Solving Recursive Net Equations

Eike Best[⋆] and Maciej Koutny[⋆⋆]

Abstract. This paper describes a denotational approach to the Petri net semantics of recursive expressions. A domain of nets is identified such that the solution of a given recursive equation can be found by fixpoint approximation from some suitable starting point. In turn, a suitable starting point can be found by fixpoint approximation on a derived domain of sets of trees. The paper explains the theory on a series of examples and then summarises the most important results.
Keywords: Denotational semantics, Petri nets, Recursion.

1 Introduction

Process algebras [1, 10, 11, 13, 18] are widely used as models for the specification of concurrent programs, and so are Petri nets and their extensions [4, 12, 16, 21]. In a process algebra one may study the algebraic properties of connectives related to operators known from programming languages. In particular, a general operator expressing (guarded or unguarded) recursion is part of many process algebras. Its transition system semantics is well-understood (e.g. [1]). In Petri net theory, emphasis tends to be on the partial order semantics of a concurrent system. In order to combine the two models and their individual advantages, it has frequently been proposed that a Petri net semantics be given to a process algebra [3, 8, 13, 17, 19, 22]. In this context, the question arises how the general recursion operator could be given a Petri net semantics. Although this question has been addressed in many of the papers just mentioned and also in [6, 14], denotational fixpoint methods that seem appropriate for recursion have not yet been completely satisfactorily applied in a Petri net setting (see, for example, the list of Further Research Topics given in [19]).

The purpose of the present paper is to close this gap by demonstrating the use of fixpoint methods in combination with elementary Petri nets. Our approach makes significant use of Devillers' idea to describe transition refinement in a Petri net by name trees [6]. The structure of this paper is as follows. First, we define a syntactic domain of expressions (or terms), wherein recursion can be expressed by a system of equations. Then we define a semantic domain which delineates the class of Petri nets that will be associated to expressions. This domain is equipped with a partial ordering suitable for approximating solutions of recursive equations. Both domains are defined in Section 2. The remaining sections of the paper develop a theory explaining that every term equation always has at least one (countable) Petri net solution, that there may not always be a unique solution, and how to compute a solution. The approximation of a solution is based on successive refinements in the style of [6]. For this reason, we explain a general refinement operator in Section 3. In Section 4 we define Petri net equations, which serve as the semantics of term equations. The successive refinements approximating a solution of a net equation have to start with some first element (Section 5). Unless re-

[⋆] Institut für Informatik, Universität Hildesheim, Marienburger Platz 22, D-31141 Hildesheim, E.Best@informatik.uni-hildesheim.de

[⋆⋆] Department of Computing Science, University of Newcastle upon Tyne, NE1 7RU, U.K., Maciej.Koutny@newcastle.ac.uk

cursion is guarded, there are several possible alternatives for choosing such a starting point depending on the equation system under consideration. We examine an auxiliary domain in which this starting point can be defined as the solution of another equation. This auxiliary domain is defined in Section 6. Section 7 is devoted to reviewing some of the main results of our theory in general terms. In Section 8, finally, having shown that solutions always exist, we show that the preceding theory allows to deduce that every net equation system associated with any given term equation system has a Petri net solution, and thus the latter has a good semantics in terms of Petri net fixpoints.

2 Expressions and boxes

In this paper we deal with recursion in its most general form. Therefore, we define a process algebra and a tailor-made class of labelled Petri nets which are general with regard to recursion but minimal in other respects.

To define the process algebra, let A be a countable alphabet of action names and let $\mathcal{X}$ be a countable set of recursion variables. We use $a, b, a_i, \ldots$ as elements of A and $X, Y, X_j, \ldots$ as elements of $\mathcal{X}$. A *recursion-free expression* or *term* E, by definition, is generated by the syntax

$$E \ ::= \ a \mid X \mid E;E \mid E+E \mid E\|E,$$

where the connectives ; (sequence), + (choice) and $\|$ (disjoint parallelism) have their usual meaning. A *recursive equation system* is a countable set of equations of the form

$$\{X_j = E_j \mid j = 1, 2, \ldots\},$$

such that the X_j are distinct variables and the E_j are recursion-free expressions. Our syntactic domain is defined to be the set of all closed recursive equation systems, which will later just be called *term equations* or (if no misunderstanding is possible) *equations*, where 'closed' means that every variable occurring on the right hand side of some equation also occurs on the left hand side of the same or a different equation. An equation system will be called a (recursive) *expression* if some seed, that is, some fixed equation, is specified. This is often done in the form

$$\mu X_i.\{X_j = E_j \mid j = 1, 2, \ldots\},$$

where i is one of the indices $1, 2, \ldots$ For reasons of brevity and clarity, we define the main constructions and state the results of this paper only for the special case of one recursive equation. This is also an expression, since the (only) index is automatically fixed. All constructions and results can easily be generalised to the case of more than one equation. We indicate such generalisations at the appropriate points later in the paper. We use the following six expressions, all of which are closed, as our running examples:

$$\begin{array}{ll} \text{Example 1:} & X = (a;X;b) \\ \text{Example 2:} & X = ((a;X;b)+c) \\ \text{Example 3:} & X = (a\|X) \\ \text{Example 4:} & X = (a;X) \\ \text{Example 5:} & X = ((a\|b)+X) \\ \text{Example 6:} & X = X. \end{array} \quad (1)$$

Intuitively, expression 1 can do any number of a-moves (including infinitely many), but never any b-move, and it never terminates properly either. Expression 2 can do whatever expression 1 could and, in addition, at any point it can make a single c-move followed by up to as many b-moves as there have been a-moves before the c-move; its execution terminates after the last b-move. In expression 3, any (finite) number of concurrent a-moves can happen, but it is not clear whether or not the execution can terminate after an infinite number of a-moves[1]. Expression 4 has the same

[1] We will return to this question later.

behaviour - but obviously not the same structure - as expression 1. Example 5 expresses a choice between one of infinitely many parallel moves of a single a and a single b such that once one of the a's (or b's) is chosen, the corresponding b (or a, respectively) must be taken. Expression 6, finally, can make no move at all.

Our objective is to associate one or more elementary (possibly infinite) Petri nets canonically to each equation. We address the questions whether this is always possible and if so, how the Petri net(s) can be computed. In order to find the answers, monotonic functions defined over a suitable semantic domain will be associated to every equation. It turns out that these functions always have fixpoints amongst which we will find our solutions. If there are two or more fixpoints then different solutions can be obtained by starting their approximation from different points.

To define the semantic domain, we recall that an (arc-weighted) *net* is a triple (S, T, W) such that S and T are disjoint sets (of *places* and *transitions*, respectively) and W is a *weight function* from the set $((S \times T) \cup (T \times S))$ to the set $\mathbf{N}$ of natural numbers. A net (S, T, W) is called an *S-net* iff

$$\forall t \in T\colon (\sum_{s \in S} W(s,t)) = 1 = (\sum_{s \in S} W(t,s)).$$

A *labelled net*, for our purpose, is a four-tuple $\Sigma = (S, T, W, \lambda)$ such that (S, T, W) is a net and λ is a partial function from S to the set $\{e, x\}$ and from T to the set A. If $\lambda(s) = e$ then s is called an *entry place*. If $\lambda(s) = x$ then s is called an *exit place*. If $\lambda(s)$ is undefined then s is called an *internal place*. If $\lambda(t) = a$ then, intuitively speaking, an execution of t denotes an execution of the action a. We will see later how $\lambda(t)$ being undefined is to be interpreted. If Σ is a labelled net, by convention, ${}^\bullet\Sigma$ and $\Sigma^\bullet$ denote the set of e-labelled places and the set of x-labelled places, respectively, of Σ. For any place $s \in S$, ${}^\bullet s$ ($s^\bullet$) denotes the set of input (output, respectively) transitions of s. For a set of places $R \subseteq S$, ${}^\bullet R = \bigcup\{{}^\bullet s \mid s \in R\}$ and $R^\bullet = \bigcup\{s^\bullet \mid s \in R\}$. Similar notation is used for transitions and sets of transitions. A *box* is defined to be a labelled net such that the following hold:

$$\begin{aligned} &{}^\bullet({}^\bullet\Sigma) = \emptyset = (\Sigma^\bullet)^\bullet \quad\wedge\quad {}^\bullet\Sigma \neq \emptyset \neq \Sigma^\bullet \\ &\wedge\quad \forall t \in T\colon {}^\bullet t \neq \emptyset \neq t^\bullet. \end{aligned}$$

A box is called *plain* or *non-operative* if $\lambda|_T$ is a function[2]. A finite box is called an *operator box* iff λ is undefined for all the transitions in T, the underlying net is the disjoint union of connected S-nets, each such S-net comprising exactly one entry place and exactly one exit place, and the net is side-condition-free[3], i.e., $\forall t \in T\colon {}^\bullet t \cap t^\bullet = \emptyset$. A *marking* M of (S, T, W, λ) is a function from S to the natural numbers. A marking is *safe* iff $M(S) \subseteq \{0, 1\}$. A safe marking can and will be represented as a subset of S. A *marked box* (Σ, M) is a box Σ together with a marking M. The *initially marked box corresponding to* Σ is defined to be $(\Sigma, {}^\bullet\Sigma)$. A box Σ is called *safe* iff all markings in $[{}^\bullet\Sigma\rangle$, that is, all markings reachable in $(\Sigma, {}^\bullet\Sigma)$, are safe. A safe box Σ is called *clean* iff all markings M in $[{}^\bullet\Sigma\rangle$ satisfy $\Sigma^\bullet \subseteq M \Rightarrow \Sigma^\bullet = M$.

Lemma: Operator boxes are safe and clean.
Proof: Directly from the S-net decomposability property.

We equip the set of plain boxes with a relation $\sqsubseteq$. A plain box $\Sigma = (S, T, W, \lambda)$ *is included in* a plain box $\Sigma' = (S', T', W', \lambda')$ (in symbols, $\Sigma \sqsubseteq \Sigma'$) iff

$$\begin{aligned} &S \subseteq S' \\ &T \subseteq T' \\ &\forall (x,y) \in (S \times T) \cup (T \times S)\colon W(x,y) = W'(x,y) \\ &\forall x \in S \cup T\colon \lambda(x) = \lambda'(x). \end{aligned}$$

Clearly, the relation $\sqsubseteq$ is a partial order. It also has least upper bounds: the places and

[2] The meaning of the word 'non-operative' will become clear later.

[3] This restriction is added for reasons of simplicity; it can be eliminated, however, as described in [6].

transitions of the limit $\Sigma = \lim_{k\geq 0} \Sigma^k$ of a sequence $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \ldots$ are, the unions of the places and transitions, respectively, of the boxes Σ^k in the sequence. For two elements x and y of Σ (one of them a place, the other a transition), the weight $W(x, y)$ is defined as the weight between x and y in the first Σ^k in which both of them occur (note that by the definition of $\sqsubseteq$, the weight does not change in subsequent approximations). Finally, for an element x in Σ, the label $\lambda(x)$ is defined as the label of x in the first and hence in all Σ^k containing x. For such chains we have:

Lemma: If all plain boxes Σ^k of a $\sqsubseteq$-chain $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \ldots$ are safe and clean then so is the limit $\lim_{k\geq 0} \Sigma^k$.

Proof: From the fact that any transition sequence of the limit can be recovered as a transition sequence of some approximation Σ^k.

A class of (so to speak) rather small plain boxes is defined as follows. A box is called an *ex box* iff its set of transitions and consequently also its weight function are empty. An *ex* box thus consists only of a nonempty set of places labelled e and a nonempty set of places labelled x.

The nets associated to term equations will be taken from the semantic domain of plain boxes defined above. The relation $\sqsubseteq$ will be used to compute them by successive approximation.

3 A tree view of refinement

A key to our understanding of recursive nets is the use of labelled trees as place and transition names. Every plain name, such as $\mathit{1}, \mathit{2}$ etc. for places and $\mathbf{1}, \mathbf{2}, t, u$ etc. for transitions, will be viewed as a special tree with a single root (labelled with $\mathit{1}, \mathit{2}, \mathbf{1}, \mathbf{2}, t, u$ etc.) which is also a leaf. Moreover, we will allow (degree-finite) trees as place or transition names, such as for example $\mathit{1}\mathbf{2}\mathit{5}$, which, by definition, is an abbreviation for the tree with root $\mathit{1}$, intermediate node $\mathbf{2}$ and leaf $\mathit{5}$[4]. Let us consider the two boxes Σ_1 and Σ_2 depicted in Figure 1. They can be written down formally in four-tuple notation as follows, where we specify the functions W_1 and W_2 by listing the pairs for which their value is 1 (all other values are implicitly assumed to be 0):

$$\begin{aligned}
\Sigma_1 = (&\{\mathit{1}, \mathit{2}, \mathit{3}, \mathit{4}\}, \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}, \{(\mathit{1}, \mathbf{1}), (\mathit{1}, \mathbf{2}),\\
&(\mathit{3}, \mathbf{2}), (\mathit{3}, \mathbf{3}), (\mathbf{1}, \mathit{2}), (\mathbf{2}, \mathit{2}), (\mathbf{2}, \mathit{4}), (\mathbf{3}, \mathit{4})\},\\
&\{(\mathit{1}, e), (\mathit{3}, e), (\mathit{2}, x), (\mathit{4}, x), (\mathbf{1}, a)\})\\
\Sigma_2 = (&\{\mathit{5}, \mathit{6}, \mathit{7}, \mathit{8}\}, \{t, u\},\\
&\{(\mathit{5}, t), (\mathit{7}, u), (t, \mathit{6}), (u, \mathit{8})\},\\
&\{(\mathit{5}, e), (\mathit{7}, e), (\mathit{6}, x), (\mathit{8}, x), (t, a), (u, b)\}).
\end{aligned}$$

Notice that λ_1 is undefined for transitions $\mathbf{2}$ and $\mathbf{3}$ of Σ_1. In our interpretation, this indicates that $\mathbf{2}$ and $\mathbf{3}$ may be refined by some plain box(es). Let us examine, in particular, the refinement of $\mathbf{2}$ and $\mathbf{3}$ by the same box Σ_2. Refinement of a transition by a box, in general, means that every input place of the transition and every e-place of the box are combined to give a new place, and similarly with the output places of the transition and the x-places of the box, and that the new places are connected in the canonical way (see also [3, 9, 19]). The reader may check that doing this in either order in the example at hand, i.e., refining $\mathbf{3}$ after $\mathbf{2}$ or the other way round, produces exactly the same net. This net is shown in Figure 2.

Using place and transition names in the shape of trees is a formal way of expressing the fact that refinement does not depend on the ordering in which it is done or, more precisely, of expressing simultaneous refinement. Thus when both $\mathbf{2}$ and $\mathbf{3}$ are refined by the box Σ_2, the input place shared by the two transitions, i.e., place $\mathit{3}$, has to be merged with the e-places $\mathit{5}$ and $\mathit{7}$ twice, once for transition $\mathbf{2}$ and once for transition $\mathbf{3}$, resulting in the four upper right places of Figure 2; moreover, the input place $\mathit{1}$ which is connected only to transition $\mathbf{2}$ but not to transition $\mathbf{3}$ has to be merged only once with $\mathit{5}$ and $\mathit{7}$, resulting in the two

[4] For brevity, we will refer to a labelled node using its label.

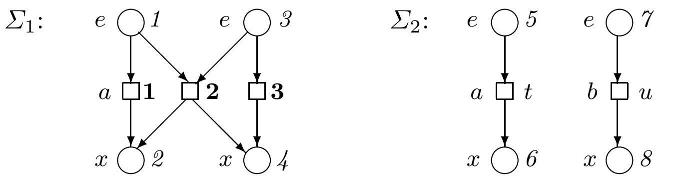


Fig. 1. A net to be refined (left) and the refining net (right)

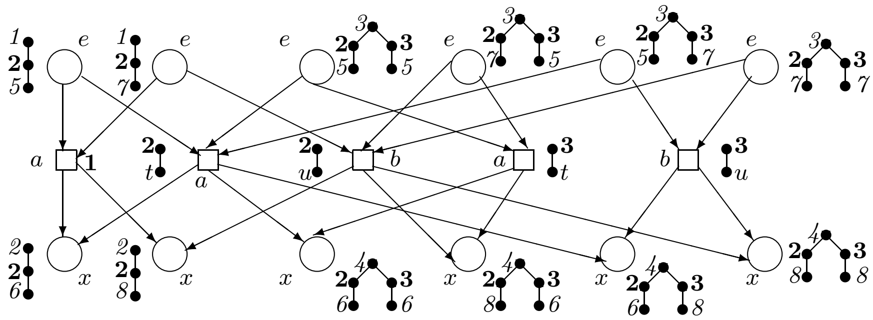


Fig. 2. A refined net

upper left hand places of Figure 2. This merging is described by the trees that serve as e-place names in the figure, and the same is true for the x-places. The transition names are also trees describing their origins. For example, the tree name of the second transition of Figure 2 reflects the fact that this transition comes from a refinement of transition **2** of Σ_1 by the transition t of Σ_2. We shall use a linear notation to express those trees. To this end, the expression $s\triangleleft(LT)$, where s is a node and LT is a list of trees, is a new tree where the trees of the list are appended to the root s. Thus, the set of places in Figure 2 can be written in linear form as follows:

$$\{\mathit{1}\triangleleft(\mathbf{2}\triangleleft(\mathit{5})), \mathit{1}\triangleleft(\mathbf{2}\triangleleft(\mathit{7})),$$
$$\mathit{3}\triangleleft(\mathbf{2}\triangleleft(\mathit{5}), \mathbf{3}\triangleleft(\mathit{5})), \mathit{3}\triangleleft(\mathbf{2}\triangleleft(\mathit{7}), \mathbf{3}\triangleleft(\mathit{5})), \ldots\}.$$

Formally, the fact that the refinement of transition **2** commutes with the refinement of transition **3** is expressed by the fact that, for instance, $\mathit{3}\triangleleft(\mathbf{2}\triangleleft(\mathit{5}), \mathbf{3}\triangleleft(\mathit{5}))$ is the same tree (and hence the same place) as $\mathit{3}\triangleleft(\mathbf{3}\triangleleft(\mathit{5}), \mathbf{2}\triangleleft(\mathit{5}))$ resulting from different orders of refinement, and similarly for the other place names.

To complete the definition of the domain of nets in which we will be solving recursive equations, we assume that each transition name is a finite tree with nodes being transition names, and each place name is a finitely branching (possibly infinite) tree of the form $t_1\triangleleft(t_2\triangleleft(\ldots t_n\triangleleft(s\triangleleft(LT))))$, where $t_1, \ldots, t_n$ ($n \geq 0$) are transition names and s is a place name (c.f. Figure 2 and 4).

4 Net equations

In the following, we will denote by Σ, Σ_1 etc. plain boxes with transitions t, u, etc., and by Ω, Ω_1 etc. operator boxes with transitions $\mathbf{1}, \mathbf{2}, \mathbf{3}$ etc. Let Ω have n transitions, ordered in some arbitrary but fixed way; we will always use the ordering indicated by the names $\mathbf{1}, \mathbf{2}, \mathbf{3}$ etc. Then the equation

$$X = (\Omega; \Pi_1, \ldots, \Pi_n)$$

defines X to be a box arising from the simultaneous refinement of the n transitions $\mathbf{1}, \mathbf{2}$ etc. of Ω by the respective plain boxes Π_1, Π_2 etc. Note that this new box is plain as well, since all refining boxes Σ_i are plain and all transitions of the operator box Ω are replaced. Note also that since Ω is finite, the property that place trees are finitely branching is preserved through refinement. We now generalise this type of equation by allowing, in the list $\Pi_1, \ldots, \Pi_n$, not just plain boxes but also recursion variables $X \in \mathcal{X}$ as place-holders for other plain boxes. More precisely, let us consider a countable system of equations of the following form:

$$\begin{array}{ll} \text{line } 1: & X_1 = (\Omega^1; \Pi_1^1, \Pi_2^1, \ldots) \\ \text{line } 2: & X_2 = (\Omega^2; \Pi_1^2, \Pi_2^2, \ldots) \\ \text{line } 3: & X_3 = (\Omega^3; \Pi_1^3, \Pi_2^3, \ldots) \\ \cdots & \end{array} \tag{2}$$

where the left hand sides of the equations are mutually distinct recursion variables and the right hand sides are tuples such that in each line j, the number of Π_i^j's equals the number of transitions of the corresponding Ω^j, and every Π_i^j is either a plain box or a variable from the set $\{X_1, X_2, X_3, \ldots\}$. Note that since Ω^j is finite by the definition of an operator box, every line of the equation system has finite length, but the number of equations may be infinite. A *solution* of the system (2) is an assignment of a plain box Σ_j to every variable X_j such that every equation is valid if every occurrence of X_j is replaced by Σ_j.

By definition, the *dependency graph* of (2) is the digraph whose nodes are the set of variables $\{X_1, X_2, X_3, \ldots\}$ and whose arcs (X_i, X_j) indicate that X_j occurs on the right hand side of the equation having X_i as its left hand side. The equation system (2) is called *recursion free* if its dependency graph has no forward-infinite directed paths (in particular, the graph must be acyclic). In particular, if none of the right hand sides of (2) has a recursion variable then the system is recursion free since its dependency graph has no arcs at all. If the system (2) is recursion free then using simultaneous refinement a set of plain boxes $\Sigma_1, \Sigma_2, \ldots$ solving it can be calculated by executing the following procedure: starting with equations that have no variables on their right hand sides, use simultaneous refinement repeatedly until all left hand sides have been calculated. That this procedure is correct follows from König's Lemma, the finiteness of the operator boxes and the absence of infinite directed paths in the dependency graph. Note that the boxes Σ_j are unique in that case.

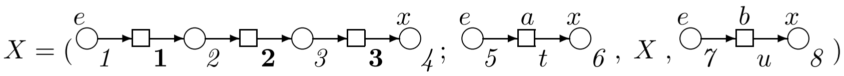
Lemma: If the system (2) is recursion free and if all plain boxes occurring on the right hand sides are safe and clean then the same is true for the box Σ_j obtained for the j'th equation ($j = 1, 2, \ldots$).

Proof: Operator boxes are safe and clean, and the property propagates through refinement as proved in [6].

In the next section we show that the equation system (2) always has a solution, even if it is not recursion free. In the general case, this solution may not be unique.

5 Approximation of solutions

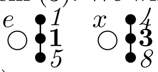
To start with, let us consider the term equation $X = (a; X; b)$ of the first example in (1). Figure 3 renders this equation as a net equation. Note how the term equation $X = (a; X; b)$ and the net equation (3) correspond to each other. The right hand side $(a; X; b)$ of the former is translated into an operator box with three unlabelled transitions which are to be refined by their corresponding boxes, which are specified explicitly for the transitions $\mathbf{1}$ and $\mathbf{3}$ and implicitly (as the variable X) for transition $\mathbf{2}$. We will show later (Section 8) that every term equation system can be turned canonically into a corresponding system of net equations. For the time being, we



(3)

Fig. 3. Equation associated to $X = (a; X; b)$

continue to restrict our attention to only one equation, i.e., to the special case $n = 1$.

Let us examine how a solution of the net equation (3) associated with $X = (a; X; b)$ might be constructed. Suppose that there exists already an approximation Σ^0 as a starting point. Then it is a good idea to insert that approximation in the place of the X on the right hand side of equation (3). This yields (by the simultaneous refinement of the three transitions $\mathbf{1}, \mathbf{2}, \mathbf{3}$ of the operator box of (3)) a new plain box Σ^1 which, hopefully, is a better approximation. We may then construct another approximation Σ^2 by the same principle, and so on. Again hopefully, the limit of the sequence $\Sigma^0, \Sigma^1, \ldots$ so constructed solves the equation system (3). We will see later that the ex box $\Sigma^0 =$ 

(with these particular names) serves as a good starting point of the approximation of a solution of equation (3). Figure 4 shows the first three approximations resulting from this starting box, where the refinement is done according to the rules given in Section 3, except that we do not represent all name trees explicitly (the reader may easily guess the missing ones by analogy). A reason why this particular box Σ^0 is a good starting point is that by this choice, we get an embedding of Σ^j into Σ^{j+1} in the sense of the partial order $\sqsubseteq$. More precisely, we have $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \ldots$, where $\sqsubseteq$ is the subnet relation defined at the end of Section 2. Figure 5 shows the limit of the infinite series of boxes whose first three elements are shown in Figure 4. The reader may check that this box satisfies the equation (3) in the strictest possible sense of the word. To see this, replace the variable X on the right hand side of equation (3) by the plain box $\Sigma_{X=(a;X;b)}$ shown in Figure 5, perform the refinement of the three transitions $\mathbf{1}, \mathbf{2}, \mathbf{3}$ according to the rules given in Section 3, and check that the result is not just isomorphic but even identical to $\Sigma_{X=(a;X;b)}$, i.e., it has the same places and the same transitions with the same connections and the same labels. Note also that, as might be expected, with the canonical initial marking $^\bullet\Sigma_{X=(a;X;b)}$ (i.e., a token on its e place), this solution has the same behaviour as the expression $X = (a; X; b)$.

It may be checked that this approach works similarly for the second example $X = ((a; X; b) + c)$, whose net equation is the same as equation (3) except that the operator box has one more transition $\mathbf{4}$ from the e-place to the x-place and the list of Σ_i^j on the right hand side has one more element with a c-labelled transition. In fact, if we use exactly the same starting box Σ^0 as before (with the same names and the same labels), then the approximation yields a limit which is very much like $\Sigma_{X=(a;X;b)}$ in Figure 5, except that the places of the upper line and the places of the lower line are joined by vertical transitions labelled c.

Once an appropriate starting point Σ^0 is chosen, the approximation of a solution of a given net equation is quite automatic (we just keep refining successive approximations into the right hand side of the equation under consideration and compute the limit) and leads to a solution in all cases. More precisely, we have the following result whose proof can be found in [15]:

Theorem: (i) Let Σ^0, Σ^1 be two boxes such that if Σ^1 results from applying the right hand side of the equation (2) to Σ^0 and $\Sigma^0 \sqsubseteq \Sigma^1$, then if this iteration is repeated

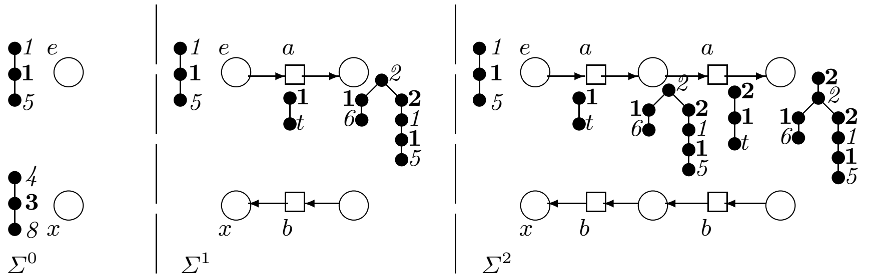


Fig. 4. Three approximations Σ^0, Σ^1 and Σ^2 of net equation (3)

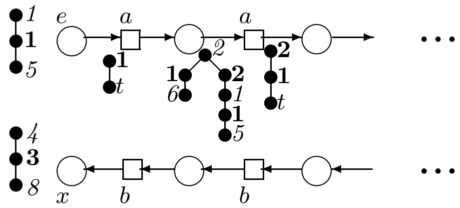


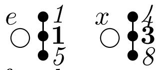
Fig. 5. A box $\Sigma_{X=(a;X;b)}$ solving the net equation (3)

we have $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \sqsubseteq \ldots$, and the box $\lim_{k\geq 0} \Sigma^k$ is a (safe and clean, provided that the nets on the right hand sides were safe and clean) solution of (2). Moreover, if the sets of e-places and the sets of x-places of Σ^0 and Σ^1 are the same then they are the same in all Σ^k and hence also in the limit.

(ii) If Σ is a solution of the equation (2) then by taking an *ex* box Σ^0 with entry places ${}^\bullet\Sigma$ and exit places $\Sigma^\bullet$, and by applying the same procedure as in (i), we obtain $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \sqsubseteq \ldots$ and $\Sigma = \lim_{k\geq 0} \Sigma^k$.

Note that (i) does not require Σ^0 to be an *ex* box. The requirement that the initial approximation is included in the next one already guarantees that the limit is a solution.

6 Tree equations

In this section we deal systematically with the question how to find a good starting point Σ^0 for the approximation. By the last theorem, we wish to ensure that such a starting point satisfies $\Sigma^0 \sqsubseteq \Sigma^1$ for the next approximation Σ^1. For instance, we may ask why the box  is so suitable a starting point for the approximation of the solutions of our first two expressions $X = (a; X; b)$ and $X = ((a; X; b) + c)$. First of all, it is a good idea always to start with an *ex* box, because these boxes are minimal with respect to the number of transitions they contain. It is even better (if possible) to start with an *ex* box containing only one e place and one x place, because these boxes are minimal with respect to $\sqsubseteq$. (Note that we have excluded the empty box[5])

Finally, by (ii) in the last theorem, we do not lose any solution by restricting our starting point only to *ex* boxes. Let us call this generic

[5] A reason for excluding the empty box is that the empty box, put in sequence with another box, would yield a non-safe net; there are also other reasons.

starting box

$$\underbrace{(P_X \cup Q_X, \emptyset, \emptyset, \{(p,e),(q,x) \mid p \in P_X, q \in Q_X\})}_{\Sigma^0}.$$

In order to exclude the empty box, the sets P_X of entry place and Q_X of exit places should both be nonempty (and preferrably singletons). They are disjoint, of course. The *ex* box Σ^0 is supposed to satisfy the property

$$\Sigma^0 \sqsubseteq \Sigma^1$$

in order to be the starting point of an approximation of a solution of the equation under consideration. Because of this, some conditions can be imposed on the sets P_X and Q_X, essentially by performing the refinement at P_X and Q_X and limiting the required inequality to these sets. As both are sets of trees, they may be defined as solutions of an inequation ranging over the complete lattice of sets-of-trees with the subset ordering relation. In order to express such a condition, we use two operations on sets of trees: the ordinary set union, and the appending construct $\lhd$ lifted to sets of trees. More precisely, $s \lhd (LST)$ creates a set of trees from a given node s and a list of sets of trees LST. For instance, $3 \lhd (\{4,5\},\{6\})$ equals the set of trees $\{3 \lhd (4,6), 3 \lhd (5,6)\}$, and $3 \lhd (\{4\})$ is the singleton set $\{3 \lhd (4)\}$. We also define $s \lhd (\ldots,\emptyset,\ldots) = \emptyset$.

In determining the conditions that P_X and Q_X have to satisfy, then, we simply carry out the refinement specified by the net equation under consideration, in moving from Σ^0 to Σ^1. Thus, for example, equation (3) (see Figure 3) leads to the following tree inequations for an *ex* box starting the approximation of $X = (a;X;b)$:

$$\begin{aligned} P_X &\subseteq 1 \lhd (\mathbf{1} \lhd (\{5\})) \\ \text{and}\;\; Q_X &\subseteq 4 \lhd (\mathbf{3} \lhd (\{8\})). \end{aligned} \tag{4}$$

To see why, consider the e-place (for P_X) and the x-place (for Q_X) of the operator box of equation (3) and check the way in which the transitions bordering on those places ($\mathbf{1}$ for P_X and $\mathbf{3}$ for Q_X) are refined by the corresponding simple boxes, viz. the first one for $\mathbf{1}$ and the third one for $\mathbf{3}$. In general, the right hand sides of such inequations can be expressed in terms of union and $\lhd$. As both operators are monotonic on the complete lattice of sets-of-trees, functions formed from them have fixpoints defined as usual. Moreover, every P_X and Q_X solving an inequation of the form (4) must be subsets, respectively, of P_X and Q_X solving the corresponding equation

$$\begin{aligned} P_X &= 1 \lhd (\mathbf{1} \lhd (\{5\})) \\ \text{and}\;\; Q_X &= 4 \lhd (\mathbf{3} \lhd (\{8\})). \end{aligned} \tag{5}$$

In our particular case, it so happens that both the inequation and the equation have only one nonempty solution, namely the two singleton sets of trees $P_X = \{1 \lhd (\mathbf{1} \lhd (5))\}$ and $Q_X = \{4 \lhd (\mathbf{3} \lhd (8))\}$. Note that these are exactly the places of the box Σ^0 we have chosen earlier for this example. In general, if P_X and Q_X solve the equation then they determine a starting *ex* box to which the second part of (i) in the previous theorem can be applied, that is, they remain constant throughout the approximation of the limit of (2). Conversely, by (ii), any solution of (2) determines a solution P_X and Q_X. As a result, there is a one-to-one correspondence between the solutions of an equation on nets and the solutions of the corresponding equation on (place) trees. Therefore, in what follows we can limit ourselves to considering the equations on trees only.

If the e-places and the x-places of an operator box touch only transitions that are to be refined by a constant plain box rather than a recursion variable, such as is the case in the two examples considered so far, then the right hand sides of the equations for P_X and Q_X are always constant and finite. In this case, the equation has a unique (finite) solution for P_X and Q_X (as a consequence, the net obtained as the limit is the least solution of the net equation with respect to the subnet relation $\sqsubseteq$). More generally, the same property holds if the system of net equations under consideration is *guarded*[6].

The equations for P_X and Q_X may become more complicated for unguarded recursion. This is illustrated by our third example $X = (a\|X)$ which contains X unguardedly at both ends (that is to say, the e-labelled front end and the x-labelled rear end). The net equation shown in Figure 6 corresponds to it. Note that equation (6) indeed matches $X = (a\|X)$ in the same way as equation (3) matches $X = (a; X; b)$. We now derive a box Σ^0 which is supposed to start the approximation for this example. Again we determine two (in)equations, one for the set P_X of e-places of Σ^0 and one for the set Q_X of x-places of Σ^0. Since transition **2** of the operator box, this time, does not have a constant plain box to be replaced by, there is no constant set of e-places (or x-places) to perform the refinement with. Rather we have to re-use P_X (and Q_X, respectively) in the right hand side of the tree equation. This leads directly to the following equations:

$$\begin{aligned} P_X &= 1 \lhd (\mathbf{1} \lhd (\{5\})) \,\cup\, 3 \lhd (\mathbf{2} \lhd (P_X)) \\ Q_X &= 2 \lhd (\mathbf{1} \lhd (\{6\})) \,\cup\, 4 \lhd (\mathbf{2} \lhd (Q_X)). \end{aligned} \tag{7}$$

There are two solutions for the first of these equations. One solution contains only finite trees and the other solution contains an infinite tree as well. We call these solutions P_X^{min} and P_X^{max}, respectively, and for reasons of brevity, we write the trees in these sets in the form of sequences of node labels:

$$\begin{aligned} P_X^{min} &= \{\mathit{1}\mathbf{1}\mathit{5}, \mathit{3}\mathbf{2}\mathit{1}\mathbf{1}\mathit{5}, \mathit{3}\mathbf{2}\mathit{3}\mathbf{2}\mathit{1}\mathbf{1}\mathit{5}, \ldots\} \\ P_X^{max} &= P_X^{min} \cup \{\mathit{3}\mathbf{2}\mathit{3}\mathbf{2}\mathit{3}\mathbf{2}\cdots\}, \end{aligned}$$

The equation for Q_X has two corresponding solutions. Together, they determine four starting boxes $\Sigma^{0,min}$ (using P_X^{min} and Q_X^{min}), $\Sigma^{0,max/min}$ (using P_X^{max} and Q_X^{min}), $\Sigma^{0,min/max}$ (using P_X^{min} and Q_X^{max}) and $\Sigma^{0,max}$ (using P_X^{max} and Q_X^{max}) to begin the approximation of the solution of the net equation (6) for $X = a\|X$. Depending on which starting point of the approximation is chosen, different solutions of equation (6) are obtained. Two of them (intuitively, the minimal and maximal ones) are shown in Figure 7 (but again we do not represent all name trees). It is not hard to check that both boxes are indeed solutions of equation (6) in the same strict sense as before. Note that the interleaving behaviour of the two solutions is the same, but if infinitely broad behaviours are allowed then the first one has a terminating behaviour while the second has not (since the initial token on its isolated e-place remains there and cannot be moved to the corresponding x-place). To end this section, we illustrate the technique on the three remaining examples defined in (1), albeit not in the same degree of detail as before. The fourth term equation $X = (a; X)$ corresponds to the net equation shown in Figure 8. This net equation further leads to the following tree equations:

$$\begin{aligned} P_X &= \mathit{1} \lhd (\mathbf{1} \lhd (\{\mathit{4}\})) \\ \text{and}\ \ Q_X &= \mathit{3} \lhd (\mathbf{2} \lhd (Q_X)). \end{aligned} \tag{9}$$

Note that the only solution for P_X is the singleton set $\{\mathit{1}\mathbf{1}\mathit{4}\}$ while Q_X has no unique solution, corresponding to the fact that the expression $X = (a; X)$ is front-guarded but not rear-guarded. The minimal solution for Q_X is the empty set $\emptyset$, but the singleton set $\{\mathit{3}\mathbf{2}\mathit{3}\mathbf{2}\mathit{3}\cdots\}$ containing an infinite tree is also a solution. As we start the approximation with a nonempty box Σ^0, only the latter solution for Q_X is permissible as the set of x-places of Σ^0. When begun with the one-e-place one x-place box just defined, the approximation leads to a solution whose structure is similar to the box shown in Figure 5, except that

[6] In general, an instance of (2) is called *guarded* iff its dependency graph contains no infinite path of the form $X^1X^2\ldots$ such that each X^{j+1} is refined in the equation for X^j into a transition which is connected to an e-place, or each X^{j+1} is refined in the equation for X^j into a transition which is connected to an x-place.

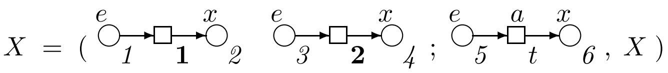
 (6)

Fig. 6. Equation associated to $X = (a \| X)$

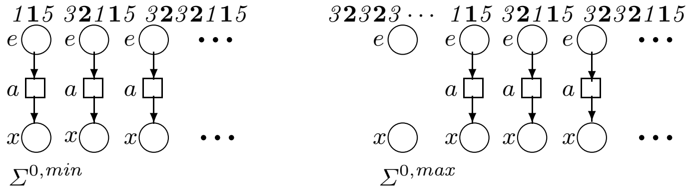


Fig. 7. Two solutions to $X = (a \| X)$

the lower line is replaced by a single x-place, namely the element of Q_X.

The fifth term equation $X = ((a \| b) + X)$ corresponds to the net equation shown in Figure 9. By the same rules as above, this leads to the following tree equations:

$$\begin{aligned} P_X &= \mathit{1} \lhd (\mathbf{1} \lhd (\{\mathit{3}\}), \mathbf{2} \lhd (P_X)) \\ &\quad \cup \mathit{1} \lhd (\mathbf{1} \lhd (\{\mathit{5}\}), \mathbf{2} \lhd (P_X)) \\ Q_X &= \mathit{2} \lhd (\mathbf{1} \lhd (\{\mathit{4}\}), \mathbf{2} \lhd (Q_X)) \\ &\quad \cup \mathit{2} \lhd (\mathbf{1} \lhd (\{\mathit{6}\}), \mathbf{2} \lhd (Q_X)). \end{aligned} \tag{11}$$

Note how the branching in the operator box leads to corresponding branchings in the trees. Equations (11) are interesting because, as it turns out, their largest solutions have uncountably many trees. Starting the approximation with such an uncountably large box leads to an uncountably large solution of the equation (10). Intuitively speaking, this solution corresponds to the infinite simultaneous refinement

$$(\ \mathbf{1} + \mathbf{2} + \mathbf{3} + \dots\ ;\ a \| b, a \| b, a \| b, \dots\ ).$$

It is described in [2, 6]. Notice why the set of places is necessarily uncountable: in the box $(\mathbf{1}; a \| b)$ there are two e-places and two x-places; in $(\mathbf{1} + \mathbf{2}; a \| b, a \| b)$ there are four e-places and four x-places; in $(\mathbf{1} + \mathbf{2} + \mathbf{3}; a \| b, a \| b, a \| b)$ there are eight e-places and eight x-places; and so on. This uncountability can be seen keeping in mind that the right hand sides of equation (11) are recursive unions of binary trees. Nevertheless, equation (11) also has countable solutions. For instance, a singleton set of trees whose only member is shown in Figure 10 (and which is a solution of the inequation on sets-of-trees corresponding to equation (11)), leads in the limit to a countable solution for P_X. Starting the approximation with a countable solution of (11), one gets a countably infinite solution of the equation (10). Intuitively speaking, this solution corresponds to the infinite choice

$$(a \| b) + (a \| b) + (a \| b) + \dots$$

whose box, being made of countably many finite alternatives, is countable[7].

[7] More precisely, this box can be described as follows. There are infinitely many pairs of transitions labelled a and b, respectively; let us call them a_1, b_1 (the first pair), a_2, b_2 (the second pair) and so on. There is a countable infinity of e-places. Split this set into two halves both of which are again countably infinite. The left half is taken as preplaces of a_1, the right half as preplaces of b_1. The left half is split again into two countable infinities, and so is the right half. The left left half and the left right half are taken as preplaces of a_2, the right left half and

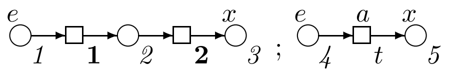
$X = ($  $, X\ )$ (8)

Fig. 8. Equation associated to $X = (a; X)$

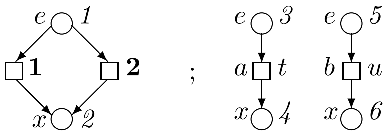
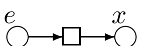
$X = ($  $, X\ )$ (10)

Fig. 9. Equation associated to $X = ((a \| b) + X)$

Finally, the sixth term equation $X = X$ leads to the following net equation:

$X = ($  $;\ X\ ).$ (12)

By the rules of the game, (12) leads to the following tree equations for the entry and exit places of the starting box:

$$\begin{aligned} P_X &= \mathit{1} \lhd (\mathbf{1} \lhd (P_X)) \\ \text{and}\ \ Q_X &= \mathit{2} \lhd (\mathbf{1} \lhd (Q_X)). \end{aligned} \tag{13}$$

We have $P_X = \emptyset = Q_X$ amongst the set of possible solutions, but we also have nonempty singleton solutions. In this case, the approximations never change anything, and the box solving $X = X$ has exactly one e-place, one x-place, and no transitions.

7 Main results

In this section we state the results that underpin the theory exemplified in the previous sections. Proofs can be found in [15]. We have the following theorems, (A)-(J), which we will phrase for the special case $n = 1$ of the equation system (2):

the right right half as preplaces of b_2, and so on like a binary tree of infinities. The same is done symmetrically with the x places.

(A) The right hand side of the equation system (2) determines an operator which is monotonic w.r.t. $\sqsubseteq$ relation.

(B) The right hand sides of the tree-set equations associated to a variable X determine an operator which is monotonic on the complete lattice of sets of trees with $\subseteq$. Hence every tree-set equation has a unique minimal solution and a unique maximal solution P_X^{min} and P_X^{max} (and similarly for Q_X).

(C) P_X^{min} is always a set of finite trees (which may be countably infinite, as in example 3, or empty, as in example 5).

(D) P_X^{max} is always a nonempty set. *Proof sketch:* We need to construct a tree τ which is a member of P_X^{max}. In general, if τ satisfies $\{\tau\} \subseteq f(\{\tau\})$, where f is the operator determined by the right hand side of the tree equation for P_X, then $\tau \in P_X^{max}$. We may construct such a tree by induction, this time on the domain of trees, starting with the root of an arbitrary set on the right hand side of the tree equation. Note that this equation is always of the form $P = \mathit{set} \cup \ldots \cup \mathit{set}$, so that the construction is well-defined. *End of proof sketch for* (D). Note that the tree shown in Figure 10 is an instance of this construction.

(E) Let P be any set of trees satisfying only the ($\subseteq$) part of the tree equations. Then

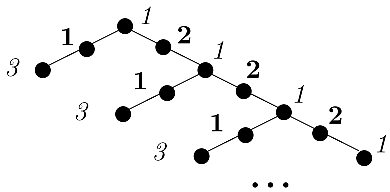


Fig. 10. Infinite tree belonging to a countable solution of equation (11)

$\bigcup_{i\geq 0} f^i(P)$ is a solution of the equations, where f is as in (D). Moreover, both sets lead to the same solution of (2).

(F) If P_X satisfies only the ($\subseteq$) part of the tree equations and if Σ^0 is a box constructed from P_X and a corresponding Q_X, then starting the approximation of (2) with Σ^0, an ascending (in the $\sqsubseteq$ sense) chain of boxes whose limit solves (2) is obtained.

(G) It is always possible to start approximating a solution of (2) with a single-entry single-exit box Σ^0. *Proof:* From the proof of (D) and (E).

(H) Every equation has a countable nonempty solution. *Proof:* From (D) and the fact that the limit construction preserves countability of nets.

(J) If the net equation (2) is guarded then the associated tree equations have unique solutions and the associated solution obtained by successive approximations is unique and the least one with respect to $\sqsubseteq$.

If instead of a single equation, we are given a finite set of n equations (2) then the theory can be generalised. Instead of considering the lattice of boxes, we then consider the extended lattice of n-fold Cartesian product of the set of boxes. If we are given a countably infinite system of equations, we consider the standard extension to the corresponding function space representing the infinite Cartesian product. All constructions and results remain the same (with changes produced by the extensions).

8 Petri net semantics of recursive expressions

In this section, we summarise the constructions defined in this paper in general terms. Let a closed recursive expression

$$\mu X_i\{X_i = E_i \mid i = 1, 2, \ldots\}$$

be given. As E_i is recursion free, it is possible to represent it as one or more (auxiliary) operator boxes, even if it is deeply nested. In fact, the three operator boxes shown in Figure 11, which correspond to the three connectives of the syntax for recursion free expressions, are sufficient for that purpose. To understand this translation, consider the expression

$$X = ((a\|(b + (X;b)));(X + (b\|X)))$$

as an example. It can be unravelled into six equations as follows:

$$\begin{array}{lll} X \ = X_1; X_2 & X_1 = a\|X_3 & X_2 = X + X_4 \\ X_3 = b + X_5 & X_4 = b\|X & X_5 = X; b. \end{array}$$

Every such equation can be translated directly into a net equation of the form $X = (\Omega; \Sigma_1, \Sigma_2)$, where Ω is one of the operator boxes of Figure 11[8]. In this way, a net equation system (2) can be associated in a canonical way to the given recursive expression. This

[8] For simplicity, we exclude (trivial) equations of the form $X = a$ and $X = X'$.

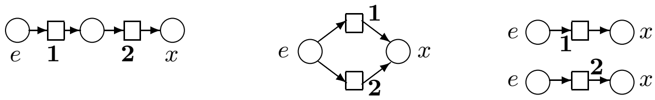


Fig. 11. Three operator boxes for ; (left), + (middle) and $\|$ (right)

system can be solved using the techniques and results described earlier. Note however that a box being a solution of the net equation does not necessarily mean that it is also a good semantics of the corresponding expression. Fortunately, as it is proved in [15] that any such limit has exactly the same behaviour as the expression it comes from.

Summarising, the following overall picture emerges from our discussions:

- Any term equation has an associated net equation, and any net equation has a solution.
- Any solution of a net equation can be derived as an approximation from some suitable starting box (namely, the box consisting only of the e-places and the x-places of the solution).
- Conversely, any starting box Σ^0 satisfying $\Sigma^0 \sqsubseteq \Sigma^1$ for the next approximation Σ^1 leads to a solution.
- If recursion is guarded then both the net equation and the tree equations derived from it have unique solutions.
- Otherwise, the equations may have different solutions (cf. example 4).
- In all cases, it is possible to start the approximation with a well chosen single entry / single exit box.
- There is always a countable solution.

The fact that for expressions such as $X = (a\|X)$, more than one solution exists may be unsatisfactory, and therefore it may make sense to examine the space of possible solutions for a canonical one. While the function defined by the net equation is monotonic, it is not a function on a complete partial order, and hence the notion of minimal or maximal fixpoints (which could lead to some canonicity) do not make sense. However, they do make sense on the auxiliary lattice of sets-of-trees on which the function derived from the associated tree equations is defined. Since the minimal solution may be empty, only the maximal solution offers itself as a candidate for being 'canonical' in all cases. One may therefore take the point of view that the box approximated from a starting box that corresponds to the maximal solution of the tree equation, is 'the' semantics of the given expression; it is unique, of course. For example 5, this would mean that the uncountable box indicated earlier - rather than the countable one - is 'the' semantics of the expression $X = ((a\|b) + X)$, and for example 3, it would mean that the box shown on the r.h.s of Figure 7, rather than the box on the l.h.s., is 'the' semantics of $X = (a\|X)$. This view is taken in [6]. Here, we have demonstrated that if the uniqueness of a net associated to an expression is stressed, this view (of taking the maximal fixpoint of the tree equation) appears to be the only reasonable choice because the minimal fixpoint is not always applicable.

9 Concluding remarks

We have demonstrated how a tree-based general naming discipline for refinement leads to a technique for solving recursive Petri net equations. This technique is orthogonal to the general relabelling technique described in [5] to capture synchronisation and other transition-based operators. These two techniques can be combined with each other to yield the net semantics of process algebras featuring, beside recursion and the other control operators, also synchronisation and restriction.

Acknowledgements

This work owes to Raymond Devillers who invented and formalised the tree naming device in a more general form than needed here (applicable also to nets with side conditions), and to Javier Esparza who did a formalisation for guardedly recursive expressions [7]. Both also gave useful comments on this paper. We are also indebted to Mogens Nielsen and Ugo Montanari for comments. This work was done within the Esprit Basic Research Working Group 6067 CALIBAN.

References

1. J.Baeten, W.P.Weijland: *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18 (1990).
2. E.Best, R.Devillers, J.Esparza: General Refinement and Recursion Operators for the Petri Box Calculus. Springer-Verlag, LNCS Vol.665, 130-140 (1993).
3. E.Best, R.Devillers, J.Hall: The Petri Box Calculus: a New Causal Algebra with Multilabel Communication. Advances in Petri Nets 1992, G.Rozenberg (ed.), Springer-Verlag, Lecture Notes in Computer Science Vol. 609, 21-69 (1992).
4. E.Best, C.Fernández: *Nonsequential Processes. A Petri Net View.* Springer-Verlag, EATCS Monographs on Theoretical Computer Science Vol. 13 (1988).
5. E.Best, M.Koutny: A Refined View of the Box Algebra. Proceedings of the Petri Net Conference'95, G.De Michelis, M.Diaz (eds.), Springer-Verlag, Lecture Notes in Computer Science (1995).
6. R.Devillers: S-invariant Analysis of Petri Boxes. Technical Report LIT-273, Laboratoire d'Informatique Théorique, Université Libre de Bruxelles (1993). To appear in Acta Informatica (1995).
7. J.Esparza: Fixpoint Definition of Recursion in the Box Algebra. Memorandum, Universität Hildesheim (December 1991).
8. U.Goltz: On Representing CCS Programs by Finite Petri Nets. Proc. MFCS'88, Springer-Verlag, Lecture Notes in Computer Science Vol. 324, 339-350 (1988).
9. U.Goltz, R.van Glabbeek: Refinement of Actions in Causality Based Models. Springer-Verlag, Lecture Notes in Computer Science Vol.430, 267-300 (1989).
10. M.B.Hennessy: Algebraic Theory of Processes. The MIT Press (1988).
11. C.A.R.Hoare: *Communicating Sequential Processes.* Prentice Hall (1985).
12. R.Janicki, M.Koutny: Structure of Concurrency. Theoretical Computer Science Vol. 112, 5-52 (1993).
13. R.Janicki, P.E.Lauer: *Specification and Analysis of Concurrent Systems - the COSY Approach.* Springer-Verlag, EATCS Monographs on Theoretical Computer Science (1992).
14. A.Kiehn: Petri Net Systems and their Closure Properties. Springer-Verlag LNCS Vol.424, 306-328 (1990).
15. M.Koutny and E.Best: Operational Semantics for the Box Algebra. Draft paper (1995).
16. K.Jensen: Coloured Petri Nets. Basic Concepts. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1992).
17. P.E.Lauer: Path Expressions as Petri Nets, or Petri Nets with Fewer Tears. Technical Report MRM/70, Computing Laboratory, University of Newcastle upon Tyne (1974).
18. R.Milner: *Communication and Concurrency.* Prentice Hall (1989).
19. E.R.Olderog: *Nets, Terms and Formulas.* Cambridge Tracts in Theoretical Computer Science 23 (1991).
20. G.Plotkin: Domains. Department of Computer Science, University of Edinburgh (1983).
21. W.Reisig: *Petri Nets. An Introduction.* EATCS Monographs on Theoretical Computer Science Vol. 3, Springer-Verlag (1985).
22. D.Taubner: *Finite Representation of CCS and TCSP Programs by Automata and Petri Nets.* Springer-Verlag, Lecture Notes in Computer Science Vol. 369 (1989).

This article was processed using the LaTeX macro package with LLNCS style