) transitions are allowed. In other words, for each transition, its pre-set and post-set are both singletons. The posets in the language accepted by an S-system satisfy the property that they are disjoint unions of chains, that is, they are outermost parallel.

A *communication-free (CF) system* [5] is one where joining transitions are not allowed. The posets accepted by a CF-system are right parallel.

The table in Figure 4 summarizes the closure properties of our net subclasses. Counter-examples are also listed.

Theorem 9 *Given a right series-rational expression e, one can construct a 1-safe CF-system $\mathcal{N}$ such that $PL(\mathcal{N}) = L(e) \subseteq RSP(A)$.*

Theorem 10 *Given an outermost series-rational expression e, one can construct a 1-safe S-system $\mathcal{N}$ such that $PL(\mathcal{N}) = L(e) \subseteq OSP(A)$.*

Closure under	*Sequencing*	*Iteration*	*Union*	*Parallel*
1-safe SR-systems	Yes	Yes	Yes	Yes
1-safe CF-systems	No, $(a\|\|b)c$	No, $(a\|\|b)^+$	No, $(a\|\|b) \cup c$	Yes
1-safe S-systems	No, $a(b\|\|c)$	No, $(a\|\|b)^+$	No, $(a\|\|b) \cup c$	Yes

Fig. 4. Constructions

4 From nets to expressions

All the formal language machinery is now in place, and we can start the job of extracting syntax from systems. We crucially use the fact that for 1-safe systems the width of the sp-terms accepted is always bounded.

4.1 1-safe nets

Lemma 11 *Given a 1-safe system $\mathcal{N}$, there is a series-rational expression for $PL(\mathcal{N}) \cap SP(A)$.*

Proof. Our proof is by induction on the width of $PL(\mathcal{N}) \cap SP(A)$. We construct expressions for the following sets:

$P^k_{M \to N}$ is the set of sp-terms of width k which are runs of $\mathcal{N}$ from the marking M to the marking N, such that the term is a parallel composition (or an atomic action).

$S^k_{M \to N}$ is the set of sp-terms of width k which are runs of $\mathcal{N}$ from the marking M to the marking N, such that the term is a sequential product (or an atomic action).

We will let $L^k_{M \to N} = P^k_{M \to N} \cup S^k_{M \to N}$.

We now define the sets of parallel terms. $P^1_{M \to N}$ is the set of all a such that a transition labelled a takes M to N. For $k > 1$, we have:

$$P^k_{M \to N} = \bigcup_{\substack{j_1 + j_2 = k \\ M_1 + M_2 = M, N_1 + N_2 = N}} L^{j_1}_{M_1 \to N_1} || L^{j_2}_{M_2 \to N_2}.$$

The construction of $S^k_{M \to N}$ is from $P^k_{M \to N}$ and follows traditional automata-theoretic lines. We will need to work with a specified set of connected intermediate markings R. We define by induction on its size $|R|$ the set $S^{k,R}_{M \to N}$ of terms from $S^k_{M \to N}$ which only use intermediate markings from the set R. We let $L^{k,R}_{M \to N} = P^k_{M \to N} \cup S^{k,R}_{M \to N}$.

Then $S^{k,\emptyset}_{M \to N}$ is nothing but $P^k_{M \to N}$. Inductively, for $r \notin R$, we have

$$S^{k,R \cup \{r\}}_{M \to N} = S^{k,R}_{M \to N} \cup \bigcup_{\substack{j_1, j_2, j_3 \leq k \\ j_1 = k \text{ or } j_2 = k \text{ or } j_3 = k}} L^{j_1,R}_{M \to r} (L^{j_2,R}_{r \to r})^* L^{j_3,R}_{r \to N}. \qquad (2)$$

This is a generalization of the usual McNaughton-Yamada expression used for word languages. The conditions on the union ensure that the width of the term added is k.

By taking the entire set of such intermediate markings, we have the desired series-rational expression.

Theorem 12 *The following classes of sp-languages are equivalent:*

(1) The series-rational languages.

(2) The poset languages accepted by 1-safe SR-systems.

(3) The sp-languages accepted by 1-safe systems.

(4) The recognizable sp-languages of bounded width.

(5) The MSO-definable sp-languages of bounded width.

Proof. (1⇒2⇒3): Theorem 8.

(3⇒1): Let K be the poset language accepted by a 1-safe system. Hence $L = K \cap SP(A)$ is an sp-language. By the lemma above, there is a series-rational expression for it.

(1⟺4): From [13], we know that the series-rational sp-languages are the recognizable sp-languages of bounded width. (The paper also gives a purely algebraic condition, expressed in terms of nilpotency of the parallel product, which corresponds to bounded width.)

(4⟺5): From [12], we know that these coincide with the MSO-definable sp-languages of bounded width.

The paper [13] also characterizes these languages as those accepted by certain "branching automata". It would be interesting to find a direct translation between these and 1-safe nets preserving the accepted sp-language. From the results of [13], we also get:

Corollary 13 *The languages accepted by 1-safe SR-systems are closed under intersection, direct and inverse morphism.*

4.2 Subclasses

Theorem 14 *The following classes of rsp-languages are equivalent, and closed under intersection.*

(1) The right series-rational languages.

(2) The poset languages accepted by 1-safe CF-systems.

(3) The rsp-languages accepted by 1-safe systems.

(4) The recognizable rsp-languages of bounded width.

(5) The MSO-definable rsp-languages of bounded width.

Proof. (1⇒2⇒3): Theorem 9.

(3⇒1): Let K be the poset language accepted by a 1-safe system. Then it accepts the rsp-language $L = K \cap RSP(A)$. Now $K \cap SP(A)$ is series-rational and $RSP(A)$ is MSO-definable, therefore series-rational by Theorem 12, so their intersection L is series-rational by Corollary 13. Since $L \subseteq RSP(A)$, the generating expression for it must be right series-rational.

(1⟺4⟺5): Proposition 3 and Proposition 2. The closure under intersection follows.

By simple modifications of the preceding arguments, we get:

Theorem 15 *The following classes of osp-languages are equivalent, and closed under intersection.*

(1) The outermost series-rational languages.

(2) The poset languages accepted by 1-safe S-systems.

(3) The osp-languages accepted by 1-safe systems.

(4) The recognizable osp-languages of bounded width.

(5) The MSO-definable osp-languages of bounded width.

Discussion. Figure 5 summarizes the results of this paper. We believe our characterizations also work for k-safe nets, using the results of [2]. Extending the work to general (unsafe) nets is more challenging.

Petri nets are typically meant to model infinite behaviour. A theory of ω-rational expressions for N-free poset languages has been developed by Kuske [12], and a similar development can be done in our setup without too much difficulty [18].

Nets have a richer interaction structure than can be described by sp-terms. In our setup, the obvious way to move towards the full class of 1-safe Petri nets would be to include in our terms an operation corresponding to synchronization (as has been done in PBC [1].) However, carrying all the equivalences forward seems to be difficult.

References

1. E. Best, R. Devillers and M. Koutny. *Petri net algebra*, Springer (2001).
2. E. Best and H. Wimmel. Reducing k-safe Petri nets to pomset-equivalent 1-safe Petri nets, *Proc. ATPN*, Aarhus (M. Nielsen and D. Simpson, eds.), LNCS **1825** (2000) 63–82.
3. P. Degano, R. de Nicola and U. Montanari. A partial order semantics for CCS, TCS **75** (1990) 223–262.
4. V. Diekert and G. Rozenberg, eds. *The book of traces*, World Scientific (1995).
5. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes, Fund. Inf. **30** (1997) 23–41.

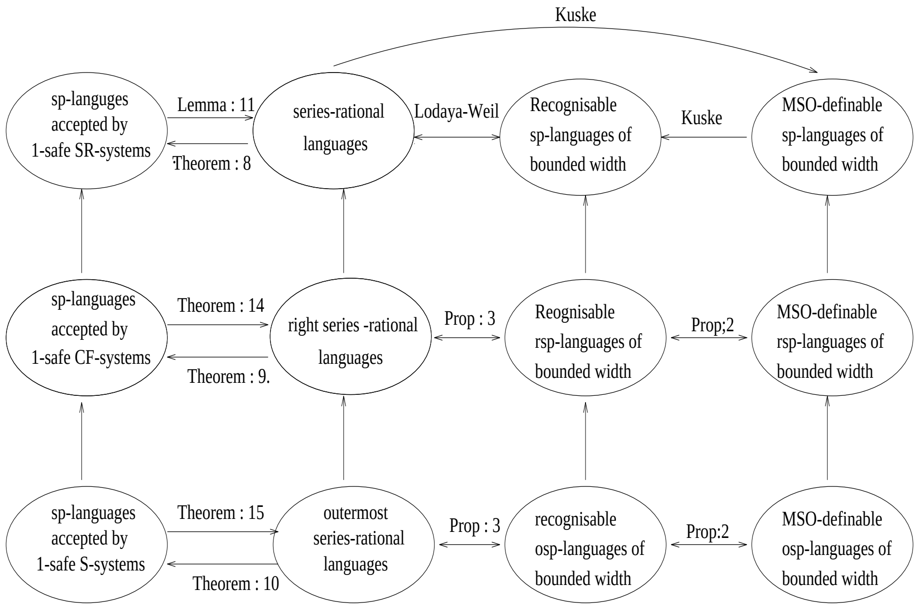

Fig. 5. Relationships

6. V.K. Garg and M.T. Ragunath. Concurrent regular expressions and their relationship to Petri nets, TCS **96** (1992) 285–304.
7. J.L. Gischer. The equational theory of pomsets, TCS **61** (1988) 199–224.
8. R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency, *Proc. PARLE*, Eindhoven (J.W. de Bakker, A.J. Nijman and P.C. Treleaven, eds.), Vol. 2, LNCS **259** (1987) 224–242.
9. U. Goltz and A. Mycroft. On representing CCS programs by finite Petri nets, *Proc. ICALP*, Antwerpen (J. Paredaens, ed.), LNCS **172** (1984) 196–208.
10. J. Grabowski. On partial languages, Fund. Inform. **IV** (1981) 427–498.
11. M.H.T. Hack. *Analysis of production schemata by Petri nets*, M.S. thesis, Dept. Elec. Engg., MIT (1972).
12. D. Kuske. Towards a language theory for infinite N-free posets, TCS **299** (2003), 347–386.
13. K. Lodaya and P. Weil. Series-parallel languages and the bounded-width property, TCS **237** (2000) 347–380.
14. E. Ochmański. Regular behaviour of concurrent systems, Bull. EATCS **27** (1985) 56–67.
15. E.-R. Olderog. *Nets, terms and formulas*, Cambridge (1991).
16. C.-A. Petri. Fundamentals of a theory of asynchronous information flow, *Proc. IFIP*, Münich (C.M. Popplewell, ed.), North-Holland (1962) 386–390.
17. V. Pratt. Modelling concurrency with partial orders, IJPP **15**(1) (1986) 33–71.
18. D. Ranganayakulu and K. Lodaya. Infinite series-parallel posets of 1-safe nets, *Proc. Algorithms & Artificial Systems*, (P. Thangavel, ed.), Allied (2003) 107–124.
19. W. Reisig. *Petri nets, an introduction*, Springer (1985).
20. D. Taubner. *Finite representations of CCS and TCSP programs by automata and Petri nets*, LNCS **369** (1989).