Operational and Denotational Semantics for the Box Algebra

Maciej Koutny* and Eike Best†

February 11, 1997

Abstract

This paper describes general theory underpinning the operational semantics and the denotational Petri net semantics of the box algebra including recursion. For the operational semantics, inductive rules for process expressions are given. For the net semantics, a general mechanism of refinement and relabelling is introduced, using which the connectives of the algebra are defined. The paper also 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. The consistency of the two semantics is demonstrated. The theory is generic for a wide class of algebraic operators and synchronisation schemes.

Keywords: Petri nets, process algebra, refinement, denotational semantics, operational semantics, recursion, bisimulation.

Contents

*Department of Computing Science, University of Newcastle, Newcastle upon Tyne NE1 7RU, United Kingdom, email: Maciej.Koutny@newcastle.ac.uk

†Institut für Informatik, Universität Hildesheim, Marienburger Platz 22, D-31141 Hildesheim, Germany, email: E.Best@informatik.uni-hildesheim.de

1 Introduction

Process algebras [1, 19, 20, 22, 28] are widely used as models for the specification of concurrent systems, and so are Petri nets and their extensions [6, 21, 23, 32]. In process algebra, emphasis tends to be on the study of algebraic properties of operators related to common constructs used in programming languages, such as selection and recursion. In Petri net theory, emphasis tends to be on the partial order semantics of a concurrent system, and on structural characterisation of its behavioural properties. In order to combine the two models and their particular advantages, it has often been proposed that a Petri net semantics be given to a process algebra [5, 17, 22, 26, 29, 33]. This paper has also originated from such an approach and as such belongs to the area of comparative study of the relationship between process algebras and Petri nets. It differs from the prior work on this subject in that it introduces very general mechanisms using which a wide range of operators on expressions and nets can be treated in a uniform and consistent way. The treatment covers the standard operators found in process algebras, including fully general recursion, as well as other, less common constructs. It can be argued that it is generic for a wide class of models combining process algebras and Petri nets. It is worth noting that there is a methodological novelty in the line of investigation followed in this paper. Typically, a research in this area starts with a given process algebra which has a fixed syntax and semantics of process terms. This is followed by an attempt to define, for each process algebra operator a corresponding operator on nets, and concluded with a proof of a behavioural equivalence of the two. In our approach, the order of the first two steps is reversed. That is, we first introduce a class of operators on Petri nets which guarantee a compositional behaviour of the resulting nets, and only after that we define a corresponding algebra of process expressions. Such a shift of emphasis is not purely a matter of taste. Basically, it allows one to make the whole approach generic since we can focus right from the start on those operators which can be given clear representation in the domain of Petri nets. After that, deriving a corresponding term algebra and transition rules poses little problem. The other direction could prove to be much more difficult to follow since we feel that deriving a process algebra operator from a well formed (generic) net operator is relatively easy, whereas deriving a net operator from a well formed (generic) term operator may be very complex.

1.1 Extending the box algebra

The work presented in this paper consolidates and extends the *box algebra* (or PBC, for Petri Box Calculus) which is a model of concurrent system combining process algebra and Petri nets.

The box algebra [4, 5, 25] is a variant of CCS [28], and has originally been introduced with the aim of supporting the compositional translation from high level concurrent programming languages into Petri nets. It was to serve two purposes, that of being a target of a translation from a concurrent language and that of being a source of a translation into Petri nets; both translations were required to be compositional. To achieve its first purpose, the PBC provides some modifications with respect to CCS, such as extending its synchronisation operation with a multisynchronisation capability[1] and separating concurrent composition from synchronisation. To achieve its second purpose, it has been provided with a compositional Petri net semantics in [4] and [5]. The PBC has also been given a net-independent operational semantics [25, 24] in the SOS (Structured Operational Semantics) style of [30]. One of the results of this paper is the demonstration of a full consistency of these two semantics.

The main purpose of the present paper is to show how one can, in a generic way, define an algebra of nets which then gives rise to a process algebra with the semantics given through the SOS rules, so that the latter is equivalent to the standard semantics of compositionally defined nets. The technical device underpinning our model is *net refinement* which is a net operation generalising transition refinement, renaming and synchronisation. The main advantage of net refinement lies in its ability to express a wide range of different net operations and, as a result, the main technical part of the paper is phrased in terms of a single generic net operator. Simultaneous refinement emerged from

[1] Not unlike that described in synchronous CCS [28], but retaining asynchronous concurrency.

an observation that the separation between concurrent composition and synchronisation in the box algebra can be generalised so that it leads to a division of the set of operations into two classes. The first class of operators, which includes choice and concurrent composition, affects control flow and thus the places of an associated net; the other class, with typical members being restriction and synchronisation, affects the observable activities and thus the transitions of an associated net. It also turns out that any member of the former class of operations can be captured formally by transition refinement in a Petri net, and any member of the latter by a sufficiently powerful notion of general relabelling (itself combining transition synchronisation with renaming and/or restriction). Essentially, transition refinement comprises one half of the original PBC operators – and more – as special cases, while general relabelling comprises the other half – and more – as special cases.

Armed with net refinement, we define in this paper a class of Petri nets – called *operator boxes* – which may serve as generalised (n-ary) operators in a net based process algebra, and another class of nets – called *plain boxes* – which may serve as basic semantic objects (associated, for instance, with parallel programs). In particular, operator nets can capture all familiar constructs found in process algebras. This allows us to form very general expressions; essentially, every n-ary expression corresponds to an operator box with n transitions, and the net associated with such an expression arises from simultaneously refining these n transitions by the n constituent nets of the expression. In this way, operator boxes can be seen as functions taking n plain boxes as input and yielding a plain box as output.

Having defined very general net operators, we show how one can define and solve arbitrary recursive definitions within the Petri net framework. The solution is based on a standard denotational approach of taking fixpoints of monotonic functions over the domain of plain boxes. We also show that the solution is consistent with the SOS semantics of recursive process algebra terms.

As an application of the generic approach, we show how it can be applied to the original box algebra, as well as to a fragment of CCS. We also indicate how the concurrent composition operators of TCSP and COSY can be dealt with.

Many of the results of this paper, as well as extensive motivation, are described informally in two tutorial-style papers [7, 8]. This work also builds on previous work by Raymond Devillers and Javier Esparza [4, 11, 12, 15]. There is a similarity (but not a complete agreement) between our way of representing markings algebraically and the proved transitions of [9].

1.2 Outline of the paper

In the first part of this paper – sections 2 and 3 – we develop criteria for expressions formed from Petri nets using net refinement to have an equivalent (i.e., sound and complete) operational SOS semantics. These criteria lead to certain restrictions on the structure and behaviour of the corresponding nets (this being the reason for calling them boxes), rather than with any concrete syntax. We will motivate these restrictions carefully, and as a result, we will be able to derive generic soundness and completeness theorems purely on the level of boxes.

In the second part of the paper – section 4 – we show that the set of boxes forms a reflexive domain in the sense that functions thereon, or on a Cartesian product thereof (as given by operator boxes), have fixpoints. This fact is due to the judicious definition of transition refinement we give in section 2, and it leads to a treatment of recursive expressions in denotational style [31]. The main result of section 4 states that every recursive equation has at least one solution. Uniqueness of solutions is not always guaranteed, but we will show that uniqueness can be proved for guarded equations.

That this theory is orthogonal to the sound and complete SOS semantics derived in section 3, and hence that they can be used in combination, will be shown in the third part of the paper – sections 5 and 6. In section 5 we will turn the net-based approach of section 3 into a concrete (albeit still very general and parametric) syntax which allows any operator box, as defined in section 3, to be viewed as a (process) algebraic operator. For expressions formed from such generalised operators, we define both a mapping to boxes (called their denotational semantics) and – taking advantage of the ground prepared in section 3 – a generic inference rule (called their operational semantics). As the main results of this section, we transfer the soundness and completeness results to this concrete syntax and SOS rule. We also define several variants yielding different semantics (steps in terms of transitions, steps in terms of labels, partial orders). Finally, in section 6, we restrict our attention to the original PBC defined in [5] and its structured operational (SOS) semantics defined in [25, 24], for which we essentially show that they are instances of the more general theory developed in the preceding sections, and that soundness and completeness of the latter with respect to the former follow directly from the general soundness and completeness results stated in section 5.

Almost all proofs have been moved to appendices.

2 Nets, Labels and Refinement

Throughout the paper we use the standard mathematical notation. In particular, multisets will be signified by the angle brackets, e.g., $\langle x, x, y\rangle$ is the multiset comprising two copies of x and one copy of y. The multiset sum, subtraction and multiplication by a natural number will respectively be denoted by '$+$', '$-$' and '$\cdot$'. $\mathsf{mult}(X)$ will denote the set of finite multisets over a set X. A function $f : X \to Y$ can also be represented as the set $\{(x, f(x)) \mid x \in X\}$. For a finite subset $X' = \{x_1, \ldots, x_k\}$ of X, we will use $f\langle X'\rangle$ to denote the multiset $\langle f(x_1), \ldots, f(x_k)\rangle$. Finally, $\uplus$ will denote the disjoint set union.

We shall assume the reader has at least an elementary familiarity with the Petri net theory, like that expounded in [32]. However, in order to make the presentation self-contained, all the relevant concepts and definitions will be introduced. We begin by defining labelled nets, in section 2.1, and net refinement, in section 2.2.

2.1 Labelled nets

2.1.1 Relabellings

We assume an infinite set Lab of *labels* to be given. The intuition behind an element $a \in \mathsf{Lab}$ is that it denotes some interface activity. The term 'label' has been chosen because the elements of Lab will serve as transition labels. Later – in section 6.1 – when we deal with concrete instances of the general model, some additional structure will be imposed on Lab. A *relabelling* over Lab is any relation

$$\rho \subseteq \mathsf{mult}(\mathsf{Lab}) \times \mathsf{Lab}$$

The set of all relabellings over Lab is denoted by Relab. The intuition behind a pair $(\alpha, a) \in \rho$ is that it specifies some interface change in a concurrent system. In the literature, there are three types of commonly employed interface changes, all of which can be captured by suitably chosen ρ:

> *Synchronisation.* Whereby one wants to ensure that a set of n transitions labelled $a_1, \ldots, a_n$ can be synchronised to yield another transition labelled a. This can be modelled by a pair $(\langle a_1, \ldots, a_n\rangle, a) \in \rho$. We may well have $(\langle a_1, \ldots, a_n\rangle, a) \in \rho$ and $(\langle a_1, \ldots, a_n\rangle, b) \in \rho$ for $a \neq b$, i.e., ρ does not have to be a function. Note that since we do not exclude the possibility that some of the a_i's are the same, multisets rather than sets of labels are used in the definition of ρ.
>
> *Basic relabelling.* Where one wishes to replace the label of each transition labelled a with another label, b. We can achieve this by including the pair $(\langle a\rangle, b)$ in ρ and stipulating that no other pair of the form $(\langle a\rangle, x)$ belongs to ρ.
>
> *Restriction.* If one wants to remove all transitions with a given label a then it suffices to ensure that no pair of the form $(\langle a\rangle, x)$ is in ρ.

Two specific relabellings are of particular importance. The identity relabelling,

$$\rho_{id} = \{(\langle a \rangle, a) \mid a \in \mathsf{Lab}\}$$

captures the 'nothing changes' interface non-change. The constant relabelling,

$$\rho_a = \{(\langle b \rangle, a) \mid b \in \mathsf{Lab}\},$$

where a is a label in Lab, can be identified with a itself. Thus we consider the set of labels Lab to be embedded in the set of relabellings Relab.

2.1.2 Nets and markings

The class of Petri nets we chose to work with are what we call labelled nets. Not surprisingly, such a general term has several different meanings in the Petri net theory. In this paper, by a *labelled net* we will mean a tuple

$$\Sigma = (S, T, W, \lambda, M)$$

such that: S and T are disjoint sets of respectively *places* and *transitions*; W is a *weight function* from the set $(S \times T) \cup (T \times S)$ to the set of natural numbers $\mathbf{N}$; λ is a *labelling* function for places and transitions such that $\lambda(s) \in \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$, for every place $s \in S$, and $\lambda(t) \in \mathsf{Relab}$ is a relabelling, for every transition $t \in T$; and M is a *marking* which is a mapping assigning a natural number to each place $s \in S$. We adopt the standard rules about representing nets as directed graphs, viz. places are represented as circles, transitions as rectangles, the weight function is indicated by arcs, and markings are shown by placing tokens inside circles. To avoid ambiguity, we will sometime decorate the various components of Σ with the index Σ. Figure 1 shows the graph of a labelled net Σ such that:

$$\begin{aligned}
S &= \{p, q, r, s\} \\
T &= \{t, u, w\} \\
\lambda &= \{(p, \mathsf{e}), (q, \mathsf{i}), (r, \mathsf{x}), (s, \mathsf{e}), (t, a), (u, b), (w, a)\} \\
M &= \{(p, 1), (q, 0), (r, 0), (s, 1)\}
\end{aligned}$$

and the weight function W satisfies $W(S \times T \cup T \times S) = \{0, 1\}$ and

$$W^{-1}(\{1\}) \;=\; \{(p, t), (t, q), (q, u), (u, r), (s, w), (w, s)\}.$$

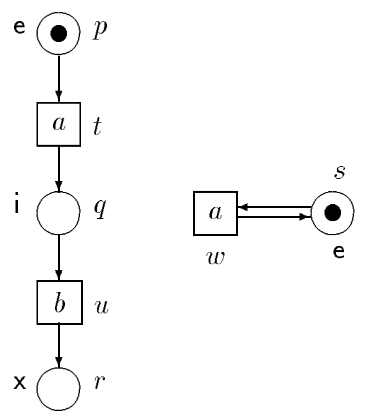


Figure 1: A labelled net Σ.

2.1.3 Related notions and notations

If the labelling of a place s in a labelled net Σ is e then s is an *entry* place, if i then s is an *internal* place, and if x then s is an *exit* place. By convention, ${}^{\circ}\Sigma$ and Σ° denote, respectively, the entry and exit places of Σ. For every place (transition) x, we use ${}^{\bullet}x$ to denote its pre-set, i.e., the set of all transitions (places) y such that there is an arc from y to x, that is, $W(y,x) > 0$. The post-set $x^{\bullet}$ is defined in a similar way. The pre- and post-set notation extends in the usual way to sets R of places and transitions, e.g., ${}^{\bullet}R = \bigcup\{{}^{\bullet}r \mid r \in R\}$. In what follows, all nets are assumed to be T*−restricted*, i.e., the pre-set and post-set of each transition are nonempty. Notice, however, that no assumption of that kind is made for places, i.e., a place s can be *isolated* which means that ${}^{\bullet}s = \emptyset = s^{\bullet}$. For the labelled net of figure 1 we have ${}^{\circ}\Sigma_0 = \{p, s\}$, $\Sigma_0^{\circ} = \{r\}$, ${}^{\bullet}p = \emptyset$, $p^{\bullet} = \{t\}$ and $\{p, q\}^{\bullet} = \{t, u\} = {}^{\bullet}\{q, r\}$.

The labelled net Σ is *finite* if both S and T are, *simple* if W always returns 0 or 1, and *pure* if for all transitions $t \in T$, ${}^{\bullet}t \cap t^{\bullet} = \emptyset$. The net in figure 1 is finite and simple, but is not pure.

A set $U \subseteq T$ is called a set of *independent* transitions if for all distinct $t, u \in U$, $({}^{\bullet}t \cup t^{\bullet}) \cap ({}^{\bullet}u \cup u^{\bullet}) = \emptyset$, and for all $t \in U$ and $s \in S$, $W(t,s) \leq 1$ and $W(s,t) \leq 1$. Ind_{Σ} will denote the set of all finite sets of independent transitions. For the net of figure 1, we have $\mathsf{Ind}_{\Sigma} = \{\emptyset, \{t\}, \{u\}, \{w\}, \{t, w\}, \{u, w\}\}$.

The marking M of Σ is *safe* (or more precisely 1-safe) if $M(S) \subseteq \{0, 1\}$. We will call ${}^{\circ}\Sigma$ the *initial marking* of Σ, and Σ° the *terminal marking*. A safe marking can and will be identified with the set of places to which it assigns 1. A safe marking is *clean* if it is not a proper superset of ${}^{\circ}\Sigma$ nor Σ°. The marking of the net in figure 1 is both safe and clean. It would cease to be clean if we added a token to it.

We will use three explicit ways of changing the marking of Σ, viz. we define $\lfloor\Sigma\rfloor$, $\overline{\Sigma}$ and $\underline{\Sigma}$ as, respectively, $(S, T, W, \lambda, \emptyset)$, $(S, T, W, \lambda, {}^{\circ}\Sigma)$ and $(S, T, W, \lambda, \Sigma^{\circ})$. Note that $\lfloor.\rfloor$, $\overline{(.)}$ and $\underline{(.)}$ are syntactic operations having nothing to do with derivability in the sense of the step sequence semantics defined next. The $\lfloor.\rfloor$ operator will also be applied elementwise to tuples of nets.

2.1.4 Step sequences

We adopt the standard finite step sequence semantics for a labelled net Σ. A *finite* set of transitions (a step) U is *enabled* by Σ if for every place $s \in S$,

$$M(s) \geq \sum_{t \in U} W(s,t).$$

We denote this by $\Sigma\,[U\rangle$ or $M\,[U\rangle$ (meaning: step U is enabled). An enabled step U can be *executed* leading to a follower marking M' defined, for every place $s \in S$, by

$$M'(s) = M(s) + \sum_{t \in U}(W(t,s) - W(s,t)).$$

Depending on the context, we will denote this by $M\,[U\rangle\,M'$ (meaning: marking M can be transformed through step U into marking M') or $\Sigma\,[U\rangle\,\Theta$ (meaning: net Σ can be transformed through step U into net Θ), where Θ is the labelled net (S, T, W, λ, M'). Transition labelling may be extended to steps, by taking the multiset of labels $\lambda\langle U\rangle$ instead of U. Notice that the label of a step (itself a set of transitions) needs to be a multiset rather than a set since different transitions may have the same label. It is possible to define steps even more generally, as multisets of concurrently enabled transitions (see [3]), but in the 1-safe context we shall follow in this paper, this adds nothing to the theory. Although we use the same term 'step' to refer to both a finite set of transitions and a finite multiset of labels, it will always be clear from the context which one is meant.

A *finite step sequence* of Σ is any finite (possibly empty) sequence $\sigma = U_1 \ldots U_k$ of finite sets of transitions for which there are labelled nets $\Sigma_0, \ldots, \Sigma_k$ such that $\Sigma = \Sigma_0$ and for every $1 \leq i \leq k$, $\Sigma_{i-1}\,[U_i\rangle\,\Sigma_i$. Depending on the context, this shall be denoted by one of the following notations:

$$\Sigma\,[\sigma\rangle\,\Sigma_k \quad , \quad M_{\Sigma}\,[\sigma\rangle\,M_{\Sigma_k} \quad , \quad \Sigma_k \in [\Sigma\rangle \quad \text{and} \quad M_{\Sigma_k} \in [M_{\Sigma}\rangle\ .$$

The marking M_{Σ_k} will be called *reachable* from M_Σ, and Σ_k *derivable* from Σ.

A step is another word for *concurrent occurrence of transitions*. Consider once again the labelled net shown in figure 1. Note that transitions t and w are enabled concurrently and, hence, $\{t, w\}$ is an enabled step of the net. After this step has been executed, transitions u and w are enabled concurrently and, hence, $\{t, w\}\{u, w\}$ is a step sequence of the net. A step does not need to be maximal. Thus, for instance, $\{t\}$ is also a step of the net, and so are $\{t\}\{u\}\{w\}\{w\}$ and $\{t, w\}\{u\}\{w\}\emptyset\emptyset$. In terms of labelled steps, the step sequence $\{t, w\}\{u\}\{w\}$ corresponds to $\langle a, a\rangle\langle b\rangle\langle a\rangle$. Notice that two different step sequences may correspond to the same labelled step sequence. For example, both $\{t\}\{w\}\emptyset\{u\}$ and $\{w\}\{t\}\emptyset\{u\}$ correspond to $\langle a\rangle\langle a\rangle\emptyset\langle b\rangle$.

2.2 Net refinement

In this section we describe a general mechanism by which transition refinement and interface change defined by a relabelling are combined together. Both operations are defined for a *finite*, *simple* and *pure* labelled net *without* isolated places, and with the *empty* marking,

$$\Omega = (S, T, W, \lambda, \emptyset)$$

which serves as a pattern for gluing together a tuple of nets denoted by $\mathbf{\Sigma}$ (one for every transition in Ω) along their entry and exit interfaces. The relabellings annotating the transitions of Ω specify the interface changes to which the nets in $\mathbf{\Sigma}$ are subjected. We assume that $T = \{v_1, \ldots, v_n\}$ $(n \geq 1)$ is an explicitly named and implicitly ordered set of n transitions which is used to index the tuple of nets, i.e.,

$$\mathbf{\Sigma} = (\Sigma_{v_1}, \ldots, \Sigma_{v_n})$$

(or, for short, $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$). We call $\mathbf{\Sigma}$ an Ω*–tuple* (of nets[2]), and for every set $Q \subseteq T$ we write

$$\mathbf{\Sigma}_Q = \{\Sigma_v \mid v \in Q\}.$$

Sometimes, $\mathbf{\Sigma}$ will be treated as the set $\mathbf{\Sigma}_T$. Finally, we assume that, for all $v \in T$, ${}^\circ\Sigma_v \neq \emptyset \neq \Sigma_v^\circ$ and

$$\Sigma_v = (S_v, T_v, W_v, \lambda_v, M_v).$$

2.2.1 Intuition behind net refinement

In the approach we follow, net refinement is carried out by taking Ω and $\mathbf{\Sigma}$ and creating a new labelled net $\mathsf{op}_\Omega(\mathbf{\Sigma})$ by first applying to every Σ_v of $\mathbf{\Sigma}$ an interface change specified by the relabelling $\lambda(v)$ of the corresponding transition v, and then using the result to refine each transition v of Ω. Although the construction of $\mathsf{op}_\Omega(\mathbf{\Sigma})$ will be expressed through a single definition, in section 2.2.5, we found it easier to introduce it by describing separately the interface change operation (phase 1 below) and transition refinement (phase 2).

Phase 1 – interface change

Here, relabellings $\rho_v = \lambda(v)$ are applied to the nets Σ_v. As a result, we obtain a tuple of n intermediate nets, $\mathbf{\Theta} = (\Theta_1, \ldots, \Theta_n)$, which are subsequently fed into the second phase, transition refinement. Applying relabelling ρ_v to net Σ_v affects only its transitions (and so indirectly the weight function). Basically, all the places together with their markings and labels are retained, but the transition set is redefined completely, in the following way.

Suppose t, u and w are transitions in Σ_v such that the multiset $\langle\lambda_v(t), \lambda_v(u), \lambda_v(w)\rangle$ belongs to the domain of the relabelling ρ_v, for example,

$$(\langle\lambda_v(t), \lambda_v(u), \lambda_v(w)\rangle, a) \in \rho_v.$$

[2]We will later use Ω–tuples of other types, e.g., Ω–tuples of variables.

This is interpreted as signifying that transitions t, u and w can be synchronised to yield a complex transition labelled a. The connectivity of the new transition is inherited from the connectivities of transitions t, u and w, that is, it simply acquires all (i.e., the sum of) the arc weights leading to and from the three transitions. The original transitions are not carried forward to Θ_v.

Taking as an example a specific pair of nets shown in figure 2 where t, y, u and w are transition names and a, b, c and d are transition labels, and two relabellings,

$$\begin{array}{rrcll} & \rho_1 & = & (\rho_{id} \setminus \{(\langle b\rangle, b)\}) \cup \{(\langle b\rangle, f)\} & \text{(used for } \Sigma_1) \\ \text{and} & \rho_2 & = & (\rho_{id} \setminus \{(\langle d\rangle, d)\}) \cup \{(\langle a, d\rangle, g)\} & \text{(used for } \Sigma_2) \end{array}$$

we obtain the two nets shown in figure 3 (where place and transition names are omitted).

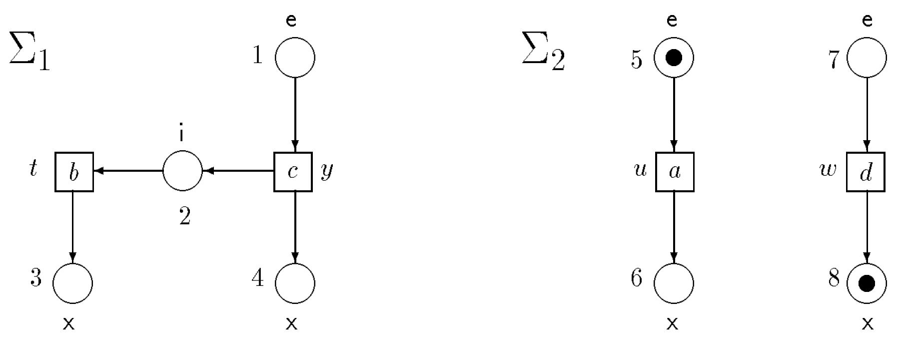


Figure 2: Two refining nets Σ_1 and Σ_2.

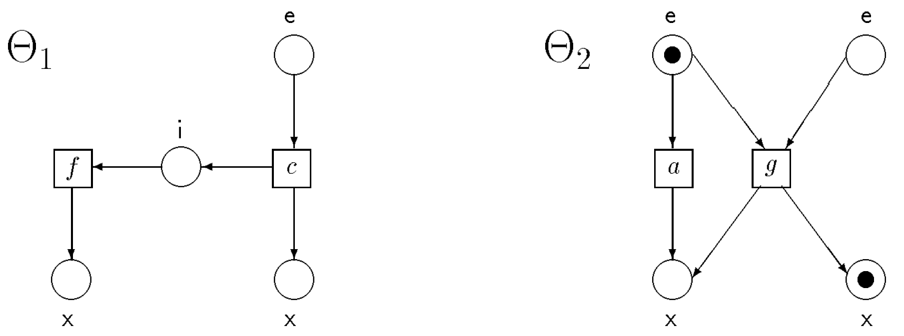


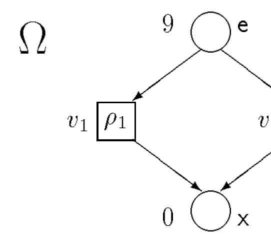
Figure 3: Interface change using ρ_1 (yielding Θ_1 from Σ_1) and ρ_2 (yielding Θ_2 from Σ_2).

In Θ_1, the transition labelled f (exemplifying basic transition relabelling) has been created by transition t and the pair $(\langle b\rangle, f) \in \rho_1$, and the transition labelled c has been created by transition y and the pair $(\langle c\rangle, c) \in \rho_1$. In Θ_2, the transition labelled a has been created by transition u and the pair $(\langle a\rangle, a) \in \rho_2$. The transition labelled g in Θ_2 (exemplifying transition synchronisation) has been created by the two transitions u and w of Σ_2 and the pair $(\langle a, d\rangle, g) \in \rho_2$. No transition was created using only transition w since no pair of the form $(\langle d\rangle, x)$ could be found in ρ_2, and thus ρ_2 specifies restriction on label d. Note that the very general definition of relabelling allows one to capture in a single definition several diverse interface changing operations.

Phase 2 – transition refinement

We now use the n-tuple $\mathbf{\Theta}$ from phase 1 and Ω to construct the final net, $\mathsf{op}_\Omega(\mathbf{\Sigma})$. This time, the exit and entry places of nets in $\boldsymbol{\Theta}$, rather than transitions, are joined together using the connectivity of the corresponding transitions in Ω. The rule for forming new places is described thus.

Suppose that s is a place in Ω with one transition in its pre-set, v_1, and two transitions in its post-set, v_2 and v_3. Then, for all possible combinations of three places, s_1, s_2 and s_3, such that s_1 is an exit place of Θ_1, and s_2 and s_3 are entry places of, respectively, Θ_2 and Θ_3, we construct a new place. The connectivity and marking of the new place are inherited from the places s_1, s_2 and s_3, while its label is the same as that of s. The places s, s_1, s_2 and s_3 are not carried forward to the final net. However, all the transitions and the internal places of nets $\boldsymbol{\Theta}$, together with the arcs between them, are carried forward to the final net. Continuing our example, and taking:



we obtain, by applying phase 2 construction to $\boldsymbol{\Theta} = (\Theta_1, \Theta_2)$, the net displayed in figure 4. For example, the only marked entry place in $\mathsf{op}_\Omega(\Sigma)$ results from combining the only marked entry place of Θ_2 with the only entry place of Θ_1 on account of place 9 of Ω.

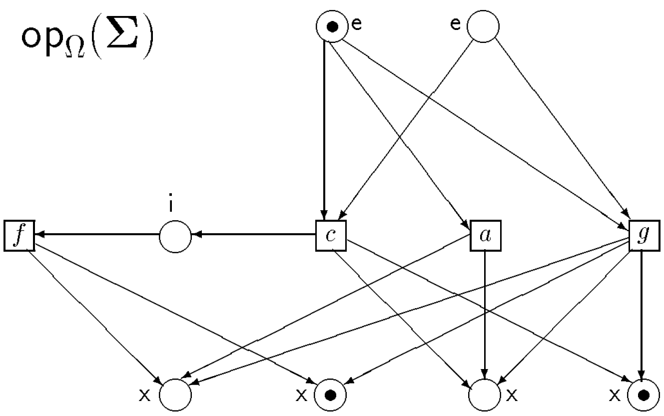


Figure 4: Refinement of the transitions of Ω by the nets of figure 3.

The formal definition of $\mathsf{op}_\Omega(\Sigma)$ given in section 2.2.5 below is not split into two distinct phases. Rather, these are combined together (with very little effort, since the interface change and transition refinement are orthogonal operations), into a single operation called *net refinement*.

2.2.2 A tree view of refinement

As far as net refinement as such is concerned, the names (identities) of newly constructed transitions and places are irrelevant, provided that we always choose them fresh. However, in our approach to solving recursive definitions on boxes – in section 4 – it is the *names* of places and transitions which play a crucial role since we use them to define the inclusion order on the domain of labelled nets. A key to our construction of recursive nets is the use of labelled trees as place and transition names.[3]

[3] We define such trees up to isomorphism.

This essentially allows the systematic recording of the heritage of places and transitions in the net resulting from net refinement.

First of all, we shall assume that there are two disjoint infinite pools of primitive place and transition names, $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}$. Each $\tau \in \mathsf{P}_{\mathsf{root}} \cup \mathsf{T}_{\mathsf{root}}$ can be viewed as a special tree with a single root labelled with τ which is also a leaf. Moreover, we shall use more complex trees as transition and place names, and use a linear notation to express those trees. To this end, the expression $x \lhd \mathcal{Q}$, where $x \in \mathsf{P}_{\mathsf{root}} \cup \mathsf{T}_{\mathsf{root}} \cup (\mathsf{T}_{\mathsf{root}} \times \mathsf{Lab})$ and $\mathcal{Q}$ is a nonempty set of trees, is a new tree where the trees of the set $\mathcal{Q}$ are appended to the root node labelled by x. If $\mathcal{Q} = \{q\}$ is a singleton set then $x \lhd \mathcal{Q}$ will simply be denoted by $x \lhd q$. For example, the tree in the right bottom corner of figure 5 can be written down as $0 \lhd \{v_1 \lhd 4, v_2 \lhd 8\}$.

We shall further assume that in nets like Ω, all places and transitions are primitive names (i.e., single root trees) from respectively $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}$. In other nets, such as those in $\mathbf{\Sigma}$, the trees used as names can be more complex. Each transition tree is a *finite* tree labelled by the elements of $\mathsf{T}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}\times\mathsf{Lab}$, and each place tree is a possibly infinite tree labelled by the primitive names from $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}$, which has the form

$$t_1 \lhd \cdots \lhd t_n \lhd \tau$$

where $t_1, \ldots, t_n \in \mathsf{T}_{\mathsf{root}}$, $n \geq 0$, and τ is a tree with the root node being labelled by a primitive place name in $\mathsf{P}_{\mathsf{root}}$. Thus no confusion will be possible between transition-trees and place-trees; the latter always have a label from $\mathsf{P}_{\mathsf{root}}$ and the former never (it is assumed that $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}} \times \mathsf{Lab}$ are disjoint sets). We comprise all these trees (including $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}$) in our sets of allowed transition and place names, denoted respectively by $\mathsf{P}_{\mathsf{tree}}$ and $\mathsf{T}_{\mathsf{tree}}$. The definition of net refinement will be done in such a way that, provided all names in Ω belong to $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}$, and all names in $\mathbf{\Sigma}$ belong to $\mathsf{P}_{\mathsf{tree}}$ and $\mathsf{T}_{\mathsf{tree}}$, then all names in $\mathsf{op}_{\Omega}(\mathbf{\Sigma})$ belong to $\mathsf{P}_{\mathsf{tree}}$ and $\mathsf{T}_{\mathsf{tree}}$, too.

2.2.3 Example of net refinement continued

The construction of the new transitions and places can now be redone in terms of name trees and operations on trees. One other explanation needs to be added at this point. There is no guarantee that the places of the different nets in $\mathbf{\Sigma}$ are disjoint. Indeed, we shall impose no restrictions on the names of the places and transitions of nets in $\mathbf{\Sigma}$. It may even happen that exactly *the same net*, with the same names and the same labels, appears several times in $\mathbf{\Sigma}$. This might lead to some undesirable identifications if we were to carry them through to phase 2 unchanged. So, although in phase 1 there is no real manipulation on the places of nets $\mathbf{\Sigma}$, when constructing the intermediate tuple $\mathbf{\Theta}$, we will simply change the name of each place q of Σ_v to $v \lhd q$, effectively tagging it with the corresponding transition to be refined. As a result, the place sets of different Θ_v's will be disjoint.

We now can present an exact representation of the net $\mathsf{op}_{\Omega}(\mathbf{\Sigma})$ which served as our running example, by explicitly showing all transition and place names; see figure 5. For instance, the tree name of the only marked e-place means that this place is the result of combining places 1 (through the refinement of transition v_1) and 5 (through the refinement of transition v_2) on account of place 9 of Ω. The name of the only i-place means that this place has been carried from the refining net into the refined net via transition v_1. The name of the transition labelled f means that this transition has been created from t by refining v_1 and has the label f. The name of the g-labelled transition means that it has been created from two transitions, u and w, by refining v_2 and has the label g.

The label of a new transition is contained in the tree (more precisely, it is a part of the label of it root) since our definition of relabelling allows the situation that the same set of transitions can synchronise in more than one way, yielding possibly different results. For example, if we redefined ρ_2 as

$$\rho_2 = (\rho_{id} \setminus \{(\langle d\rangle, d)\}) \cup \{(\langle a, d\rangle, g), (\langle a, d\rangle, h)\}$$

then in figures 4 and 5 another transition would be present, labelled h and having the same pre- and post- places as that labelled g. To distinguish between these two transitions, we include their labels in the name trees.

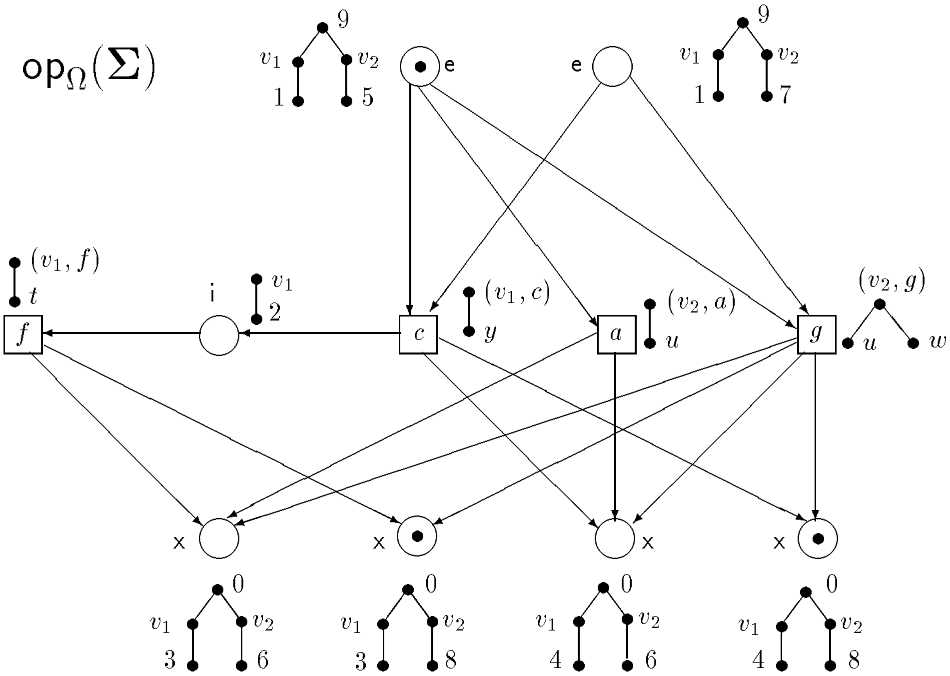


Figure 5: Place names and transition names of the net of figure 4.

2.2.4 Why this notion of refinement?

To begin with, one might ask the question why only sets of transitions are allowed to synchronise, and not multisets. The answer is that such an extension would not increase the descriptive power of the model, since later in this paper we only deal with nets which are safe and for whom refinement always yields safe nets. As a result, no transition obtained through the multiset synchronisation would ever be enabled. However, such a synchronisation would need to be carefully considered if we were to extend the model by allowing non-safe nets. There is also a related subtle point concerning the arc weights. One might observe that, by safeness, no transition which is joined with a place by an arc weight greater than one can ever be enabled. Hence, one could be tempted to postulate that all nets should be simple (i.e., only weights of 0 and 1 be allowed). This could be achieved by redefining the synchronisation in phase 1 above in such a way that only independent sets of transitions be allowed to combine to yield new transitions. The resulting model would be simpler and all the results in sections 3 and 3.3 would go through without any change. However, with such a definition of synchronisation, in section 4, net refinement would fail to be a monotonic mapping, rendering our treatment of recursion there no longer valid.

The notion of net refinement described here is very similar to those usually found in the literature (e.g., [18]). The tree view of refinement, which at this point may seem ad-hoc or overly complicated to the reader, leads to a number of interesting and useful results.

- As was mentioned already, it is amenable to defining recursive nets using an inclusion order. We will describe this in detail later on.
- Consider a more general setting when only some of the transitions in Ω are refined. This raises the question whether or not the result of successive refinements depends on the order in which they are done; using the tree names it is almost immediate to see that it does not [4, 12, 14]. Indeed, this had been the original motivation for introducing the tree view of refinement.

- Consider the case that Ω is non-simple and/or non-pure. As it is more fully explained in [12, 14], the tree naming device allows one to define transition refinement in this case as well. However, in the following, we will continue to restrict ourselves to pure and simple Ω's and to the simultaneous refinement of all transitions of Ω.

2.2.5 Formal definition of net refinement

Recall that $\Omega = (S, T, W, \lambda, \emptyset)$ and $\Sigma_v = (S_v, T_v, W_v, \lambda_v, M_v)$, for all $v \in T$. Under the assumptions made earlier about Ω and $\mathbf{\Sigma}$, the result of a simultaneous substitution of nets $\mathbf{\Sigma}$ for the transitions in Ω is a labelled net:

$$\mathsf{op}_\Omega(\mathbf{\Sigma}) \quad = \quad (S_0, T_0, W_0, \lambda_0, M_0)$$

whose components are defined as follows.

Places, their labels and markings

The set of places S_0 is defined as the union

$$S_0 \quad = \quad \bigcup_{v \in T} \mathsf{ST}^v_{\mathsf{new}} \cup \bigcup_{s \in S} \mathsf{SP}^s_{\mathsf{new}}$$

where the sets $\mathsf{ST}^v_{\mathsf{new}}$ and $\mathsf{SP}^s_{\mathsf{new}}$ are defined thus:

(a) For every $v \in T$, $\mathsf{ST}^v_{\mathsf{new}}$ comprises all places $p = v \lhd q$ such that q is an internal place in Σ_v. The label of p is i (for 'internal') and its marking is $M_v(q)$.

(b) Let $s \in S$ be a place such that $v_1, \ldots, v_k$ are the transitions forming its pre-set, and $v_{k+1}, \ldots, v_{k+m}$ are the transitions forming its post-set. Then $\mathsf{SP}^s_{\mathsf{new}}$ comprises all places

$$p = s \lhd \{v_1 \lhd q_1, \ldots, v_k \lhd q_k, v_{k+1} \lhd q_{k+1}, \ldots, v_{k+m} \lhd q_{k+m}\}$$

such that q_i is an exit place in Σ_{v_i}, for $1 \leq i \leq k$, and q_i is an entry place in Σ_{v_i}, for $k < i \leq k+m$. The label of p is $\lambda(s)$ and its marking is $M_{v_1}(q_1) + \cdots + M_{v_{k+m}}(q_{k+m})$.

Transitions and their labels

The set of transitions T_0 is defined as the union

$$T_0 \quad = \quad \bigcup_{v \in T} \mathsf{T}^v_{\mathsf{new}}$$

where each set $\mathsf{T}^v_{\mathsf{new}}$ comprises all transitions $u = (v, a) \lhd Q$ such that $Q \subseteq T_v$ is a nonempty set and $(\lambda_v\langle Q\rangle, a) \in \lambda(v)$. The label of u is a, and we denote $\mathsf{trees}(u) = Q$ (that is, the function $\mathsf{trees}(.)$ 'remembers' the transitions from which a new transition has been constructed).

Weight function

Let $u = (v, a) \lhd Q$ be a transition in $\mathsf{T}^v_{\mathsf{new}}$ (i.e., $Q = \mathsf{trees}(u)$). The weight function W_0 may return a non-zero value for such a u in only two cases:

(c) $p \in \mathsf{ST}^v_{\mathsf{new}}$ is as in (a) above (note: it is the same v in $\mathsf{T}^v_{\mathsf{new}}$ and $\mathsf{ST}^v_{\mathsf{new}}$). Then we define:

$$W_0(p, u) \quad = \quad \sum_{t \in Q} W_v(q, t)$$

$$\text{and} \quad W_0(u, p) \quad = \quad \sum_{t \in Q} W_v(t, q),$$

(d) $p \in \mathsf{SP}^s_{\mathsf{new}}$ is as in (b) above and $v \in {}^\bullet s \cup s^\bullet$. Then we define:

$$W_0(p, u) \quad = \quad \sum_{v_j = v,\, t \in Q} W_v(q_j, t)$$

$$\text{and} \quad W_0(u, p) \quad = \quad \sum_{v_j = v,\, t \in Q} W_v(t, q_j).$$

It should be noted that the above definition could be lifted to isomorphism classes of labelled nets. (The same applies to other definitions introduced so far.) However, we will not be going along this route. One reason has already been mentioned (we use concrete names of transitions and places to solve recursive definitions). The other is that the operational semantics for compositionally defined nets in section 3.3 will directly access the names of the transitions involved, as it is based on the one-step relation $[U\rangle$. It is not clear whether this technique can be lifted to the level of isomorphism classes of nets.

3 Boxes and Compositionality

In section 3.1 we delineate two classes of Petri nets, which are collectively referred to as boxes. They serve different purposes, and hence they have different properties. In section 3.2 we review and discuss these properties with hindsight. Finally, in section 3.3, we define SOS-style semantics for composable boxes. Essentially, we formalise the rule that 'a composed net can make a step if and only if its parts can make this step', and we show that under some assumptions (in particular, that all boxes enjoy the properties postulated for them), this rule is a provable theorem, with 'soundness' and 'completeness' being the two directions of this equivalence.

3.1 Boxes

All interfaces (i.e., composition points) of boxes are expressed by labellings of places and transitions. In section 3.1.1, we define plain boxes which serve as basic semantic objects of interest. In section 3.1.4, we define operator boxes, but in order to motivate their somewhat involved definition, we discuss the properties to be enjoyed by operator boxes in sections 3.1.2 and 3.1.3. Operator boxes serve as functions capable of creating new plain boxes from tuples of plain boxes. Section 3.2 contains a further discussion of the properties imposed on boxes and shows what can happen if they fail to be satisfied.

3.1.1 Plain boxes

A *box* is a labelled net Σ such that there is at least one entry and at least one exit place (Σ is *ex–restricted*) and, moreover, its entry places are free from incoming arcs and its exit places are free from outgoing arcs (Σ is *ex–directed*). A box is ex if its set of i-places and its set of transitions (and hence also its set of arcs) are empty.

We require Σ to be ex–restricted to ensure that it can be composed with other boxes in such a way that the composed object still does possess intended properties. We will see later that, and how, the theory depends on this assumption, cf. sections 3.2 and 4.5.5. The reason for requiring ex–directedness is more subtle and is motivated by our desire to obtain certain natural (and desirable) behavioural properties of compositionally defined boxes. We return to this issue in section 3.2.

There are two main classes of boxes which we will be interested in, namely *plain* boxes and *operator* boxes. The former are the objects to be associated with algebraic expressions. By definition, Σ is *plain* if for each transition t, $\lambda(t)$ is a constant relabelling (recall that by our convention, thus, every transition label is an element of Lab). Operator boxes, by contrast, are patterns (functions) for defining operations on plain boxes. They will be introduced in section 3.1.4.

We turn to behavioural requirements for plain boxes. A plain box Σ is a *static box* if $\Sigma = \lfloor\Sigma\rfloor$ (i.e., its marking M_Σ is empty) and all markings reachable from $^\circ\Sigma$ are safe and clean. A plain box Σ is a *dynamic box* if $M_\Sigma \neq \emptyset$, $\lfloor\Sigma\rfloor$ is a static box, and all markings reachable from M_Σ are safe and clean. Note that when a box Σ is started from the initial marking $^\circ\Sigma$ then its markings are always nonempty: $^\circ\Sigma$ itself is nonempty because of ex-restrictedness and all its successor markings are nonempty because of T-restrictedness. On the other hand, the empty marking of a box has no successor markings except itself (reachable by the empty step sequence). Thus, the distinction between static and dynamic boxes is invariant over behaviour.

It is the static boxes Σ (or, equivalently, the dynamic boxes of the form $\overline{\Sigma}$) which in our approach ultimately provide a denotational semantics of algebraic expressions in process algebras such as PBC and CCS. Two behavioural conditions were imposed on the markings M reachable from the initial marking, $M \in [{}^\circ\Sigma\rangle$. The first is that we require M to be safe in order to ensure that the semantics of the boxes is as simple as possible and, in particular, that we can directly use a partial order semantics of nets in the style of Mazurkiewicz [27]. The second condition, that M is always a clean marking, is a consequence of the first condition and our wish to use iterative constructs in the algebra of nets which is developed in the rest of the paper. We will return to this question in section 3.2. Although static boxes are of our primary interest, we also need to be able to represent intermediate markings. For this reason we introduced dynamic boxes which, by definition, include all nets Θ such that Θ is derivable from $\overline{\Sigma}$, for some static box Σ.

Proposition 3.1 If Σ is a dynamic box and $\Sigma\ [U\rangle\ \Theta$ then Θ is a dynamic box and U is a set of independent transitions in Ind_Σ.

Proof. Θ has a nonempty marking since Σ is T-restricted. Clearly, all markings reachable from M_Θ are safe and clean since those reachable from M_Σ are. Hence Θ is a dynamic box. $U \in \mathsf{Ind}_\Sigma$ follows from the safeness of all markings reachable from M_Σ and the property that every subset of U is a step enabled at M_Σ. ■ 3.1

Within the set of dynamic boxes we further distinguish two special subclasses, called *entry* and *exit* boxes, which comprise all dynamic boxes Σ such that M_Σ is, respectively, ${}^\circ\Sigma$ and Σ°. The static, dynamic, entry and exit boxes will, respectively, be denoted by Box^s, Box^d, Box^e and Box^x.

We could now take any simple labelled net Ω and, using the definition of net refinement, derive from it a net operator, op_Ω. However, such an approach could only be seen as 'syntactic compositionality' since not all such operators op_Ω would enjoy what we regard as desirable behavioural properties. By this we mean compositionality properties which ensure that the behaviour of the resulting nets could be expressed from the behaviours of its constituent nets in a way similar to that used in the structural operational semantics (SOS) employed by process algebras. As it will turn out, we will be able to select 'suitable' Ω's by imposing simple conditions on their structure and behaviour (i.e., markings reachable from the initial markings).

3.1.2 Requirements for operator boxes

The discussion presented in this section and in section 3.2 is deliberately informal in order to better convey our intuition and motivation concerning the various conditions used in the definition of boxes. Although we do not formulate nor prove any formal results, the reader would presumably have little or no difficulty in extracting these from the text that follows.

To be able to define an SOS-style rule for compositionally defined nets, we shall require that the following be satisfied:

Requirement 1 (R1)
Operations on static boxes should not lead outside the semantic domain of static boxes.

Requirement 2 (R2)
Nets derivable from initially marked compositional boxes should be decomposable.

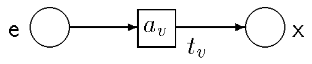
Requirement 3 (R3)
Steps executed by compositional boxes should be decomposable into substeps corresponding to the way the nets were composed.

To analyse the consequences of the above three requirements (and also make them more concrete), suppose that

$$\Omega = (S, T, W, \lambda, \emptyset)$$

is some (candidate) operator box. Let $\mathbf{\Theta}$ be a simple Ω-tuple of static boxes defined thus, for every $v \in T$:

Θ_v



where $a_v \neq a_w$ for $v \neq w$. To simplify the argument, but without losing its generality, we also assume that only the identity relabelling ρ_{id} is used to annotate the transitions in Ω. We start by making an easy observation that, due to the simple form of the Θ_v's, there is an isomorphism isom (which ignores transition labelling but preserves place labelling) between $\mathsf{op}_\Omega(\mathbf{\Theta})$ and Ω such that $\mathsf{isom}((v, a_v) \lhd t_v) = v$, for all $v \in T$. In other words, we can think of the net $\mathsf{op}_\Omega(\mathbf{\Theta})$ as though it were Ω with each transition v being labelled by a_v.
The first requirement (R1) means that $\mathsf{op}_\Omega(\mathbf{\Theta})$ should be a static box; hence, since isom preserves place labelling between $\mathsf{op}_\Omega(\mathbf{\Theta})$ and Ω, the latter must satisfy the defining conditions of a static box. As a result, we obtain the first condition to be used in the definition of operator boxes:

> Condition 1 (C1)
> All markings reachable from the initial marking of Ω are safe and clean.

We interpret the second requirement (R2) as saying that no matter how $\overline{\mathsf{op}_\Omega(\mathbf{\Theta})}$ evolves, the resulting net Σ should be deriveable compositionally using the same operator box; that is $\Sigma = \mathsf{op}_\Omega(\mathbf{\Phi})$ for some suitable Ω–tuple $\mathbf{\Phi}$ of boxes which, in particular, implies that $\lfloor \Phi_v \rfloor = \Theta_v$, for all $v \in T$.
Suppose now that M is the marking of a net Ψ derivable from $\overline{\Omega}$. Since Ω and $\mathsf{op}_\Omega(\mathbf{\Theta})$ are isomorphic, there is a net Σ derivable from $\overline{\mathsf{op}_\Omega(\mathbf{\Theta})}$ which isomorphic to Ψ. According to what we just said, there should be an Ω–tuple $\mathbf{\Phi}$ of boxes such that $\Sigma = \mathsf{op}_\Omega(\mathbf{\Phi})$ and $\lfloor \mathbf{\Phi} \rfloor = \mathbf{\Theta}$. We then can show that each Φ_v is either Θ_v or $\underline{\Theta}_v$ or $\overline{\Theta}_v$, in the following way: Clearly, M_{Φ_v} is a safe marking since otherwise M_Σ would not be safe (according to the definition of net refinement). Hence, if Φ_v does not satisfy the property, then both its entry and exit places must hold a token and ${}^\bullet v \cap v^\bullet = \emptyset$ (since otherwise M_Σ would not be safe). It is then easy to see that Σ's transition $(v, a_v) \lhd t_v$ is enabled at M_Σ and its execution leads to a marking which is not safe, contradicting $\mathsf{op}_\Omega(\mathbf{\Theta})$ being a static box.
The observation that each Φ_v is one of Θ_v, $\underline{\Theta}_v$ and $\overline{\Theta}_v$ leads to the idea that the marking M can be seen as a combination of pre- and post-sets of some of the transitions in Ω. Following this, we introduce two sets of transitions of Ω,

$$\begin{array}{rcl} \phi & = & \{v \mid \Phi_v = \overline{\Theta}_v\} \\ \text{and} \quad \psi & = & \{v \mid \Phi_v = \underline{\Theta}_v\}. \end{array}$$

The pair (ϕ, ψ) of sets of transitions of Ω (later called a *factorisation* of M) enjoys two interesting properties. First of all, we observe that ${}^\bullet\phi \cup \psi^\bullet = M$, which follows from $\mathsf{isom}(M_\Sigma) = M$ and

$$M_\Sigma \;=\; \bigcup_{s \in {}^\bullet\phi \cup \psi^\bullet} \mathsf{SP}^s_{\mathsf{new}} \qquad \text{and} \qquad {}^\bullet\phi \cup \psi^\bullet \;=\; \mathsf{isom}(\bigcup_{s \in {}^\bullet\phi \cup \psi^\bullet} \mathsf{SP}^s_{\mathsf{new}})$$

where the latter follows from $\mathsf{isom}((v, a_v) \lhd t_v) = v$, for all $v \in T$. We then note that the pre-sets of transitions in ϕ and the post-sets of transitions in ψ are mutually disjoint, which can easily be shown to follow from the safeness of M_Σ. Hence

$$M \;=\; \biguplus_{v \in \phi} {}^\bullet v \;\uplus\; \biguplus_{v \in \psi} v^\bullet.$$

As a result, M can be *partitioned* into disjoint sets of tokens corresponding to the pre– or post–sets of some of the transitions in Ω. The above leads us to

Condition 2a (C2a)
Each marking reachable from $^\circ\Omega$ has a factorisation, i.e., a pair (ϕ, ψ) as above.

The next requirement (R3) is directly motivated by the typical SOS operational semantics rule used in process algebras. Namely, whenever a term $\mathsf{op}(e_1, \ldots, e_n)$ can make a move

$$\mathsf{op}(e_1, \ldots, e_n) \xrightarrow{A}$$

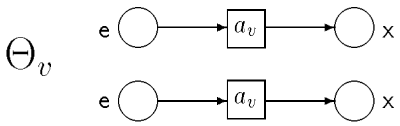
then A can be decomposed (not necessarily in a unique way) into individual moves $e_i \xrightarrow{A_i}$ so that the combination of the A_i's yields A. In CCS, for example, if $p \mid q \xrightarrow{\tau} p' \mid q'$ then one of the following holds:

$$\begin{array}{rcl} & p \xrightarrow{\tau} p' \quad \text{and} \quad q = q' \\ & p = p' \quad \text{and} \quad q \xrightarrow{\tau} q' \\ \text{or} & p \xrightarrow{a} p' \quad \text{and} \quad q \xrightarrow{\widehat{a}} q' \end{array}$$

where a is a visible action and $\widehat{a}$ is its complement.
In terms of nets of the current example, we interpret (R3) as implying that if Σ is derivable from $\mathsf{op}_\Omega(\mathbf{\Theta})$ and $\Sigma\ [U\rangle$ then there should be $\mathbf{\Phi}$ such that $\Sigma = \mathsf{op}_\Omega(\mathbf{\Phi})$, and steps U_v (for $v \in T$) such that $\Phi_v\ [U_v\rangle$ (for $v \in T$) and the labels of the transitions in the sets U_v are the same as those of the transitions in U. This leads, after a discussion similar to that carried out above,[4] to the conclusion that:

Condition 2 (C2)
If M is a marking reachable from the initial marking of Ω and U is a step enabled at M then there is a factorisation (ϕ, ψ) of (the places marked by) M such that $U \subseteq \phi$.

Note that (C2a) is (C2) with $U = \emptyset$ so only (C2) will be needed. The last issue we need to discuss is whether Ω should be pure. We claim that the first requirement (R1) entails that if v is a transition enabled at a marking reachable from $^\circ\Omega$ then $^\bullet v \cap v^\bullet = \emptyset$. This can be shown by taking the tuple $\mathbf{\Theta}$ with Θ_v changed to a concurrent composition as follows:



and refining such a box Θ_v into a net satisfying $^\bullet v \cap v^\bullet \neq \emptyset$. The details are left to the reader[5] who can also consult [4, 12]. In the definition of an operator box, we use slightly stronger, but simpler condition:

Condition 3 (C3)
Ω is pure.

To conclude, we have argued that if we base our net algebra on the specific notion of net refinement defined earlier on, and if we require that the resulting nets satisfy compositionality properties formulated as requirements (R1)–(R3), then Ω should satisfy conditions (C1)–(C3). The reader may wonder whether considering more complicated Θ_v may lead to conditions that are stronger than (C1)–(C3). However, this is not the case. To wit, it will be our aim in the rest of this paper to try to convince the reader that (C1)–(C3) do indeed give rise to a well–formed net algebra.

[4] Here the assumption that $a_v \neq a_w$ for $v \neq w$ is needed in order to map uniquely the labels of transitions in U to the (transitions in) nets $\mathbf{\Theta}$.

[5] Essentially, one can see that the resulting net is 2-bounded but not 1-safe.

3.1.3 Factorisations

Let Ω be a simple labelled net. A *factorisation* of a set of places R of Ω is a pair of sets of transitions $\mu = (\phi, \psi)$ such that:

$$R \;=\; \biguplus_{v\in\phi} {}^\bullet v \;\uplus\; \biguplus_{v\in\psi} v^\bullet.$$

We will use $\widetilde{\mu}$ to denote $\phi \cup \psi$ and call two factorisations, μ and κ, *disjoint* if $\widetilde{\mu} \cap \widetilde{\kappa} = \emptyset$. By fact_Ω we denote the set of all factorisations of the safe markings reachable from ${}^\circ\Omega$, and call Ω *factorisable* if for every safe marking M reachable from ${}^\circ\Omega$ and every step of transitions U enabled at M, there is a factorisation (ϕ, ψ) of M such that $U \subseteq \phi$.

3.1.4 Operator boxes

A simple finite box Ω without isolated places is an *operator box* if it is factorisable, pure, and all markings reachable from ${}^\circ\Omega$ are safe and clean. Note that this encodes conditions (C2), (C3) and (C1), respectively.

The domain of application of the net operator op_Ω which is induced by Ω, denoted by dom_Ω, is defined thus. Firstly, it comprises all Ω–tuples of static boxes. Secondly, for every $\mu = (\phi, \psi) \in \mathsf{fact}_\Omega$, dom_Ω comprises all Ω–tuples of boxes $\mathbf{\Sigma}$ such that

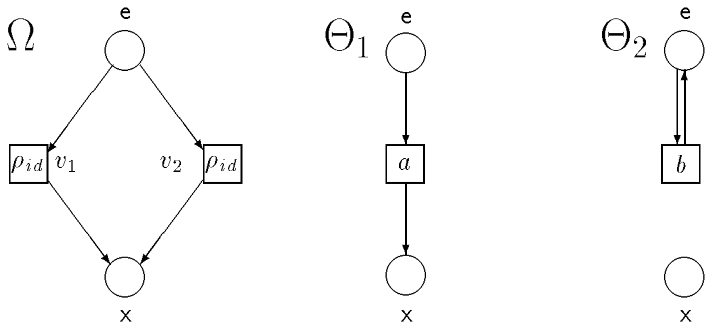
$$\begin{array}{rcl} \mathbf{\Sigma}_\phi & \subseteq & \mathsf{Box}^d \\ \mathbf{\Sigma}_\psi & \subseteq & \mathsf{Box}^x \\ \text{and} \quad \mathbf{\Sigma} \backslash \mathbf{\Sigma}_{\widetilde{\mu}} & \subseteq & \mathsf{Box}^s. \end{array}$$

We will denote $\mathbf{\Sigma} \in \mathsf{dom}_\Omega^\mu$ in such a case. The second part of the definition of the domain is correct since it is always the case that $\phi \cap \psi = \emptyset$ (cf. lemma A.3(2) in appendix A).

In the definition of dom_Ω^μ, condition $\mathbf{\Sigma}_\phi \subseteq \mathsf{Box}^d$ describes those transitions which contribute pre-sets to the factorisation μ (or more precisely, nets to be inserted for them which contribute *non-empty*, but not necessarily terminal, markings to the result, $\mathsf{op}_\Omega(\mathbf{\Sigma})$); $\mathbf{\Sigma}_\psi \subseteq \mathsf{Box}^x$ denotes transitions which contribute post-sets to μ (more precisely, nets which contribute *terminal* markings to $\mathsf{op}_\Omega(\mathbf{\Sigma})$); and $\mathbf{\Sigma} \backslash \mathbf{\Sigma}_{\widetilde{\mu}} \subseteq \mathsf{Box}^s$ denotes the remaining transitions (or nets), not contributing to the current marking. Condition $\mathbf{\Sigma}_\psi \subseteq \mathsf{Box}^x$ is needed explicitly because – as will be discussed later in more detail – the terminal marking of some net can also be interpreted as the initial marking of another net within $\mathsf{op}_\Omega(\mathbf{\Sigma})$. Condition $\mathbf{\Sigma}_\phi \subseteq \mathsf{Box}^d$ cannot be replaced by $\mathbf{\Sigma}_\phi \subseteq \mathsf{Box}^e$ because reachable intermediate markings (of the nets to be inserted for transitions) need to be taken into account. Condition $\mathbf{\Sigma}_\psi \subseteq \mathsf{Box}^x$ cannot be replaced by $\mathbf{\Sigma}_\psi \subseteq \mathsf{Box}^d$ because this would essentially lead to modelling backward behaviour. It may not be immediately clear that the above definition of an operator box is what is needed. Indeed, even the seemingly straightforward result that its application never leads outside the semantic domain (theorem 3.4 below), comes only after a lengthy sequence of auxiliary lemmata proved in appendix A.

3.2 Discussion of some properties of boxes

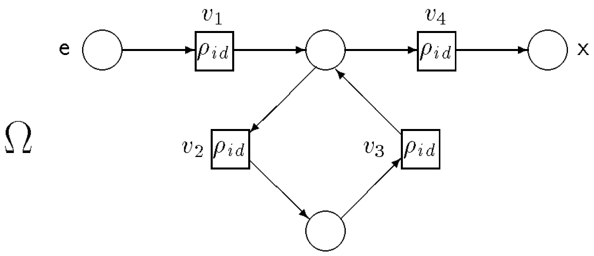
We first explain why we require boxes to be ex–directed, by taking as an example an operator box Ω and two nets, Θ_1 and Θ_2, defined thus:



In this example, Ω models a choice construct which is a part of the PBC algebra (and of almost any other process algebra, too). Intuitively, $\overline{\mathsf{op}_\Omega(\mathbf{\Theta})}$ should behave either as $\overline{\Theta}_1$ or as $\overline{\Theta}_2$. However, this is not the case since it is possible to execute a step sequence (of labels) $\langle b\rangle\langle b\rangle\langle b\rangle\langle a\rangle$, which could not possibly be generated by any of the two nets in $\mathbf{\Theta}$. Now, the reason for this is that Θ_2 can execute, starting and ending at its initial marking, a nonempty step sequence. We prevent such an undesirable effect by the structural condition that there be no arcs incoming to the entry places. A similar condition is imposed on exit places. Nevertheless, in general, if one is willing to sacrifice this behavioural modularity then the framework lends itself to the purpose as well – all the formalism goes through without change when ex-directedness is not required.

To see why the property of ex–restrictedness has been introduced, consider the above example, except that box Θ_2 is replaced by the empty net $\Theta_2' = (\emptyset, \emptyset, \emptyset, \emptyset)$ (which would be admissible in the absence of ex–restrictedness). The notion of refinement is still well-defined, and the result of refining the two transitions of Ω simultaneously by Θ_1 and Θ_2', respectively, is an isolated a-labelled transition. Such a transition can occur many times concurrently to itself, which seems to be counterintuitive; see also section 4.5.5 for further discussion.

We now turn to the property that all markings reachable from the initial marking in a dynamic or static box Θ should be clean. Suppose that this is not the case and, for example, ${}^\circ\Theta\ [\{t_1\}\ldots\{t_k\}\rangle\ M$ where M is a safe marking such that $\Theta^\circ \subseteq M$ and $\Theta^\circ \neq M$. Then it can be shown that a contradiction can arise between our wish to model iterative constructs and the required safeness of boxes. To be more concrete, consider the following operator box modelling a simple iterative construct:



We can take an Ω–tuple $\mathbf{\Theta}$ where $\Theta_{v_2} = \Theta$ and the remaining Θ_v's are as at the beginning of this section. It can be shown that the following is a valid sequence starting at the initial marking of $\mathsf{op}_\Omega(\mathbf{\Theta})$ and leading to a marking which is not safe:

$$\sigma = \{(v_1, a_{v_1}) \lhd t_{v_1}\}\sigma'\{(v_3, a_{v_3}) \lhd t_{v_3}\}\sigma'$$

where $\sigma' = \{(v_2, \lambda_{v_2}(t_1)) \lhd t_1\}\ldots\{(v_2, \lambda_{v_2}(t_k)) \lhd t_k\}$.

Finally, consider the N-box shown in figure 6. It is not factorisable. To see this, consider the marking obtained from the initial marking after occurrences of v_1 and v_3. Its 'left' token can be factored out using v_3 and the 'right' token can be factored out using v_2, but the 'middle' token is orphaned. We will return to this example later on.

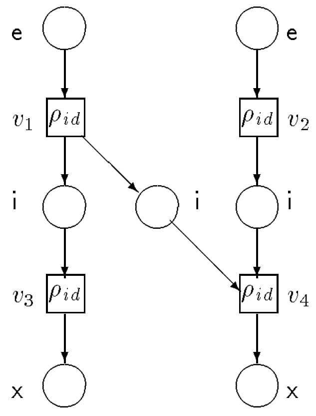


Figure 6: A non-factorisable box.

3.3 Operational semantics, its soundness and completeness

In its most basic form, the SOS rule for process algebras stipulates that if a term $\mathsf{op}(e_1, \ldots, e_n)$ can make a move A then A can be composed out of some individual moves of its sub-terms. And conversely, if the sub-terms can individually make moves then the whole term can make a combined move.

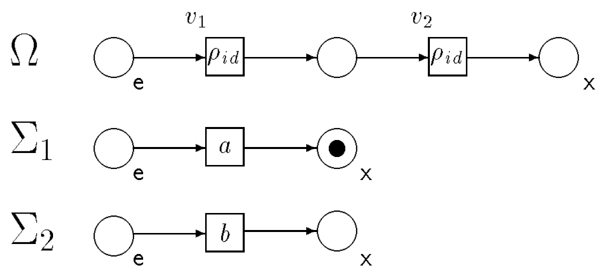
Let Ω be an operator box and $\mathbf{\Sigma} \in \mathsf{dom}_\Omega$. When dealing with the SOS semantics of the compositionally defined box $\mathsf{op}_\Omega(\mathbf{\Sigma})$, we use the notation

$$(\Omega : \mathbf{\Sigma}) \xrightarrow{U} (\Omega : \mathbf{\Theta}) \tag{1}$$

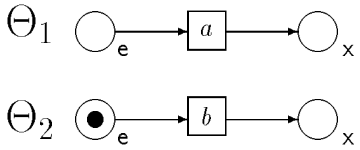
to express the fact that the boxes $\mathbf{\Sigma}$ can individually make moves which, when combined, yield step U and lead to new boxes $\mathbf{\Theta}$. This will be the case whenever for each Σ_v in $\mathbf{\Sigma}$ there is a (possibly empty) set of transitions $U_v = \{u_1, \ldots, u_k\}$ of $\mathsf{T}^v_{\mathsf{new}}$ (cf. section 2.2.5) such that

$$\Sigma_v \; [\, \mathsf{trees}(u_1) \uplus \cdots \uplus \mathsf{trees}(u_k) \,\rangle \; \Theta_v$$

and U is the union of all the sets U_v for $v \in T$. It can be shown that (1) implies $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Sigma})$ $[U\rangle$ $\mathsf{op}_\Omega(\mathbf{\Theta})$ which is the (easier) half of the announced SOS rule for boxes. The converse, however, does not in general hold true. For consider the following three nets:



Then $\mathsf{op}_\Omega(\mathbf{\Sigma})$ $[\{t\}\rangle$ where t is a transition labelled b, yet no nonempty move at all of the form $(\Omega : \mathbf{\Sigma}) \xrightarrow{U}$ is possible. This is so because when composing the nets, the token contributed by Σ_1 is inserted into the composed net in such a way that it could have been contributed by Σ_2. More precisely, we have $\mathsf{op}_\Omega(\mathbf{\Sigma}) = \mathsf{op}_\Omega(\mathbf{\Theta})$, where Θ_1 and Θ_2 are boxes defined in the following way:



i.e., $\Theta_1 = \lfloor\Sigma_1\rfloor$ and $\Theta_2 = \overline{\Sigma_2}$ and, conversely, $\Sigma_1 = \underline{\Theta_1}$ and $\Sigma_2 = \lfloor\Theta_2\rfloor$. And the situation is now different since $(\Omega : \mathbf{\Theta}) \xrightarrow{\{t\}}$. The conclusion to be drawn from this example is that the markings in the nets $\mathbf{\Sigma}$ may need to be rearranged before attempting to derive a move which is admitted by their composition. Such rearrangement is formalised using the relation $\equiv_\Omega$ defined next.

Let Ω be an operator box, and let $\mathbf{\Sigma}$ and $\mathbf{\Theta}$ be Ω–tuples of boxes such that $\lfloor\mathbf{\Sigma}\rfloor = \lfloor\mathbf{\Theta}\rfloor$. We write $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$ if there are disjoint factorisations, $\eta = (\eta_1, \eta_2)$ and $\psi = (\psi_1, \psi_2)$, of the same set of places of Ω such that $\Sigma_v = \Theta_v$ for $v \notin \widetilde{\eta} \cup \widetilde{\psi}$ and

$$\begin{array}{rrcl} & \mathbf{\Sigma}_{\eta_1} \cup \mathbf{\Theta}_{\psi_1} & \subseteq & \mathsf{Box}^e \\ & \mathbf{\Sigma}_{\eta_2} \cup \mathbf{\Theta}_{\psi_2} & \subseteq & \mathsf{Box}^x \\ \text{and} & \mathbf{\Sigma}_{\widetilde{\psi}} \cup \mathbf{\Theta}_{\widetilde{\eta}} & \subseteq & \mathsf{Box}^s. \end{array}$$

Note that $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Sigma}$ ($\equiv_\Omega$ is reflexive) since we can take all-empty factorisations. The relation $\equiv_\Omega$ enjoys two basic properties: it is closed with respect to the domain of op_Ω and it relates tuples of boxes which yield the same nets through the composition specified by op_Ω (lemma A.11 in the appendix). That is, if $\mathbf{\Sigma} \in \mathsf{dom}_\Omega$ and $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$ then $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Sigma}) = \mathsf{op}_\Omega(\mathbf{\Theta})$.

We now formulate two central results of this paper. Together, they mean that for the chosen class of operator boxes, the standard step sequence semantics of compositionally defined nets obeys a variant of the SOS rule introduced originally for process algebras. Below it is assumed that Ω is an operator box and $\mathbf{\Sigma} \in \mathsf{dom}_\Omega$.

Theorem 3.2 (Soundness of the SOS rule)
If $(\Omega : \mathbf{\Sigma}) \xrightarrow{U} (\Omega : \mathbf{\Theta})$ then $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Sigma})\ [U\rangle\ \mathsf{op}_\Omega(\mathbf{\Theta})$. □

Theorem 3.3 (Completeness of the SOS rule)
If $\mathsf{op}_\Omega(\mathbf{\Sigma})\ [U\rangle\ \Sigma$ then there are $\mathbf{\Theta}, \mathbf{\Gamma} \in \mathsf{dom}_\Omega$ such that $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$, $\mathsf{op}_\Omega(\mathbf{\Gamma}) = \Sigma$ and

$$(\Omega : \mathbf{\Theta}) \xrightarrow{U} (\Omega : \mathbf{\Gamma}).$$

□

The SOS rule can be presented in a way similar to that found in process algebras:

$$\boxed{\frac{\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta} \quad , \quad (\Omega : \mathbf{\Theta}) \xrightarrow{U} (\Omega : \mathbf{\Gamma}) \quad , \quad \mathsf{op}_\Omega(\mathbf{\Gamma}) = \Sigma}{\mathsf{op}_\Omega(\mathbf{\Sigma})\ [U\rangle\ \Sigma}} \tag{2}$$

Note that theorem 3.2 (together with the basic properties of $\equiv_\Omega$) verifies the soundness of the above rule, whereas theorem 3.3 verifies its completeness. Both theorems are proved in appendix A, and their proofs rely on factorisability property of the operator boxes.

Theorem 3.4 If Ω is an operator box then $\mathsf{op}_\Omega(\mathsf{dom}_\Omega) \subseteq \mathsf{Box}^s \cup \mathsf{Box}^d$. □

The above is a basic soundness result which makes op_Ω acceptable with regard to requirement (R1). There are other interesting behavioural properties of compositionally defined boxes which can be derived (more or less directly) from the last three theorems. We only mention one, which captures requirement (R2). It follows directly from theorem 3.3 and the assumption that $°\Omega$ has a factorisation.

Corollary 3.5 If Ω is an operator box, $\mathbf{\Sigma}$ is an Ω-tuple of static boxes and Σ is a net derivable from $\overline{\mathsf{op}_\Omega(\mathbf{\Sigma})}$, then there is $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ such that $\Sigma = \mathsf{op}_\Omega(\mathbf{\Theta})$. □

The various net operations used by the PBC, including its original (and other) synchronisation disciplines can be accommodated in the meta-scheme (2) presented above. How this can be done in general, is described below in sections 5 and 6. Section 3.2 presented an example of a non-factorisable box where the completeness theorem fails (one can show this using an argument similar to that used there to demonstrate that the N-box is non-factorisable).

4 Recursion

In this section we investigate the problem of solving recursive definitions on boxes. In our approach, it is the *names* of places and transitions which play a crucial role since we use them to define an *inclusion order* on the domain of plain boxes, and then solve recursive definitions by successive approximations with respect to the inclusion order. This should be contrasted with a conceivable alternative approach of defining such an order through injective mappings (in such a case nets could be defined up to isomorphism, and the names of places and transitions would not matter). As discussed in [5], one of the problems with the latter approach is that the resulting order is not necessarily a partial order but only a pre-order; antisymmetry is not necessarily guaranteed if two nets can be injectively embedded into each other. For this reason, and also because the tree naming device has turned out to be ready-made for use in solving recursive equations, we have decided to explore not the latter but the former approach. In particular, we shall demonstrate that it enjoys two important properties:

- *Generality.* We can solve arbitrarily complex recursive definitions on boxes (theorem 4.7 in this section).
- *Correctness.* The boxes which are generated as solutions possess behavioural properties which match those of the corresponding recursive expressions in the associated process algebra (theorems 5.1 and 5.2 in the next section).

It is outside our present knowledge to say whether the second approach could be made to enjoy similar properties, perhaps by using approximation theory on pre-orders [31]. This question remains to be addressed. In any case, the alternative approach to solving equations on nets would always admit at least as many non-isomorphic solutions as the one proposed here (since any approximation based on inclusion order induces a solution based on injective mappings). Hence, e.g., there might be recursive equations for whom inclusion order based method would yield a *unique* solution, which is usually regarded as a desirable property, whereas the injective mapping based method would lead to several non-isomorphic solutions.

This section is structured as follows. Section 4.1 contains a statement of the problem to be solved, as well as some basic definitions. In section 4.2 we define an inclusion order on boxes and provide some basic results on the solvability of recursive definitions on boxes; in particular, we show that operator boxes induce continuous mappings on the domain of boxes. Section 4.3 shows an example of how to use inclusion order to solve recursive equations. Section 4.4 is devoted to finding boxes which can be used as starting points for the approximation of solutions of recursive equations. It also contains the main result of the whole section which states that every recursive equation has a solution. Section 4.5 discusses more examples. Relevant proofs appear in appendix B.

4.1 Equations on boxes

Let Var be a nonempty, possibly uncountable, set of recursion variables which are used as place–holders for static boxes. We consider a system of *equations on boxes* of the following form (one equation for every $X \in \mathsf{Var}$):

$$X \stackrel{\mathrm{df}}{=} (\Omega_X : \mathbf{\Delta}_X). \tag{3}$$

In the right-hand side, Ω_X is an operator box and $\mathbf{\Delta}_X$ is an Ω_X–tuple of static boxes and/or recursion variables. A *solution* of the system (3) is an assignment of a static box Σ_Y to every variable $Y \in \mathsf{Var}$ so that if every occurrence of Y is replaced in every $\mathbf{\Delta}_X$ by Σ_Y then, for all variables $X \in \mathsf{Var}$,

$$\Sigma_X \quad = \quad \mathsf{op}_{\Omega_X}(\mathbf{\Delta}_X).$$

By definition, the *dependency graph* of (3) is the directed graph whose nodes are the variables Var and whose arcs (X, Y) indicate that Y occurs on the right-hand side of the equation having X as its left-hand side. The equation system (3) is called *recursion bounded* if its dependency graph has no forward-infinite directed paths (note that, as a consequence, the graph must then be acyclic); in particular, if none of the right-hand sides of (3) has a recursion variable then the system is recursion bounded since its dependency graph has no arcs at all. If the system (3) is recursion bounded then using net refinement a set of static boxes solving it can be calculated by executing the following procedure: starting with equations that have no variables on their right-hand sides (such equations must exist by recursion boundedness), use net refinement repeatedly until all left-hand sides have been calculated. That such a procedure is correct follows from König's lemma, theorem 3.4, the finiteness of operator boxes and the absence of infinite directed paths in the dependency graph. The boxes solving the system of equations are unique in that case. Thus,

Proposition 4.1 If the system (3) is recursion bounded then it has a unique solution. □

We will later show that the equation system (3) always has a solution; however, such a solution does not have to be unique. Throughout the rest of this section we only consider a single recursive equation:

$$X \stackrel{\mathrm{df}}{=} (\Omega : \mathbf{\Delta}) \tag{4}$$

for a singleton set $\mathsf{Var} = \{X\}$. All the results are easily lifted to the general case using standard techniques on Cartesian product spaces (for a finite set of equations) and, respectively, function spaces (for infinite sets of equations). For every box Σ, we will use $\mathbf{\Delta}^{\Sigma}$ to denote the tuple $\mathbf{\Delta}$ appearing in the recursive equation (4) with all the occurrences of variable X replaced by Σ.

In the rest of this section, we restrict our interest to the set of *plain boxes with empty markings*, Box^{rec}. Their empty markings, for simplicity, will be omitted. The right-hand side of the recursive equation (4) induces a mapping $\mathsf{F} : \mathsf{Box}^{rec} \to \mathsf{Box}^{rec}$ defined, for every box $\Sigma \in \mathsf{Box}^{rec}$, by

$$\mathsf{F}(\Sigma) = \mathsf{op}_{\Omega}(\mathbf{\Delta}^{\Sigma}). \tag{5}$$

With such a definition, solving equation (4) amounts to finding fixpoints of the mapping F in the domain of *static* boxes. Note that the codomain of F is well defined which follows directly from the definition of net refinement, i.e., for every $\Sigma \in \mathsf{Box}^{rec}$, $\mathsf{op}_{\Omega}(\mathbf{\Delta}^{\Sigma})$ is in Box^{rec} (see lemma A.1(8) in appendix A).

4.2 Approximating solutions of recursive equations

We define an *inclusion order*, $\sqsubseteq$, on boxes in Box^{rec}. Let Σ and Θ be two boxes in Box^{rec}. Then $\Sigma \sqsubseteq \Theta$ if $S_\Sigma \subseteq S_\Theta$, $T_\Sigma \subseteq T_\Theta$, $\lambda_\Sigma \subseteq \lambda_\Theta$ and $W_\Sigma \subseteq W_\Theta$. Note carefully that $W_\Sigma \subseteq W_\Theta$ not just simply means that Θ has 'more arrows' than Σ. For instance, if $s \in S_\Sigma$, $t \in T_\Sigma$ and there is no arrow from s to t in Σ then $((s,t),0) \in W_\Sigma$ and hence $((s,t),0) \in W_\Theta$, i.e., in Θ there is also no arrow from s to t. Clearly, $\sqsubseteq$ is a partial order on Box^{rec}. Note, however, that $\sqsubseteq$ does not turn $(\mathsf{Box}^{rec}, \sqsubseteq)$ into a lattice. For example, the least element is missing (the empty net is not a box) and, also, not every two elements have a least upper bound (as they may be incompatible so that their union cannot be formed), nor greatest lower bound, for that matter. However, it is possible to form chains and limits in the usual way. A *chain* is an infinite sequence of boxes in Box^{rec}, $\chi = (\Sigma_0, \Sigma_1, \ldots)$ such that $\Sigma_0 \sqsubseteq \Sigma_1 \sqsubseteq \ldots$ and its *limit* is defined as:

$$\bigsqcup \chi \;=\; (\bigcup_{i=0}^{\infty} S_{\Sigma_i}, \bigcup_{i=0}^{\infty} T_{\Sigma_i}, \bigcup_{i=0}^{\infty} W_{\Sigma_i}, \bigcup_{i=0}^{\infty} \lambda_{\Sigma_i}).$$

It is easy to see that $\bigsqcup \chi$ is always a box in Box^{rec} which is the least upper bound, in the sense of $\sqsubseteq$, for the boxes in χ. The first basic result needed for solving recursive equations is:

Theorem 4.2 If χ is a chain of static boxes then $\bigsqcup \chi$ is also a static box. □

A mapping $f : \mathsf{Box}^{rec} \to \mathsf{Box}^{rec}$ is *monotonic* if, for all Σ and Θ in Box^{rec}, $\Sigma \sqsubseteq \Theta$ implies $f(\Sigma) \sqsubseteq f(\Theta)$. It is *continuous* if for every chain $\chi = (\Sigma_0, \Sigma_1, \ldots)$ of boxes in Box^{rec},

$$f(\bigsqcup \chi) = \bigsqcup (f(\Sigma_0), f(\Sigma_1), \ldots)$$

As the reader might already suspect, both properties are enjoyed by the mapping F defined in (5) and induced by the right-hand side of the recursive equation (4).

Theorem 4.3 The function F defined in (5) is monotonic and continuous. □

Essentially, what the theorem says is that every operator box induces a monotonic and continuous mapping on the domain of plain boxes. With the last two results, we can attempt to solve equation (4), by defining a chain formed through successive applications of the mapping F to some initial approximation of a solution. For every static box Σ^0, let

$$\mathsf{chain}(\Sigma^0) = (\Sigma^0, \Sigma^1, \Sigma^2, \ldots)$$

where $\Sigma^i = \mathsf{F}(\Sigma^{i-1})$, for all $i \geq 1$. We will call Σ^0 a *seed* box for the equation (4) if $\Sigma^0 \sqsubseteq \Sigma^1$. Note that in such a case, by the monotonicity of F and theorem 3.4, $\mathsf{chain}(\Sigma^0)$ is a chain of static boxes. We now observe that the limit construction applied to the chain derived from a seed box always yields a solution. This follows directly from theorems 3.4, 4.2 and 4.3, using standard techniques of fixpoint theory of continuous functions over partial order domains [10].

Theorem 4.4 If Σ^0 is a seed box then $\bigsqcup \mathsf{chain}(\Sigma^0)$ is a solution of equation (4). □

We next obtain that each solution is uniquely identified by its entry and exit places, and that it is sufficient to consider ex boxes as the only seeds.

Theorem 4.5 Let Σ and Θ be solutions of equation (4), and $\Sigma^0 = ({}^\circ\Sigma \cup \Sigma^\circ, \emptyset, \emptyset, {}^\circ\Sigma \times \{\mathsf{e}\} \cup \Sigma^\circ \times \{\mathsf{x}\})$.

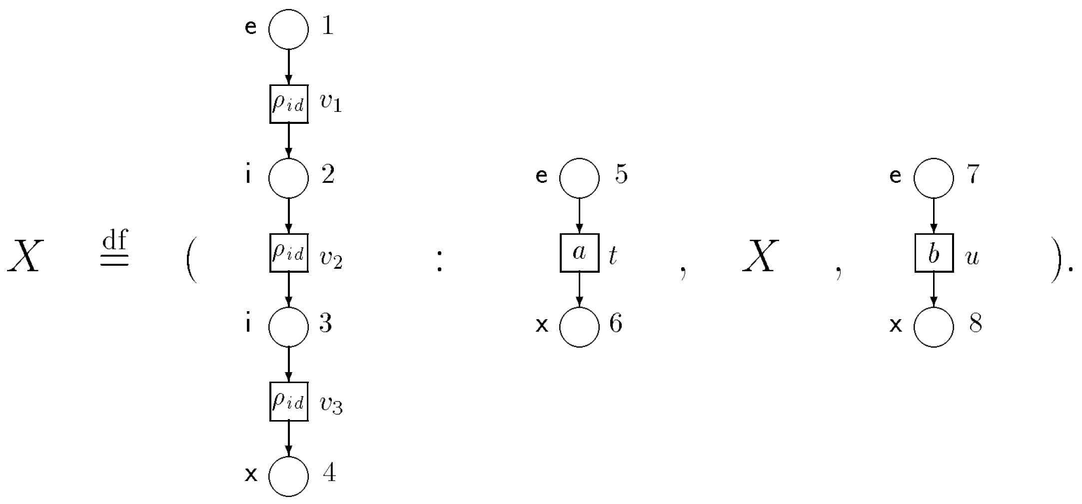
(1) If ${}^\circ\Sigma = {}^\circ\Theta$ and $\Sigma^\circ = \Theta^\circ$ then $\Sigma = \Theta$.

(2) Σ^0 is a seed box for (4) such that $\bigsqcup \mathsf{chain}(\Sigma^0) = \Sigma$. □

What we still lack to complete the discussion of the recursive equation (4) is a theorem saying that it always has a solution. This result will be postponed until section 4.4, after we will have discussed in more detail the seed boxes. First, however, we present an example.

4.3 Example of the limit construction

Let us consider as an example the following instance of equation (4):



(6)

Note that on the right-hand side there is a 3-transition operator box with three arguments, namely, two static boxes Δ_1 and Δ_3 for v_1 and v_3, respectively, and a recursion variable X for v_2. Informally speaking,[6] equation (6) corresponds to the recursive equation $X \stackrel{\mathrm{df}}{=} a; X; b$, where the semicolon represents sequential composition.

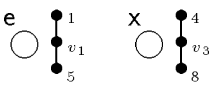
Let us examine how a solution of the net equation (6) might be constructed using the limit construction. Suppose that there already exists an approximation Σ^0 given as a starting point, a hypothetical seed box. Then inserting that approximation in the place of the X on the right-hand side of equation (6) yields a new box

$$\Sigma^1 = \mathsf{op}_\Omega(\Delta_1, \Sigma^0, \Delta_3)$$

which is a better approximation. We then construct another approximation

$$\Sigma^2 = \mathsf{op}_\Omega(\Delta_1, \Sigma^1, \Delta_3)$$

by the same principle, and so on. We already know that if Σ^0 is a seed then the limit of the chain $\Sigma^0, \Sigma^1, \ldots$ so constructed exists and solves the equation (6). We claim that the ex box

$\Sigma^0 =$ 

(7)

(with these particular place names) serves as a good seed for (6). Figure 7 shows the first three approximations resulting from this seed, except that we do not represent all name trees explicitly (the reader may easily guess the missing ones by analogy).

Figure 8 shows the limit of $\mathsf{chain}(\Sigma^0)$, i.e., the limit of the infinite series of boxes whose first three elements are shown in figure 7. The reader may check that this box satisfies the equation (6) in the strictest possible sense of the word. To see this, replace the variable X on the right-hand side of equation (6) by the box Σ shown in figure 8, perform net refinement $\mathsf{op}_\Omega(\Delta_1, \Sigma, \Delta_3)$, where Ω is the operator box of (6), and check that the result is not just isomorphic but even *identical* to Σ, i.e., it has the same places and the same transitions with the same connections and the same labels.

Thus, once an appropriate starting point Σ^0 has been 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, as we have shown, leads to a solution in all cases. But one still could say that the way in which the example equation has been solved was not fully satisfactory since the seed box was 'guessed' rather than 'computed'. This will be remedied in the next section.

[6] This will be made more precise in sections 5 and 6 below.

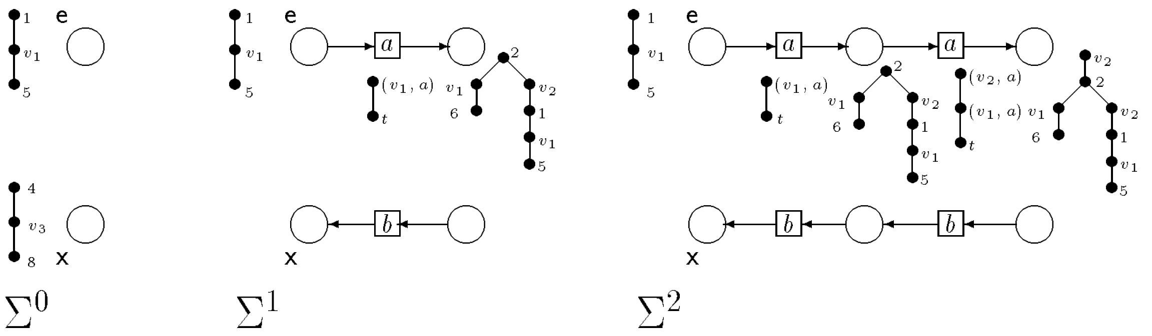


Figure 7: Three approximations Σ^0, Σ^1 and Σ^2 for equation (6).

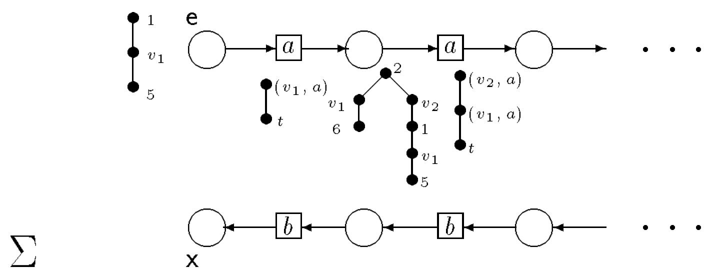


Figure 8: A box $\Sigma = \bigsqcup(\Sigma^0, \Sigma^1, \Sigma^2, \ldots)$ solving equation (6).

4.4 Deriving seed boxes

In this section we will address a question how to find a suitable starting point, i.e., a seed box Σ^0, for the approximation of a solution of the recursive equation (4). In particular, we want to ensure that $\Sigma^0 \sqsubseteq \Sigma^1$ holds for Σ^0 and the next approximation Σ^1.

One might ask why the box (7) is a suitable starting point for the approximation of the solutions of equation (6). 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 and because, as a consequence of theorem 4.5(2), all solutions can be so obtained. 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 $\sqsubseteq$. We will come back to the latter point later. Let us give the starting ex box a generic name,

$$\Sigma^0 \;=\; (\mathsf{P_e} \cup \mathsf{P_x}, \emptyset, \emptyset, \mathsf{P_e} \times \{\mathsf{e}\} \cup \mathsf{P_x} \times \{\mathsf{x}\}) \tag{8}$$

and develop conditions for it to be a seed box. First, because of ex-restrictedness, $\mathsf{P_e}$ and $\mathsf{P_x}$ should both be nonempty. They should also be disjoint, otherwise the labelling function would be ill-defined. Finally, with a view to obtain the inclusion property, $\Sigma^0 \sqsubseteq \Sigma^1$, some conditions can be imposed on the sets $\mathsf{P_e}$ and $\mathsf{P_x}$. More precisely, due to a particularly simple form of Σ^0, $\Sigma^0 \sqsubseteq \Sigma^1$ translates into two inclusions over sets of trees of the form defined in section 2.2.2. In order to express such conditions, we use the appending construct $\lhd$ lifted to sets of trees, in the following way (below, $\mathsf{P_{tree}}$ denotes the set of all trees used as place names, as defined in section 2.2.2).

Let $s \in \mathsf{P_{root}}$ and $v_1, \ldots, v_m \in \mathsf{T_{root}}$ ($m \geq 1$) be distinct single root trees, and $P_1, \ldots, P_m$ be subsets of $\mathsf{P_{tree}}$. Then

$$s \lhd (v_1 : P_1, \ldots, v_m : P_m) = \{s \lhd \{v_1 \lhd p_1, \ldots, v_m \lhd p_m\} \mid \forall i : \; p_i \in P_i\}.$$

We will need two mappings, $\mathsf{f_e}, \mathsf{f_x} : 2^{\mathsf{P_{tree}}} \to 2^{\mathsf{P_{tree}}}$, defined thus. Suppose that the entry places of the operator box Ω (c.f. equation (4)) are $s_1, \ldots, s_k$ and that the exit places are $r_1, \ldots, r_m$. Moreover, assume that the post-set of each place s_i consists of transitions $v_i^1, \dots, v_i^{k_i}$, and the pre-set of each place r_j consists of transitions $t_j^1, \dots, t_j^{m_j}$ (note that the pre-sets of the s_i's and the post-sets of t_j's are all empty). Then, for every subset P of $\mathsf{P}_{\mathsf{tree}}$,

$$\begin{aligned} \mathsf{f_e}(P) &= \bigcup_{i=1}^{k} s_i \lhd (v_i^1 : P_i^1, \dots, v_i^{k_i} : P_i^{k_i}) \\ \text{and} \quad \mathsf{f_x}(P) &= \bigcup_{j=1}^{m} r_j \lhd (t_j^1 : Q_j^1, \dots, t_j^{m_j} : Q_j^{m_j}) \end{aligned} \tag{9}$$

where $P_i^l = P$ if $\Delta_{v_i^l}$ in the tuple $\mathbf{\Delta}$ is the variable X, and $P_i^l = {}^\circ(\Delta_{v_i^l})$ otherwise; and similarly, $P_j^l = P$ if $\Delta_{t_j^l} = X$, and $P_j^l = (\Delta_{t_j^l})^\circ$ otherwise. It is a matter of simple check to verify that the two functions can be used to calculate the entry and exit places of Σ^1, i.e., we have

$$\begin{aligned} \mathsf{f_e}({}^\circ(\Sigma^0)) &= {}^\circ\mathsf{F}(\Sigma^0) &= {}^\circ(\Sigma^1) \\ \text{and} \quad \mathsf{f_x}((\Sigma^0)^\circ) &= \mathsf{F}(\Sigma^0)^\circ &= (\Sigma^1)^\circ. \end{aligned}$$

As a result, Σ^0 is a seed box if and only if the following two inclusions hold

$$\begin{aligned} \mathsf{P_e} &\subseteq \mathsf{f_e}(\mathsf{P_e}) \\ \text{and} \quad \mathsf{P_x} &\subseteq \mathsf{f_x}(\mathsf{P_x}) \end{aligned} \tag{10}$$

and, moreover, $\mathsf{P_e}$ and $\mathsf{P_x}$ are nonempty sets (see lemma B.3). Thus we have reduced the problem of finding a seed box for equation (4) to that of finding two nonempty sets of trees, $\mathsf{P_e}$ and $\mathsf{P_x}$, satisfying (10). And, as it turns out, we can always find two trees, $\mathsf{spine_e}$ and $\mathsf{spine_x}$, such that $\mathsf{P_e} = \{\mathsf{spine_e}\}$ and $\mathsf{P_x} = \{\mathsf{spine_x}\}$ do satisfy (10) and hence yield a seed box. It is both instructive (for the examples that follow) and necessary (for proving technical results) to demonstrate that such trees, $\mathsf{spine_e}$ and $\mathsf{spine_x}$, can be constructed. We only show this for $\mathsf{spine_e}$, the other case being similar.

Construction 4.6 (of tree $\mathsf{spine_e}$)
Take the first component of the union for $\mathsf{f_e}$ in (9) (this is possible because Ω has at least one entry place and thus $k \geq 1$). Suppose that $s = s_1$, $k_1 = l$ and $v_1 = v_1^1, \dots, v_l = v_1^l$. Without loss of generality, we may assume that that the instances of X in $\Delta_{v_1}, \dots, \Delta_{v_l}$ are $\Delta_{v_1}, \dots, \Delta_{v_n}$, where $0 \leq n \leq l$. Take, from each set ${}^\circ(\Delta_{v_i})$, for $n < i \leq l$, any tree τ_i (note that the τ_i's exist since a box has always at least one entry place).
If $n = 0$, i.e., if no Δ_{v_i} is the variable X, then we define

$$\mathsf{spine_e} = s \lhd \{v_1 \lhd \tau_1, \dots, v_l \lhd \tau_l\}.$$

Otherwise, we construct an infinite tree $\mathsf{spine_e}$. The nodes of $\mathsf{spine_e}$ are the set of strings of integers generated by the following regular expression:

$$0\{10, 20, \dots, n0\}^*\{\epsilon, 1, 2, \dots, l, (n+1)0, \dots, l0\}.$$

There is an arc from node x to node y if and only if x is a proper prefix of y. The labelling of 0 and every node ending with $i0$ ($1 \leq i \leq n$) is s, and the labelling of every node ending with i different from 0 is v_i. We then obtain $\mathsf{spine_e}$ by inserting a fresh copy of the tree τ_i for each (leaf) node ending with $i0$, for $n < i \leq l$. I.e. each such node is identified with the root node of a copy of τ_i. The definition of $\mathsf{spine_e}$ is illustrated in figure 9, where it is assumed that $n = 2$ and $l = 3$. □

Let us call a tree $\mathsf{spine_e}$ obtained through construction 4.6 a *spine* associated with $\mathsf{f_e}$. Of course, to a given $\mathsf{f_e}$ there may be many different spines. We now observe that the tree $\mathsf{spine_e}$ defined in construction 4.6 satisfies $\{\mathsf{spine_e}\} \subseteq \mathsf{f_e}(\{\mathsf{spine_e}\})$ since, as we define transition and place trees up to isomorphism,

$$\mathsf{spine_e} = s \lhd \{v_1 \lhd \mathsf{spine_e}, \dots, v_n \lhd \mathsf{spine_e}, v_{n+1} \lhd \tau_{n+1}, \dots, v_l \lhd \tau_l\}.$$

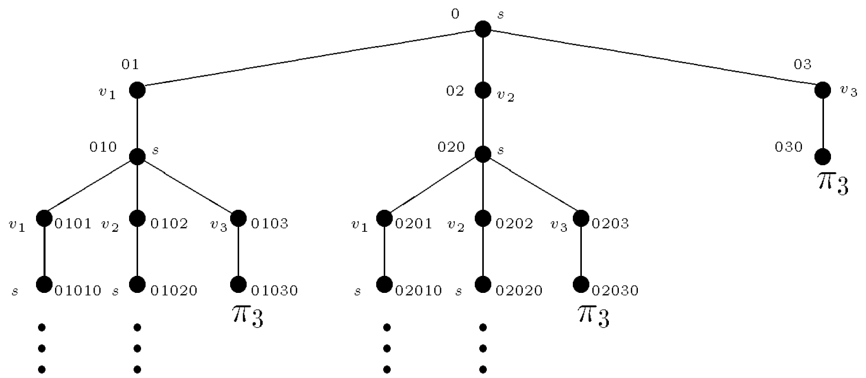


Figure 9: $\mathsf{spine_e}$ tree.

And, by symmetry, the same holds for $\mathsf{spine_x}$. It therefore follows that by setting

$$\begin{array}{rrcl} & \mathsf{P_e} & = & \{\mathsf{spine_e}\} \\ \text{and} & \mathsf{P_x} & = & \{\mathsf{spine_x}\} \end{array}$$

we obtain a valid seed box for the equation (4). As a result, we obtain that

Theorem 4.7 There is always a solution of (4). Moreover, there is a solution that can be approximated starting with a one-e-place one-x-place seed box. □

Note that we do not claim that *every* solution of (4) can be approximated by starting with a one-e-place one-x-place seed box. Indeed, we will soon discuss a counterexample.

We could now stop our interest in the properties of the seed boxes and the two functions, $\mathsf{f_e}$ and $\mathsf{f_x}$. However, in view of the one-to-one relationship between the solutions of equation (4) and the ex seed boxes, captured in theorem 4.5, we are in a position to easily and fully characterise the set of all possible solutions of (4) using fixpoints of the two auxiliary functions, $\mathsf{f_e}$ and $\mathsf{f_x}$. More precisely, each pair of nonempty sets $\mathsf{P_e}, \mathsf{P_x} \subseteq \mathsf{P_{tree}}$ satisfying

$$\begin{array}{rrcl} & \mathsf{P_e} & = & \mathsf{f_e}(\mathsf{P_e}) \\ \text{and} & \mathsf{P_x} & = & \mathsf{f_x}(\mathsf{P_x}) \end{array} \tag{11}$$

uniquely identifies a solution of the recursive equation (4) whose entry and exit places are, respectively, $\mathsf{P_e}$ and $\mathsf{P_x}$. And conversely, if one takes any solution Σ of (4) then $\mathsf{P_e} = {}^\circ\Sigma$ and $\mathsf{P_x} = \Sigma^\circ$ do satisfy (11). This leads us to examining the two functions (and their fixpoints) more closely. Our first observation is that $(2^{\mathsf{P_{tree}}}, \subseteq)$ is a complete lattice with set union as 'join' and intersection as 'meet'. Moreover, $\mathsf{f_e}$ and $\mathsf{f_x}$ are monotonic and continuous mappings on this domain (see lemma B.4). It therefore follows that Tarski's theorem [10] can be applied and, as a result, $\mathsf{f_e}$ and $\mathsf{f_x}$ have unique minimal fixpoints, $\mathsf{min_e}$ and $\mathsf{min_x}$, and unique maximal fixpoints, $\mathsf{max_e}$ and $\mathsf{max_x}$, respectively. Since every pair of fixpoints for the two mappings satisfies (10), one might always be tempted to take $\mathsf{P_e} = \mathsf{min_e}$ and $\mathsf{P_x} = \mathsf{min_x}$, and treat the corresponding solution as *the* solution of (4) (note that by lemma B.2(2), such a solution would be the minimal solution in the sense of the inclusion order $\sqsubseteq$). Unfortunately, the minimal fixpoints may be empty which can make make them unsuitable for our purposes because boxes are ex-restricted. Fortunately, we may characterise the solutions of recursive equation (4) by looking at other fixpoints of $\mathsf{f_e}$ and $\mathsf{f_x}$ (cf. appendix B):

> *Unique solution.* If $\mathsf{fix_e}$ and $\mathsf{fix_x}$ are unique nonempty fixpoints of $\mathsf{f_e}$ and $\mathsf{f_x}$, respectively, then the recursive equation has a unique solution which can be derived from
>
> $$\mathsf{P_e} = \mathsf{fix_e} \text{ and } \mathsf{P_x} = \mathsf{fix_x}.$$

Note that a unique nonempty fixpoint exist, e.g., for $\mathsf{f_e}$ if there is exactly one fixpoint, or if there are exactly two fixpoints and the minimal fixpoint is empty (we give suitable examples in sections 4.5.2 and 4.5.4)

The minimal solution. If both $\mathsf{min_e}$ and $\mathsf{min_x}$ are nonempty then the recursive equation has the minimal solution (with respect to $\sqsubseteq$) derived from

$$\mathsf{P_e} = \mathsf{min_e} \text{ and } \mathsf{P_x} = \mathsf{min_x}.$$

The maximal solution. The recursive equation *always* has the maximal solution which can be derived from

$$\mathsf{P_e} = \mathsf{max_e} \text{ and } \mathsf{P_x} = \mathsf{max_x}.$$

Minimal solutions. For every pair of nonempty minimal fixpoints $\mathsf{fix_e}$ and $\mathsf{fix_x}$ of, respectively, $\mathsf{f_e}$ and $\mathsf{f_x}$, the recursive equation has a minimal solution which can be derived from

$$\mathsf{P_e} = \mathsf{fix_e} \text{ and } \mathsf{P_x} = \mathsf{fix_x}.$$

There may be several different (even non-isomorphic) minimal solutions (see section 4.5.3 for an example). However, we do not know yet whether there is always at least one minimal solution.

Countable solutions. If the nets appearing in the right-hand side of the recursive equation (4) are all countable, then there exists a solution which is countable as well. We can derive such a solution from $\mathsf{P_e}$ and $\mathsf{P_x}$ obtained by applying the limit construction in the auxiliary domain of trees $(2^{\mathsf{P_{tree}}}, \subseteq)$, and using the trees $\mathsf{spine_e}$ and $\mathsf{spine_x}$ defined earlier in this section,

$$\mathsf{P_e} = \bigcup_{i=0}^{\infty} \mathsf{f_e}^i(\{\mathsf{spine_e}\}) \text{ and } \mathsf{P_x} = \bigcup_{i=0}^{\infty} \mathsf{f_x}^i(\{\mathsf{spine_x}\}).$$

It is possible to give a simple static condition guaranteeing that both $\mathsf{f_e}$ and $\mathsf{f_x}$ are constant mappings and therefore trivially have unique fixpoints. The recursive equation (4) is *guarded* if for each transition v in Ω, $({}^\bullet v \cup v^\bullet) \cap ({}^\circ\Omega \cup \Omega^\circ) \neq \emptyset$ implies that Δ_v is a static box rather than a variable.

Theorem 4.8 If the recursive equation (4) is guarded then it has a unique solution. □

The reverse implication does not hold (see the example in section 4.5.4). A remark we now wish to make concerns the cardinality of boxes which solve recursive equations. Clearly, taking the limit of a chain of boxes will almost always produce an infinite box even if the equation was based on finite ones. However, we do not consider this to be a problem since the main purpose of this section was to provide theoretical foundations for solving recursive equations on boxes. In practical applications, initial fragments of the limit construction could be used to derive finite approximations of an infinite solution. We also expect that a more important application of our results will be in providing a basis for a high-level net solution obtained by folding the current infinite low level nets into finite high-level net representations.

4.5 Further examples of recursive definitions

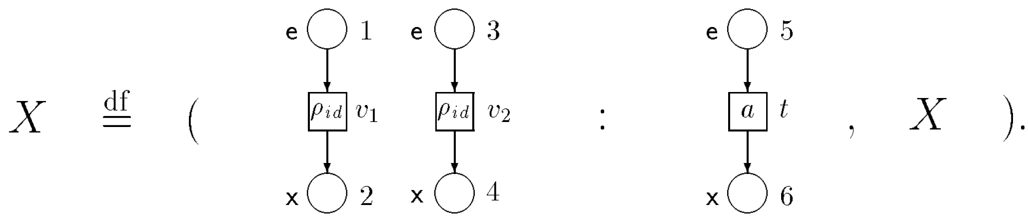
Before discussing a series of recursive definitions, let us return to the previous example (equation (6)). In this case, recursion is guarded and the two functions $\mathsf{f_e}$ and $\mathsf{f_x}$ are constant. Thus, they have unique fixpoints, namely the two singleton sets of trees

$$\begin{array}{rrcl} & \mathsf{P_e} & = & \{1 \lhd v_1 \lhd 5\} \\ \text{and} & \mathsf{P_x} & = & \{4 \lhd v_3 \lhd 8\}. \end{array}$$

Note that these are exactly the places of the box Σ^0 we have chosen earlier for this example (see (7)). Thus, also the net equation (6) has a unique solution, namely the box shown in figure 8.

4.5.1 Unbounded concurrent composition

The equations (11) for $\mathsf{P_e}$ and $\mathsf{P_x}$ may become more complex for unguarded recursion. This is illustrated by our next example which contains X unguardedly both with respect to the e-labelled front end and the x-labelled rear end of the operator box:



(12)

Informally, this corresponds to the recursive equation $X \stackrel{\mathrm{df}}{=} a \| X$, where $\|$ represents concurrent composition. We apply the preceding theory in order to derive a seed Σ^0 (with entry places $\mathsf{P_e}$ and exit places $\mathsf{P_x}$) to start the approximation of (12) and to characterise the set of all possible solutions.

Since transition v_2 of the operator box 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 reuse $\mathsf{P_e}$ (and $\mathsf{P_x}$, respectively) in the right-hand side of the tree equation. This leads to the following equations:

$$\begin{array}{rrcl} & \mathsf{P_e} & = & 1 \lhd (v_1 : \{5\}) \;\cup\; 3 \lhd (v_2 : \mathsf{P_e}) \\ \text{and} & \mathsf{P_x} & = & 2 \lhd (v_1 : \{6\}) \;\cup\; 4 \lhd (v_2 : \mathsf{P_x}). \end{array}$$

As before, there is a one-to-one correspondence between nonempty solutions of these equations and the solutions of (12), but now there is more than one solution. For example, the minimal solution for $\mathsf{P_e}$, $\mathsf{min_e}$, contains only finite trees while the maximal solution, $\mathsf{max_e}$, contains an infinite tree as well. There are no other solutions for $\mathsf{P_e}$. I.e.,

$$\begin{array}{rrcl} & \mathsf{min_e} & = & \{1 \lhd v_1 \lhd 5, 3 \lhd v_2 \lhd 1 \lhd v_1 \lhd 5, 3 \lhd v_2 \lhd 3 \lhd v_2 \lhd 1 \lhd v_1 \lhd 5, \ldots\} \\ \text{and} & \mathsf{max_e} & = & \mathsf{min_e} \cup \{3 \lhd v_2 \lhd 3 \lhd v_2 \lhd 3 \lhd v_2 \cdots\}. \end{array}$$

The equation for $\mathsf{P_x}$ also has two nonempty solutions, $\mathsf{min_x}$ and $\mathsf{max_x}$. Together, they determine four starting boxes

$$\begin{array}{rlllll} & \Sigma^{min} & \text{using} & \mathsf{min_e} & \text{and} & \mathsf{min_x} \\ & \Sigma^{max:min} & \text{using} & \mathsf{max_e} & \text{and} & \mathsf{min_x} \\ & \Sigma^{min:max} & \text{using} & \mathsf{min_e} & \text{and} & \mathsf{max_x} \\ \text{and} & \Sigma^{max} & \text{using} & \mathsf{max_e} & \text{and} & \mathsf{max_x} \end{array}$$

to begin the approximation of the solution of equation (12). Depending on which starting point is chosen, one of four different solutions of (12) is obtained. Two of them, the minimal and maximal ones, are shown in figure 10 (again we do not represent all name trees). It is not hard to check that both boxes are indeed solutions of equation (12) in the same strict sense as before. Note that the behaviour in terms of finite step sequences of the two solutions is the same. However, note also that if infinitely broad behaviours were allowed then the first solution would have a terminating behaviour while the second would not (since the initial token on its isolated e-place remains there and cannot be moved to the corresponding x-place). In that case, for instance, if $X \stackrel{\mathrm{df}}{=} a \| X$ is the first part of a sequential composition, then the second part of that composition could be executed using the first solution but not using the second solution.

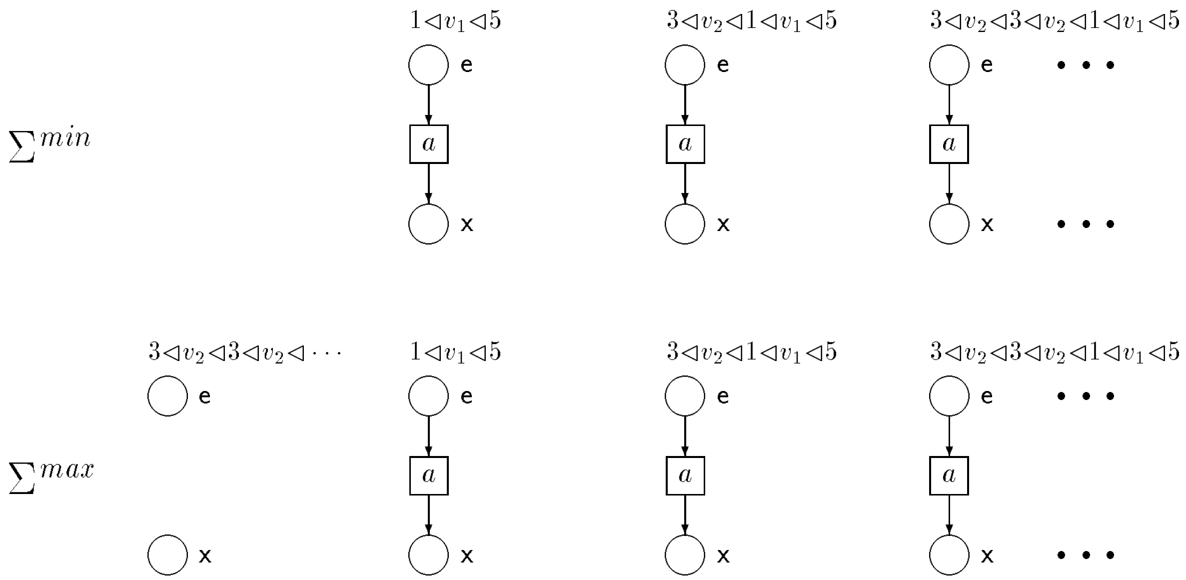
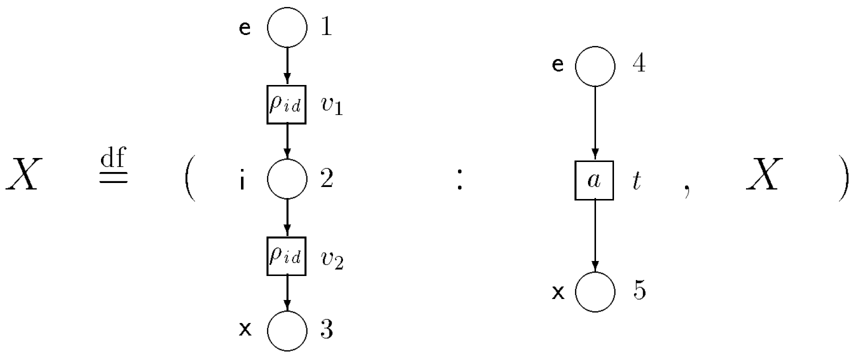


Figure 10: The minimal and maximal solutions of equation (12).

4.5.2 Rear-unguardedness

In the rest of this section, we illustrate the technique on four further examples, albeit not to the same degree of detail as before. First, consider



(13)

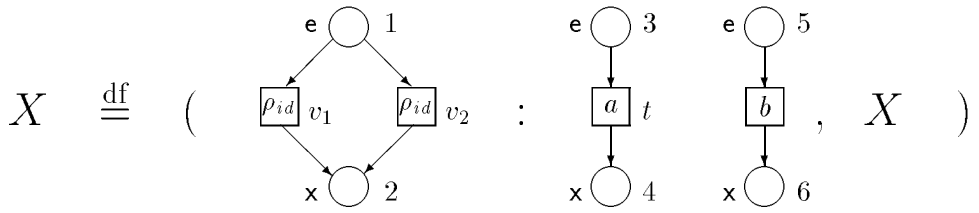
which, informally, corresponds to $X \stackrel{\mathrm{df}}{=} a;X$. This net equation leads to the following instance of the tree equations in (11):

$$\mathsf{P_e} = 1 \lhd (v_1 : \{4\})$$
$$\text{and} \quad \mathsf{P_x} = 3 \lhd (v_2 : \mathsf{P_x}).$$

Note that the only solution for $\mathsf{P_e}$ is the singleton set $\{1 \lhd v_1 \lhd 4\}$ while $\mathsf{P_x}$ has no unique solution, corresponding to the fact that the net equation is front-guarded (with respect to e) but not rear-guarded (with respect to x). The minimal solution for $\mathsf{P_x}$ is the empty set, $\mathsf{min_x} = \emptyset$, but the singleton set $\mathsf{max_x} = \{3 \lhd v_2 \lhd 3 \lhd v_2 \lhd 3 \lhd \cdots\}$ containing an infinite tree is also a solution. As we start the approximation with a box Σ^0, only the latter solution for $\mathsf{P_x}$ 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 8, except that the lower line is replaced by a single x-place, namely the only element of $\mathsf{P_x}$.

4.5.3 Concurrency within unbounded choice

As another example, consider



(14)

which informally corresponds to the equation $X \stackrel{\mathrm{df}}{=} (a\|b) \,\Box\, X$, where $\Box$ denotes choice. By the same rules as above, this leads to the following tree equations:

$$\begin{array}{rrcl} & \mathsf{P_e} & = & 1 \lhd (v_1 : \{3,5\}, v_2 : \mathsf{P_e}) \\ \text{and} & \mathsf{P_x} & = & 2 \lhd (v_1 : \{4,6\}, v_2 : \mathsf{P_x}). \end{array} \tag{15}$$

Note how the branching in the operator box leads to corresponding branchings in the trees. Equations (15) are interesting because, as it turns out, their maximal solutions have uncountably many trees. Starting the approximation with an uncountably large seed box corresponding to the maximal solution, leads to an uncountably large solution of the equation (14) (see also [4, 14]). Nevertheless, equation (15) also has countable solutions which can be obtained by starting the approximation with seeds Σ^0 that have only one e-place and only one x-place. To find such seeds, we construct spines as in construction 4.6. Figure 11 shows two spines for the set $\mathsf{P_e}$ of e-places of Σ^0. Let Σ^0 be the seed defined by the left-hand spine of figure 11 and its corresponding x-place, and let $\Sigma^{0'}$ be the seed defined by the right-hand spine of figure 11 and its corresponding x-place. Starting either with Σ^0 or with $\Sigma^{0'}$ yields a countable solution of (15). The two solutions have the same transitions (according to lemma B.2(1)), and, according to a later theorem, also the same behaviour, but they are not only incomparable in the sense of $\sqsubseteq$ but also non-isomorphic: the first - but not the second - contains a place connected to all a-labelled but to no b-labelled transition, and the second - but not the first - contains a place connected to all b-labelled but to no a-labelled transition. This is despite the fact that at every step in the approximation, isomorphic approximations are obtained (though every time with a different isomorphism). Note that in this example, the maximal solution cannot be obtained by starting with a countable seed (*a fortiori*, neither can it be obtained from a two-place seed).

Figure 11: Two spines for the e-places of equation (15).

This example is of further interest since it has uncountably many non-comparable minimal solutions, as we will discuss in the following. Intuitively speaking, every minimal solution corresponds to the infinite choice

$$(a\|b) \,\Box\, (a\|b) \,\Box\, (a\|b) \,\Box\, \ldots$$

where 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. Now, this infinite choice can be regarded as being a result of countably many binary choices since each a_i (or b_i) is in conflict with all the other transitions except for b_i (respectively, a_i). (Note that after one such conflict has been resolved the system becomes deterministic.) Each such conflict can be represented by a single entry place, say $e_{c_i c_j}$, where $i \neq j$

and $c_i \in \{a_i, b_i\}$, with arcs outgoing to both c_i and c_j. Thus countably many places are sufficient to ensure that each two transitions which are supposed to be in conflict are indeed in conflict. But this is not the end since after a pair of transitions a_k and b_k has been executed, the whole construct should be terminated as well. This means that tokens in all entry places $e_{c_i c_j}$ should be flushed as a result of the execution of a_k and b_k. And this can easily be achieved. We can designate, from each pair a_k, b_k one of the two transitions, denoted flush_k, and connect it to every entry place $e_{c_i c_j}$ such that $k \neq i$ and $k \neq j$. Together with the arcs from $e_{c_i c_j}$ to c_i and c_j these give a full connectivity of entry places. The same can be done by symmetry for the exit places. As a result, a countable number of places suffices to construct a solution. Note that there is no unique minimal solution since the choice of the flushing transitions is arbitrary.

$$\mathcal{T}_{n_1 n_2 n_3 \ldots} \quad = $$

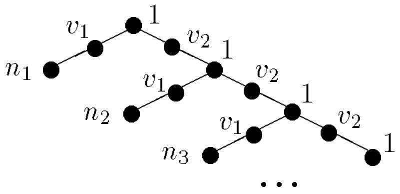


Figure 12: A tree for equation (15).

We now render the discussion of the different minimal solutions in a formal way. To this end we will consider a set of trees $\mathcal{T}$ of the form shown in figure 12, where $n_1, n_2, n_3, \ldots \in \{3, 5\}$. For every tree $\tau = \tau_{n_1 n_2 n_3 \ldots}$ we define

$$\tau^\star = \{\tau_{m_1 \ldots m_k n_i n_{i+1} n_{i+2} \ldots} \mid i \geq 1 \wedge m_1 \ldots m_k \in \{3, 5\}^*\}.$$

Using the set $\mathcal{T}$ we may now state a basic property of the set MNS of all minimal nonempty solutions of the first equation in (15). As the reader may check, we have

$$\mathsf{MNS} = \{\tau^\star \mid \tau \in \mathcal{T}\}$$

which answers one question, namely whether there are minimal nonempty solutions at all. This can be further strengthened to saying that MNS is uncountable and that its elements are disjoint sets of trees (proposition 4.9 below). Hence there are uncountably many incomparable minimal solutions for equation (14).

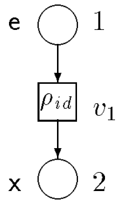
Proposition 4.9 MNS is an uncountable set of disjoint sets of trees.

Proof. Let $\asymp$ be a relation on trees in $\mathcal{T}$ such that $\tau_{n_1 n_2 \ldots} \asymp \tau_{m_1 m_2 \ldots}$ if $n_i n_{i+1} \ldots \neq m_j m_{j+1} \ldots$ for all $i, j \geq 1$. Define $\mathcal{R}$ as the set of all cliques of $\asymp$. It is easy to see that for all $\tau, \tau' \in \mathcal{T}$, either $\tau^\star \cap \tau'^\star = \emptyset$ and $\tau \asymp \tau'$, or $\tau^\star = \tau'^\star$ and $\tau \not\asymp \tau'$. Hence, to show the thesis it suffices to prove that $\mathcal{R}$ has an uncountable maximal element (with respect to set inclusion).

To conclude that $\mathcal{R}$ has a maximal element it suffices to show that if $\mathcal{Z} \subseteq \mathcal{R}$ is a linearly ordered set (with respect to set inclusion), then $\bigcup \mathcal{Z} \in \mathcal{R}$, which of course is true. Suppose that R is a maximal element in $\mathcal{R}$. We observe that for every $\tau \in \mathcal{T}$ there are only countably many $\tau' \in \mathcal{T}$ such that $\tau \not\asymp \tau'$. Hence, if R were countable, there would be $\tau \in \mathcal{T} \backslash R$ such that $R \cup \{\tau\} \in \mathcal{R}$ since $\mathcal{T}$ is uncountable. This, however, would contradict the maximality of R. ■ 4.9

4.5.4 Extreme unguardedness

As the next – somewhat unusual (because informally it corresponds to the recursive equation $X \stackrel{\mathrm{df}}{=} X$) – example, consider the following net equation:

$X \stackrel{\text{df}}{=} ($  $: \; X \;)$ (16)

By the rules of the game, (16) leads to the following tree equations for the entry and exit places of the starting box:

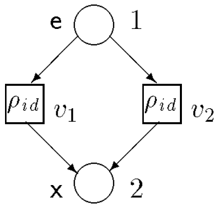
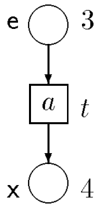
$$\mathsf{P_e} = 1 \lhd (v_1 : \mathsf{P_e})$$
$$\text{and} \quad \mathsf{P_x} = 2 \lhd (v_1 : \mathsf{P_x}).$$

We have $\mathsf{P_e} = \emptyset = \mathsf{P_x}$ amongst the set of possible solutions, but we also have nonempty singleton solutions, $\mathsf{P_e} = \{1 \lhd v_1 \lhd 1 \lhd v_1 \lhd 1 \lhd v_1 \lhd \ldots\}$ and $\mathsf{P_x} = \{2 \lhd v_1 \lhd 2 \lhd v_1 \lhd 2 \lhd v_1 \lhd \ldots\}$. In this case, the approximations never change anything, and the box solving the last net equation has exactly one e-place, one x-place, and no transitions. Note that despite the colloquial meaning of $X \stackrel{\text{df}}{=} X$, no box with a transition solves equation (16). This is a result of the finiteness of transition trees and the particular way of giving tree names in the refinement definition, by which the name of the refined transition is prepended to the tree of the refining transition(s). Thus, solving recursive equation by minimal boxes (with least behaviour) is already in-built in this definition.

We believe that this gives an intuitive semantics to the 'recursive call' $X \stackrel{\text{df}}{=} X$ because we do not model procedure entry and procedure exits as separate (silent) actions. Thus, $X \stackrel{\text{df}}{=} X$ essentially means that X is continually 'rewritten' by itself and effects no behaviour at all, be it silent or observable. The literature is split over the treatment of such unboundedly recursive calls. For instance, [1] agrees with our interpretation; other authors advocate that $X \stackrel{\text{df}}{=} X$ generates an infinite sequence of silent steps (modelling continuous entry to the procedure); and still other authors (such as [2]) consider $X \stackrel{\text{df}}{=} X$ to denote 'chaos', in the sense of generating all possible infinite sequences as their behaviour.

4.5.5 Empty nets and limit construction

Finally, consider the following equation:

$X \stackrel{\text{df}}{=} ($  $:$  $, \; X \;)$ (17)

which informally corresponds to $X \stackrel{\text{df}}{=} a \,\square\, X$. This equation can be dealt with in the same way as equation (15), and we omit the treatment. The reason why we give the example, is to show that the empty net is not suitable as a starting point of the approximation. Consider, to the contrary, that the empty net be taken as a seed box. Refinement works as usual, and we obtain as solution a net containing infinitely many isolated transitions labelled a. Thus, the solution allows an infinite a-step, which seems to be counterintuitive. If the approximation is instead done in the way we have described, then we get as solution a choice between infinitely many transitions labelled a, which is intuitively what is desired.

5 Operator-induced Expressions

In this section we introduce a general Petri net based form of expressions, which in section 6 is specialised to the PBC. Let OP be a fixed (possibly infinite) set of operator boxes. Recall that the transitions of OP are labelled by relabelling relations, that is, elements of the set Relab defined in section 2.1.1. We shall consider an algebra of process expressions over the signature:

$$\mathsf{Lab} \;\cup\; \{\mathsf{op}_\Omega \mid \Omega \in \mathsf{OP}\} \;\cup\; \{\overline{(.)}, \underline{(.)}\}$$

where Lab is the set of labels, and $\overline{(.)}$ and $\underline{(.)}$ are two unary operators whose intuitive meaning is similar to that of the corresponding operators defined for labelled nets in section 2.1. Although we use op_Ω, $\overline{(.)}$ and $\underline{(.)}$ to denote both mappings on boxes and process algebra operators, it will always be clear from the context which interpretation is meant. The labels in Lab are constants and each op_Ω in the process algebra is an n-ary operator, where n is the number of transitions in the operator box Ω. Throughout this section we shall further assume that the transition sets in the operator boxes of OP are disjoint and thus each transition name appearing in an operator box uniquely identifies that box. The proofs are included in appendix C.

5.1 Syntax

Recursion is introduced via a set of *process variables* var, which by definition comprises all process variables we might ever need. For every $K \in \mathsf{var}$ there is a single defining equation

$$K \stackrel{\mathrm{df}}{=} \mathsf{op}_\Omega(\mathbf{L})$$

such that Ω is an operator box in OP and $\mathbf{L}$ is an Ω–tuple of process variables of var and/or labels of Lab. We shall make use of four classes of expressions corresponding to previously introduced classes of plain boxes, namely the *static*, *entry*, *exit* and *dynamic* expressions, Expr^s, Expr^e, Expr^x and Expr^d; collectively referred to as *process expressions*, Expr. For technical reasons, we also need a larger set of *general* expressions, Expr^{all}. Their concrete syntax is given by:

Expr^s	E	$::=$	$a \mid K \mid \mathsf{op}_\Omega(\mathbf{E})$	(i)
Expr^e	F	$::=$	$\overline{E} \mid \mathsf{op}_\Omega(\mathbf{D})$	(ii)
Expr^x	G	$::=$	$\underline{E} \mid \mathsf{op}_\Omega(\mathbf{D})$	(iii)
Expr^d	H	$::=$	$F \mid G \mid \mathsf{op}_\Omega(\mathbf{D})$	(iv)
Expr^{all}	J	$::=$	$E \mid \overline{J} \mid \underline{J} \mid \mathsf{op}_\Omega(\mathbf{J})$	(v)

where $a \in \mathsf{Lab}$ is a label; $K \in \mathsf{var}$ is a process variable; $\Omega \in \mathsf{OP}$ is an operator box; and $\mathbf{E}$, $\mathbf{D}$ and $\mathbf{J}$ are Ω–tuples of expressions. These tuples have to satisfy some conditions determined by the domain of application of the net operator induced by Ω. More precisely, in (ii), (iii) and (iv), $\mathbf{D} \in \mathsf{Dom}_\Omega^\mu$ for some factorisation $\mu = (\phi, \psi)$ in fact_Ω which, by definition, means that

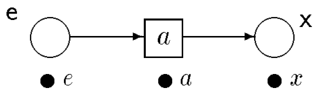
$$\begin{array}{rcl} \mathbf{D}_\phi & \subseteq & \mathsf{Expr}^d \\ \mathbf{D}_\psi & \subseteq & \mathsf{Expr}^x \\ \text{and}\quad \mathbf{D}\backslash\mathbf{D}_{\widetilde{\mu}} & \subseteq & \mathsf{Expr}^s. \end{array}$$

Furthermore, in (ii), μ is a factorisation of the initial marking ${}^\circ\Omega$ and $\mathbf{D}_\phi \subseteq \mathsf{Expr}^e$; and in (iii), μ is a factorisation of the terminal marking Ω°. The entire domain of application of an operator op_Ω is denoted by Dom_Ω and is defined as the set of all tuples of expressions $\mathbf{J}$ such that $\mathsf{op}_\Omega(\mathbf{J})$ is a process expression in Expr.

As in the case of boxes, it is convenient to have a notation for turning a dynamic expression into a corresponding static one; for each process expression J, $\lfloor J \rfloor$ is the static expression obtained by removing all the occurrences of $\overline{(.)}$ and $\underline{(.)}$. The operators $\overline{(.)}$, $\underline{(.)}$ and $\lfloor . \rfloor$ can be applied in the usual way (i.e., elementwise) to sets as well as tuples of expressions. The same will be true of the semantical mapping box defined in the next section.

5.2 Denotational semantics

We now introduce a mapping $\mathsf{box} : \mathsf{Expr} \to \mathsf{Box}^s \cup \mathsf{Box}^d$ transforming process expressions into boxes. It is defined by structural induction in the following way. We first define $\mathsf{box}(a)$ for each label $a \in \mathsf{Lab}$, as follows (note that the place and transition names in $\mathsf{box}(a)$ belong to $\mathsf{P_{root}}$ and $\mathsf{T_{root}}$):

$\mathsf{box}(a) \quad =$ 

We then define box for the process variables in var. With each defining equation $K \overset{\mathrm{df}}{=} \mathsf{op}_\Omega(\mathbf{L})$, we associate an equation on boxes

$$K \overset{\mathrm{df}}{=} (\Omega : \mathbf{\Delta})$$

where $\Delta_v = L_v$ if $L_v \in \mathsf{var}$ and $\Delta_v = \mathsf{box}(L_v)$ if $L_v \in \mathsf{Lab}$. Note that we treat process variables as though they were box variables. As we have shown in the previous section, the system of net equations we have just defined has at least one solution $\{\Sigma_K\}_{K \in \mathsf{var}}$ in static boxes. We now fix *any* such solution and define, for every process variable K, $\mathsf{box}(K) = \Sigma_K$. The translation is completed by homomorphically extending the box mapping to the remaining static and dynamic expressions, i.e., for all $\mathsf{op}_\Omega(\mathbf{J}) \in \mathsf{Expr}$ and $E \in \mathsf{Expr}^s$,

$$\begin{aligned} \mathsf{box}(\overline{E}) &= \overline{\mathsf{box}(E)} \\ \mathsf{box}(\underline{E}) &= \underline{\mathsf{box}(E)} \\ \text{and} \quad \mathsf{box}(\mathsf{op}_\Omega(\mathbf{J})) &= \mathsf{op}_\Omega(\mathsf{box}(\mathbf{J})). \end{aligned}$$

The correctness of the definition of the box mapping follows from lemma C.1(1).

5.3 Structurally similar expressions

To facilitate the introduction of the inaction rule in the operational semantics of process expressions, we define a structural similarity relation on process expressions, $\equiv$. It provides a partial identification of process expressions with the same denotational semantics. More precisely, $\equiv$ equates expressions which can be shown denotationally equal by purely syntactic means. An example of a pair of structurally similar expressions, taken from PBC, might be

$$\overline{a} \,\square\, b \quad \equiv \quad a \,\square\, \overline{b}. \tag{18}$$

A convenient way of specifying what expressions are $\equiv$–related is to define, for each operator box $\Omega \in \mathsf{OP}$, a number of 'equations' involving variables which can be instantiated with actual process expressions. For example, equation

$$\overline{E} \,\square\, F \quad = \quad E \,\square\, \overline{F}$$

when instantiated with $E = a$ and $F = b$, would yield the pair of structurally similar expressions in (18). In its full generality, $\equiv$ is more complex than the last example might suggest since we have to take into account process variables, as well as expressions with nested applications of operators.

The formal definition of structural similarity on process expressions takes into account the way in which boxes were associated with process variables, and possible factorisations of the operator boxes (since the relation we are going to define can be seen as a counterpart of the similarity relations $\equiv_\Omega$ on tuples of boxes). To start with, a relation $\equiv_{\mathsf{rec}}$ is defined on the basis of the recursive equations used to give meaning to process variables, each recursive equation $K \stackrel{\mathrm{df}}{=} \mathsf{op}_\Omega(\mathbf{L})$ contributing two pairs of structurally similar expressions,

$$\begin{array}{rcl} \overline{K} & \equiv_{\mathsf{rec}} & \overline{\mathsf{op}_\Omega(\mathbf{L})} \\ \text{and}\quad \underline{K} & \equiv_{\mathsf{rec}} & \underline{\mathsf{op}_\Omega(\mathbf{L})}. \end{array}$$

A more complicated part of the definition of $\equiv$ are the identifications which can be made about expressions of the form $\mathsf{op}_\Omega(\mathbf{J})$. Here, we basically follow the way in which the similarity relation on tuples of boxes was defined.

Let $\Omega \in \mathsf{OP}$ be an operator box and $\mathbf{V}$ be an Ω–tuple of distinct variables (place holders) ranging over expressions in Expr^{all} but different from the process variables var. For every factorisation $\mu = (\mu_1, \mu_2)$ of a set of places of Ω, let $\mathbf{V}^{[\mu]}$ be the Ω-tuple obtained from $\mathbf{V}$ by replacing V_v by $\overline{V_v}$, for $v \in \mu_1$, and V_v by $\underline{V_v}$, for $v \in \mu_2$. We define three types of equations. For every factorisation μ of ${}^\circ\Omega$, every factorisation κ of Ω°, and every pair of disjoint factorisations, η and ψ, of the same set of places which is contained in at least one marking reachable from ${}^\circ\Omega$, we have the following equations:

$$\begin{array}{rcll} \overline{\mathsf{op}_\Omega(\mathbf{V})} & = & \mathsf{op}_\Omega(\mathbf{V}^{[\mu]}) & \qquad\text{(E1)} \\ \underline{\mathsf{op}_\Omega(\mathbf{V})} & = & \mathsf{op}_\Omega(\mathbf{V}^{[\kappa]}) & \qquad\text{(E2)} \\ \mathsf{op}_\Omega(\mathbf{V}^{[\eta]}) & = & \mathsf{op}_\Omega(\mathbf{V}^{[\psi]}) & \qquad\text{(E3)} \end{array}$$

Thus (E1)–(E3) define a scheme which actually comprises more than three equations, unless Ω is very simple. Now, each equation in (E1)–(E3) induces a relation on expressions, which is obtained by instantiating the variables in $\mathbf{V}$ with arbitrary expressions in Expr^{all}. The union of all such relations, for all operator boxes in OP, will be denoted by $\equiv_{\mathsf{OP}}$.

We then define $\equiv$ to be the least equivalence relation over expressions Expr^{all} (that is, over all expressions, including those for which we did not provide a denotational semantics) which contains the two auxiliary relations we just introduced, i.e., $\equiv_{\mathsf{rec}}$ and $\equiv_{\mathsf{OP}}$, and such that for every $\Omega \in \mathsf{OP}$ and all Ω-tuples $\mathbf{E}$ and $\mathbf{F}$ of expressions in Expr^{all}, if $E_v \equiv F_v$ holds for every $v \in T_\Omega$, then $\mathsf{op}_\Omega(\mathbf{E}) \equiv \mathsf{op}_\Omega(\mathbf{F})$. Results proved in appendix C clarify the relationship between two equivalences on process expressions, namely the structural equivalence (defined by $\equiv$) and denotational equivalence (induced by the box mapping), as well as the relationship between two similarity relations, $\equiv_\Omega$ and $\equiv$, defined for tuples of boxes and process expressions, respectively.

There is a subtle point concerning the use of the expansion rule for recursion which is captured by the auxiliary relation $\equiv_{\mathsf{rec}}$. Since $\mathsf{box}(K) = \mathsf{box}(\mathsf{op}_\Omega(\mathbf{L}))$, it might seem that we could have defined $K \equiv_{\mathsf{rec}} \mathsf{op}_\Omega(\mathbf{L})$. However, this would be inconsistent with the usual treatment of recursion in other process algebras[7] and, furthermore, allowing it would yield too many identifications (although without invalidating any of the results formulated later in this section). For instance, for any defining equation of the form $K \stackrel{\mathrm{df}}{=} \mathsf{op}_\Omega(a, K, b)$ we would have

$$\overline{K} \equiv \overline{\mathsf{op}_\Omega(a, K, b)} \equiv \overline{\mathsf{op}_\Omega(a, \mathsf{op}_\Omega(a, K, b), b)} \equiv \ldots$$

while only the first equivalence, but not the second, may be derived in the present setting if Ω is the operator box in (6). As it stands, the equivalence classes of $\equiv$ are finite, provided that recursive definitions satisfy simple structural properties. For example, if all defining equations are guarded (in the sense that the equations on nets they induce are guarded) then the equivalence classes of $\equiv$ are finite, but not vice versa.

[7] By this we mean that recursion is not unfolded unnecessarily. For example, in CCS it is possible to have a transition $\mu X.a.X \xrightarrow{a} \mu X.a.X$, but not $\mu X.a.X \xrightarrow{a} a.a.a.a.\mu X.a.X$.

5.4 Operational semantics, soundness and completeness

Let $\mathsf{Proc}_{\mathsf{tree}}$ be the set of all transition trees in the boxes derived from process expressions. Each such tree has a unique label, $\mathsf{lab}(t)$, in the boxes associated with process expressions. More precisely, if t has only one node then it is the only transition of a unique $\mathsf{box}(a)$, for some label $a \in \mathsf{Lab}$, and then $\mathsf{lab}(t) = a$. A t with more than one node has the form $t = (v, a) \lhd Q$ and then, by the definition of net refinement which underpins the semantical mapping box, $\mathsf{lab}(t) = a$. We now define an inference system allowing one to derive moves of the form $D \xrightarrow{U} H$ where D and H are process expressions and U is a finite set of trees in $\mathsf{Proc}_{\mathsf{tree}}$. The idea here is that U is a valid step for the boxes associated with D and H.

The operational semantics of process expressions is defined in three stages, as follows. For every $a \in \mathsf{Lab}$ we have the axiom:

$$\boxed{\overline{a} \xrightarrow{\{t\}} \underline{a}}$$

where t is the only transition in $\mathsf{box}(a)$. For each operator box Ω in OP we have the inference rule:

$$\boxed{\frac{D_v \xrightarrow{U_v^1 \uplus \cdots \uplus U_v^{k_v}} H_v \;\;,\;\; (\mathsf{lab}\langle U_v^i \rangle, a_v^i) \in \lambda_\Omega(v)}{\mathsf{op}_\Omega(\mathbf{D}) \xrightarrow{\bigcup_{v \in T_\Omega} \{ (v,a_v^1) \lhd U_v^1, \ldots, (v,a_v^{k_v}) \lhd U_v^{k_v} \}} \mathsf{op}_\Omega(\mathbf{H})} \quad (\text{for } v \in T \text{ and } i \leq k_v)}$$

where each k_v is a natural number and $\mathsf{op}_\Omega(\mathbf{D})$ is an expression in Expr^{all}. Finally, we introduce the inaction rule:

$$\boxed{\frac{D \equiv H}{D \xrightarrow{\emptyset} H}}$$

where D is an expression in Expr^{all}. As in the case of the SOS rule for boxes, we now have two theorems which state soundness and completeness of the inference system with respect to the standard step sequence semantics of nets.

Theorem 5.1 (Soundness of the inference rules)
If D is a process expression and $D \xrightarrow{U} H$ then H is a process expression and $\mathsf{box}(D)\ [U\rangle\ \mathsf{box}(H)$.

$\square$

Theorem 5.2 (Completeness of the inference rules)
If D is a process expression and $\mathsf{box}(D)\ [U\rangle\ \Sigma$ then there are process expressions H and G such that $\mathsf{box}(G) = \Sigma$ and

$$D \xrightarrow{\emptyset} H \xrightarrow{U} G.$$

$\square$

Note that the first part of theorem 5.1 means that both the structural similarity relation and the SOS derivation relation are closed in the domain of process expressions, i.e., if D is a process expression and $D \equiv H$ or $D \xrightarrow{U} H$ then H is a process expression too. Moreover, directly from theorems 5.1 and 5.2 and the fact that we allowed an *arbitrary* solution of the system of recursive equations on nets to provide denotational semantics for process variables, we obtain that all solutions of a recursive equation on nets have the same step sequence semantics.

Corollary 5.3 If Σ and Θ are two solutions of the recursive equation on nets (4), then there is a relation $\mathcal{B} \subseteq [\overline{\Sigma}\rangle \times [\overline{\Theta}\rangle$ such that $(\overline{\Sigma}, \overline{\Theta}) \in \mathcal{B}$ and if $(\Sigma_0, \Theta_0) \in \mathcal{B}$ then the following hold:

- If $\Sigma_0 \, [U\rangle \, \Sigma_1$ then there is Θ_1 such that $\Theta_0 \, [U\rangle \, \Theta_1$ and $(\Sigma_1, \Theta_1) \in \mathcal{B}$.
- If $\Theta_0 \, [U\rangle \, \Theta_1$ then there is Σ_1 such that $\Sigma_0 \, [U\rangle \, \Sigma_1$ and $(\Sigma_1, \Theta_1) \in \mathcal{B}$. □

The relation $\mathcal{B}$ is a variant of the *strong bisimulation* of [28], and thus what the corollary says is that different solutions of a recursive equation on nets have strongly equivalent step-based reachability graphs. Recall (from the example $X \stackrel{\mathrm{df}}{=} a \| X$ discussed in section 4.5.1) that – in the presence of sequential composition – corollary 5.3 fails if infinite steps are taken into account.

5.5 Some derived results

In this section we will use the already established soundness and completeness results, to show that the same is true of three variations of the operational semantics, which are often considered in the literature.

5.5.1 Step sequence semantics

We start by lifting the soundness and completeness results to the level of step sequence semantics. Suppose that $D_1, \ldots, D_n$ $(n \geq 1)$ are process expressions and $U_1, \ldots, U_{n-1}$ are finite sets of transition trees such that

$$D_1 \xrightarrow{U_1} D_2 \xrightarrow{U_2} \cdots \xrightarrow{U_{n-1}} D_n.$$

Then σ – the maximal subsequence of $U_1 \ldots U_{n-1}$ comprising only nonempty sets – is a *step sequence* leading from D_1 to D_n. We will denote this by $D_1 \xrightarrow{\sigma} D_n$. The next result follows immediately from theorems 5.1 and 5.2.

Theorem 5.4 (Soundness and completeness of step sequence semantics)
Let D be a dynamic expression and σ be a finite step sequence of non-empty subsets of $\mathsf{Proc}_{\mathsf{tree}}$.

(1) If $D \xrightarrow{\sigma} H$ then H is a process expression and $\mathsf{box}(D) \, [\sigma\rangle \, \mathsf{box}(H)$.

(2) If $\mathsf{box}(D) \, [\sigma\rangle \, \Sigma$ then there is a process expression H such that $D \xrightarrow{\sigma} H$ and $\mathsf{box}(H) = \Sigma$.

□

5.5.2 Partial order semantics

The completeness result obtained for the step sequence semantics can be extended to the level of partial order executions since the trees $\mathsf{Proc}_{\mathsf{tree}}$ contain enough information to reconstruct the independence relation on transitions.

Proposition 5.5 There is a relation $\mathsf{ind} \subseteq \mathsf{Proc}_{\mathsf{tree}} \times \mathsf{Proc}_{\mathsf{tree}}$ such that if D is a process expression and t, u are two distinct transitions in $\mathsf{box}(D)$, then $({}^{\bullet}t \cup t^{\bullet}) \cap ({}^{\bullet}u \cup u^{\bullet}) = \emptyset$ if and only if $(t, u) \in \mathsf{ind}$. □

We now can consider two ways of introducing partial order behaviours into the model. The first one is, as always, standard to Petri nets. For two dynamic boxes, Σ and Θ, we denote $\Sigma \, [\pi\rangle \, \Theta$ if there is a finite occurrence net starting from the marking of Σ and ending at the marking of Θ which generates a labelled partial order of transition occurrences π (see [6] for details). The second partial order semantics is introduced for dynamic expressions, in the following way. Suppose σ is a step sequence leading from a dynamic expression expression D to H. We then can derive from σ and ind a partial order π labelled with transition trees $\mathsf{Proc}_{\mathsf{tree}}$ using the standard construction due to Mazurkiewicz [27]. We denote this by $D \xrightarrow{\pi} H$.

Theorem 5.6 (Soundness and completeness of partial order semantics)
Let D be a dynamic expression.

(1) If $D \xrightarrow{\pi} H$ then H is a process expression and $\mathsf{box}(D)\ [\pi\rangle\ \mathsf{box}(H)$.

(2) If $\mathsf{box}(D)\ [\pi\rangle\ \Sigma$ then there is a process expression H such that $D \xrightarrow{\pi} H$ and $\mathsf{box}(H) = \Sigma$.

Proof. Follows from propositions 3.1 and 5.5 and theorem 5.4. We also need the consistency result about the partial order semantics and step sequence semantics for safe nets proved in [3]. ■ 5.6

And, again as an immediate consequence of the last result, we obtain that different solutions of the recursive equation on nets (4) have the same partial order behaviour (one only needs to replace U by π in the formulation of corollary 5.3). And again, this is no longer true if infinite (to be precise, infinitely broad) partial order semantics is taken into account.

5.5.3 Label based inference rules

The step sequences based on trees in $\mathsf{Proc}_{\mathsf{tree}}$ are rather powerful; in particular, they carry enough information to retrieve the partial order semantics of the corresponding boxes. However, it may sometimes be sufficient to be interested only in the labels of the executed transitions, more like in the usual style of process algebras. Such a treatment can be accommodated within the scheme developed so far. First of all, we retain the structural similarity relation $\equiv$ on process expressions without any change. Next, we define moves of the form $D \xrightarrow{\gamma} H$, where D and H are as before but γ is now a finite multiset of labels, $\gamma \in \mathsf{mult}(\mathsf{Lab})$. We also keep the inaction inference rule (with $\emptyset$ denoting the empty multiset of labels), but the axiom for $a \in \mathsf{Lab}$ becomes:

$$\boxed{\overline{a} \xrightarrow{\langle a \rangle} \underline{a}}$$

and, for each operator box Ω in OP, the new rule is:

$$\boxed{\frac{D_v \xrightarrow{\gamma_v^1 + \cdots + \gamma_v^{k_v}} H_v \ \ , \ \ (\gamma_v^{k_v}, a_v^i) \in \lambda_\Omega(v)}{\mathsf{op}_\Omega(\mathbf{D}) \xrightarrow{\bigplus_{v \in T_\Omega} \langle a_v^1, \ldots, a_v^{k_v} \rangle} \mathsf{op}_\Omega(\mathbf{H})} \quad (\text{for } v \in T \text{ and } i \leq k_v)}$$

It is not difficult to prove, by induction on the structure of D, that $D \xrightarrow{\gamma} H$ if and only if there is a set of transition trees $U \subseteq \mathsf{Proc}_{\mathsf{tree}}$ such that $D \xrightarrow{U} H$ and $\mathsf{lab}\langle U \rangle = \gamma$. Hence the results and notions concerning $\xrightarrow{U}$ proved in sections 5.4 and 5.5.1 extend, after obvious modifications, to the relation $\xrightarrow{\gamma}$. In particular, a counterpart of theorem 5.4 holds.

5.6 Why factorisations are necessary for completeness

We now return to the last example in section 3.2 (see also figure 6). To see that the N-box does not have a complete SOS semantics in the style outlined above, consider defining a quaternary operator $\mathsf{op}_N(E_1, E_2, E_3, E_4)$ with four arguments (where $E_1 - E_4$ are supposed to describe the same flow of control as $v_1 - v_4$ of the N-box), and the following set of equations:

$$\begin{array}{rcl}
\overline{\mathsf{op}_N(E_1, E_2, E_3, E_4)} & = & \mathsf{op}_N(\overline{E_1}, \overline{E_2}, E_3, E_4) \\
\mathsf{op}_N(\underline{E_1}, \overline{E_2}, E_3, E_4) & = & \mathsf{op}_N(E_1, \overline{E_2}, \overline{E_3}, E_4) \\
\mathsf{op}_N(\underline{E_1}, \underline{E_2}, E_3, E_4) & = & \mathsf{op}_N(E_1, E_2, \overline{E_3}, \overline{E_4}) \\
\mathsf{op}_N(E_1, \underline{E_2}, \underline{E_3}, E_4) & = & \mathsf{op}_N(E_1, E_2, \underline{E_3}, \overline{E_4}) \\
\mathsf{op}_N(E_1, E_2, \underline{E_3}, \underline{E_4}) & = & \underline{\mathsf{op}_N(E_1, E_2, E_3, E_4)}
\end{array}$$

which, at first sight, seems to describe the control flow within the N-box. However, consider the expression

$$E \;=\; \mathsf{op}_N(\, a \,,\, (b; b') \,,\, (c; c') \,,\, d \,).$$

Then $box(\overline{E})$ can execute the step sequence $\langle a\rangle\langle b\rangle\langle c\rangle\langle b'\rangle\langle d\rangle$, leading to the terminal marking, but this cannot be derived using the above set of equations (essentially, the third of the equations can never be applied in this case).

6 The Box Algebra

The theory expounded in the previous sections is very general. Essentially, the control connectives of a process algebra were described by operator boxes and communication-based interface connectives were described by relabellings. Thus, we may have process algebras with as many control-flow operators as there are operator boxes and as many communication-based operators as there are relabellings. In this section we specialise the theory to a particular subset of them, namely the box algebra (PBC) first defined in [5]. We also give several examples, that is to say, instances of formulae defined in the previous sections.

6.1 Syntax

The PBC is based on a particular set of labels $\mathsf{Lab}_{\mathrm{PBC}}$ defined thus. Let A be a countable alphabet of (visible) *action particles* and let $\widehat{\ }: \mathsf{A} \to \mathsf{A}$ be a bijection (called *conjugation*) which satisfies, for all $a \in \mathsf{A}$, $\widehat{a} \neq a$ and $\widehat{\widehat{a}} = a$. The labels, and hence the basic actions, of the PBC are the finite multisets over A, $\mathsf{Lab}_{\mathrm{PBC}} = \mathsf{mult}(\mathsf{A})$. The elements of $\mathsf{Lab}_{\mathrm{PBC}}$ will now be denoted by $\alpha, \alpha_1, \ldots$ to distinguish them from the general labels, and the lower case letters, $a, b, \ldots$ will be used to denote action particles. For example, the label $\langle \widehat{a}, b\rangle \in \mathsf{Lab}_{\mathrm{PBC}}$ is interpreted as specifying a simultaneous execution of two visible action particles $\widehat{a}$ and b. The empty multiset $\emptyset \in \mathsf{Lab}_{\mathrm{PBC}}$ should be interpreted as specifying the execution of an action containing no visible action particles. This corresponds to the *silent* or *internal* action denoted by τ in the terminology of CCS [28]. An expression of the basic PBC considered in this paper, by definition, is a word of the following syntax[8]

$$E \;::=\; \alpha \mid E; E \mid E \,\square\, E \mid E \| E \mid E[f] \mid E \;\mathsf{sy}\; a \mid E \;\mathsf{rs}\; a \mid K. \tag{19}$$

An expression of the form α is called a *basic action* (or *multiaction*). The connectives ; (sequence), $\square$ (choice) and $\|$ (parallelism) are control flow connectives. The operators $[f]$ (basic relabelling), $\mathsf{sy}\; a$ (synchronisation) and $\mathsf{rs}\; a$ (restriction) are called activity interface operators. It is assumed that f is a function from A to A preserving conjugates; i.e., for all $a \in \mathsf{A}$, $f(\widehat{a}) = \widehat{f(a)}$. The last item in the syntax is a process variable K. It is assumed that for each such K there is exactly one defining equation $K \stackrel{\mathrm{df}}{=} expr$ where the syntax for $expr$ is:

$$expr \;::=\; L; L \mid L \,\square\, L \mid L \| L \mid L[f] \mid L \;\mathsf{sy}\; a \mid L \;\mathsf{rs}\; a \tag{20}$$

[8] We use the infix notation for the binary, and postfix notation for unary, operators. Also, instead of writing, e.g., $E \,\mathsf{op}_{\Omega_\|} F$ we prefer to use $E \| F$.

and the syntax for L is given by:

$$L \quad ::= \quad E \mid expr. \tag{21}$$

Note that (19) only defines static PBC expressions. Dynamic expressions will be introduced later on. In the basic PBC there are three transition interface modification operators, namely the basic relabelling $[f]$ carried out according to a mapping $f\colon \mathsf{A} \to \mathsf{A}$ defined for action particles, restriction $\mathsf{rs}\ a$ on labels containing a given action particle a or its conjugate $\widehat{a}$, and synchronisation $\mathsf{sy}\ a$ using a given action particle a and its conjugate $\widehat{a}$. To describe these three operations we introduce three relabellings over $\mathsf{Lab}_{\mathrm{PBC}}$ denoted, respectively, by $\rho_{[f]}$, $\rho_{\ \mathrm{rs}\ a}$ and $\rho_{\ \mathrm{sy}\ a}$.

Basic relabelling (renaming). To define $\rho_{[f]}$, f is lifted to a function on $\mathsf{Lab}_{\mathrm{PBC}}$ by $f(\emptyset) = \emptyset$ and $f(\alpha + \langle a \rangle) = f(\alpha) + \langle f(a) \rangle$. Then

$$\rho_{[f]} = \{(\langle \alpha \rangle, f(\alpha)) \mid \alpha \in \mathsf{Lab}_{\mathrm{PBC}}\}.$$

Restriction. To define $\rho_{\ \mathrm{rs}\ a}$, we remove part of the identity relabelling touching a or $\widehat{a}$, viz.:

$$\rho_{\ \mathrm{rs}\ a} = \{(\langle \alpha \rangle, \alpha) \mid \alpha \in \mathsf{mult}(\mathsf{A} \setminus \{a, \widehat{a}\})\}.$$

Synchronisation. The relabelling describing synchronisation, $\rho_{\ \mathrm{sy}\ a}$, is defined as the set of all pairs $(\langle \alpha_1, \ldots, \alpha_k \rangle, \alpha)$, where $k \geq 1$, such that

$$\sum_{i=1}^{k} \alpha_i(a) \geq k{-}1 \quad \text{and} \quad \sum_{i=1}^{k} \alpha_i(\widehat{a}) \geq k{-}1,$$

and if $k \geq 2$ then, for all $i \leq k$, $\alpha_i(a) + \alpha_i(\widehat{a}) \geq 1$. Moreover,

$$\alpha = (\alpha_1 + \cdots + \alpha_k) - ((k{-}1) \cdot \langle a, \widehat{a} \rangle).$$

For example, $(\langle \langle a, b \rangle, \langle \widehat{a}, \widehat{a} \rangle, \langle a, c \rangle \rangle, \langle b, c \rangle) \in \rho_{\ \mathrm{sy}\ a}$.

The last formula captures the synchronisation discipline introduced in [5]. It is explained more fully in that paper, and also in [7]. Here, it was sufficient to show that it can be encapsulated in a relabelling relation which belongs to Relab. The same is true for other synchronisation disciplines considered in [13] and [16].

6.2 Relabelling in other process algebras

Varying the set of labels Lab, and correspondingly the set of relabellings Relab, may serve to model other standard operations used in process algebras. General restriction on a set of labels, $L \subseteq \mathsf{Lab}$, can be modelled by

$$\rho_L = \{(\langle l \rangle, l) \mid l \in \mathsf{Lab} \setminus L\}.$$

Relabelling induced by a mapping on labels, $h\colon \mathsf{Lab} \to \mathsf{Lab}$, is expressed by

$$\rho_h = \{(\langle l \rangle, h(l)) \mid l \in \mathsf{Lab}\}.$$

CCS-like synchronisation can be modelled thus. Let $\mathsf{Lab} = Act \cup \{\tau\}$ be the set of CCS labels [28] and $\widehat{\ }\colon Act \to Act$ be a bijection on Act satisfying $\widehat{\widehat{a}} = a$ and $\widehat{a} \neq a$, for all $a \in Act$. Then CCS-like synchronisation is captured by

$$\rho_{CCS} = \rho_{id} \cup \{(\langle a, \widehat{a} \rangle, \tau) \mid a \in Act\}.$$

The first two relabellings defined for the PBC can be seen as specific instances of those defined above, e.g., the PBC restriction is a special case of general restriction by putting

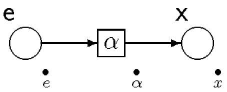
$$L \ = \ \mathsf{mult}(\mathsf{A}) \setminus \mathsf{mult}(\mathsf{A} \setminus \{a, \widehat{a}\}).$$

Moreover, CCS synchronisation becomes a special case of PBC synchronisation by considering multi-sets of cardinality 0 or 1 and equating τ with $\emptyset$.

6.3 Denotational semantics

First we give the box semantics of the recursion-free part of the basic PBC. To this end, we use as the set OP (see section 5) the six operator boxes[9] shown in figure 13, which correspond to every syntactic alternative on the right-hand side of the syntax (19) except multiaction and process variable. Note that all boxes satisfy the properties of operator boxes.[10] The full net semantics of the recursion-free static PBC is given by the function $\mathsf{box}(E)$, yielding a box for every expression E, defined as follows:

$\mathsf{box}(\alpha) \quad =$



$$
\begin{array}{lcl}
\mathsf{box}(E;F) & = & \mathsf{op}_{\Omega_;}(\mathsf{box}(E),\mathsf{box}(F)) \\
\mathsf{box}(E \,\Box\, F) & = & \mathsf{op}_{\Omega_\Box}(\mathsf{box}(E),\mathsf{box}(F)) \\
\mathsf{box}(E\|F) & = & \mathsf{op}_{\Omega_\|}(\mathsf{box}(E),\mathsf{box}(F)) \\
\mathsf{box}(E[f]) & = & \mathsf{op}_{\Omega_{[f]}}(\mathsf{box}(E)) \\
\mathsf{box}(E \;\mathsf{rs}\; a) & = & \mathsf{op}_{\Omega\ \mathrm{rs}\ a}(\mathsf{box}(E)) \\
\mathsf{box}(E \;\mathsf{sy}\; a) & = & \mathsf{op}_{\Omega\ \mathrm{sy}\ a}(\mathsf{box}(E)).
\end{array}
$$

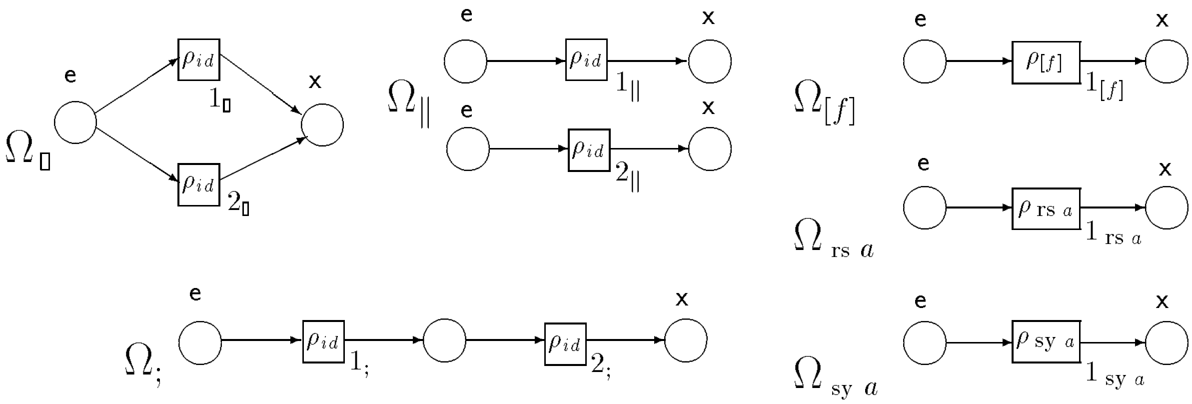


Figure 13: Operator boxes for the PBC.

The reader may observe that the defining equation $K \stackrel{\mathrm{df}}{=} expr$ generated by the syntax above may not be in the format required by our general semantics developed in the previous section. In other words, the syntax given in (20) and (21) is too liberal. To make it adhere to the one required by the general scheme it is sufficient to restrict the syntax for L to:

$$L \quad ::= \quad \alpha \mid K \tag{22}$$

which prevents the nesting of operators in equations defining process variables. Clearly, any defining equation $K \stackrel{\mathrm{df}}{=} expr$ derived using syntax (21) can be translated into a system of defining equations of the form $K_i \stackrel{\mathrm{df}}{=} expr_i$ derived using (22). Rather than providing this translation in general, we consider an example. Take the following defining equation for a process variable K:

$$K \stackrel{\mathrm{df}}{=} (a\|(b \,\Box\, (K;b)));(K \,\Box\, (b\|K)).$$

[9]Strictly speaking, the number of operators is infinite since, for example, we need a separate box for each of the operators $\mathsf{rs}\, a$, where $a \in \mathsf{A}$.

[10]In fact, every disjoint union of connected S-system boxes, having a single e place and a single x place each, does.

It can be unravelled into six equations, as follows:

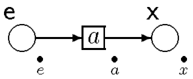
$$
\begin{array}{lllllllll}
K & \stackrel{\text{df}}{=} & K_1; K_2 & K_1 & \stackrel{\text{df}}{=} & a \| K_3 & K_2 & \stackrel{\text{df}}{=} & K \,\Box\, K_4 \\
K_3 & \stackrel{\text{df}}{=} & b \,\Box\, K_5 & K_4 & \stackrel{\text{df}}{=} & b \| K & K_5 & \stackrel{\text{df}}{=} & K; b.
\end{array}
$$

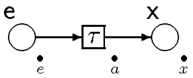
Every such equation can in turn be translated directly into a box equation of the form $K \stackrel{\text{df}}{=} (\Omega : \Delta_1, \Delta_2)$, where Ω is one of the operator boxes of figure 13. In this way, a net equation system can be associated in a canonical way with any given set of recursive definitions. This system can then be solved using the machinery described in section 4. In fact, the examples considered in section 4 correspond, in that order, to the following recursive definitions in PBC:

$$
\begin{array}{lll}
 & K = a; L \ , \ L = K; b & \text{equation (6)} \\
 & K = a \| K & \text{equation (12)} \\
 & K = a; K & \text{equation (13)} \\
 & K = (a \| b) \,\Box\, K & \text{equation (14)} \\
 & K = K[\rho_{id}] & \text{equation (16)} \\
\text{and} & K = a \,\Box\, K & \text{equation (17).}
\end{array}
$$

6.4 Box semantics of other process algebras

Continuing the considerations of section 6.2, we give the box semantics of CCS in abridged form by the following table (which implicitly also defines the syntax we use to write CCS expressions E):

$\mathsf{box}(a) \quad = $ 

$\mathsf{box}(\tau) \quad = $ 

$$
\begin{array}{lll}
\mathsf{box}(a.E) & = & \mathsf{op}_{\Omega_;}(\mathsf{box}(a), \mathsf{box}(E)) \\
\mathsf{box}(E+F) & = & \mathsf{op}_{\Omega_\Box}(\mathsf{box}(E), \mathsf{box}(F)) \\
\mathsf{box}(E|F) & = & \mathsf{op}_{\Omega_{CCS}}(\mathsf{op}_{\Omega_\|}(\mathsf{box}(E), \mathsf{box}(F))) \\
\mathsf{box}(E[h]) & = & \mathsf{op}_{\Omega_h}(\mathsf{box}(E)) \\
\mathsf{box}(E \backslash a) & = & \mathsf{op}_{\Omega_a}(\mathsf{box}(E)).
\end{array}
$$

The extra definitions that need to be made concern Ω_{CCS}, Ω_h and Ω_a. All three are two-place one-transition operator boxes, as shown in the last column of figure 13, which use respectively the relabellings ρ_{CCS}, ρ_h and $\rho_{\{a\}}$, as defined in section 6.2.

For CCS expressions containing both a and $\widehat{a}$ in one sequential component (such as, for instance, $((a.\widehat{a})|b)$ or $((a+\widehat{a})|b)$), the above net semantics creates additional τ-labelled transitions by synchronising two transitions within that sequential component. This translation is slightly non-standard, but such additional transitions are always dead in the initial marking and hence do not affect the behaviour. Note that one could avoid generating these transitions by adapting a more complicated scheme of modelling synchronisation between two subcomponents used below in the case of TCSP. Recursion can be treated exactly as in the previous section.

It is possible to provide a similar treatment for other models of concurrent behaviour, such as TCSP [20] or COSY [22]. The translation is similar to that given above with one exception, namely the concurrent composition construct has to be dealt with differently.

The TCSP concurrent composition operator, $\|_A$ (meaning 'synchronise on actions in the set A'), can be modelled – in abbreviated form – thus:

$$
\mathsf{box}(E \|_A F) = \mathsf{op}_{\Omega_{TCSP_A}}(\mathsf{op}_{\Omega_\|}(\mathsf{op}_{\Omega_{\times 1}}(\mathsf{box}(E)), \mathsf{op}_{\Omega_{\times 2}}(\mathsf{box}(F)))).
$$

The $\Omega_{\times i}$ $(i = 1, 2)$ and Ω_{TCSP_A} are simple boxes like $\Omega_{[f]}$ with the following relabellings annotating the only transition:

$$\begin{aligned} \rho_{\times i} &= \{(\langle a\rangle, (a,i)) \mid a \in A\} \cup \rho_A \quad (i = 1, 2) \\ \text{and} \quad \rho_{TCSP_A} &= \{(\langle (a,1),(a,2)\rangle, a) \mid a \in A\} \cup \rho_A \end{aligned}$$

where it is assumed that the set of labels is temporarily extended by the set $\mathsf{Lab} \times \{1,2\}$ disjoint from Lab.

The concurrent composition in COSY can be modelled in the following way. We first extend $\Omega_{\|}$ to a family of n-ary $(n \geq 1)$ concurrent composition operator boxes, $\Omega^n_{\|}$, in the obvious way. What we define next is a family of simple operator boxes $\Omega_{COSY^{a_1 \ldots a_k}_{m_1 \ldots m_k}}$ (where $k \geq 1$, $m_i \geq 1$ and $a_i \in \mathsf{Lab}$) based on the following relabellings:

$$\rho_{COSY^{a_1 \ldots a_k}_{m_1 \ldots m_k}} = \{(m_i \cdot \langle a_i \rangle, a_i) \mid i = 1, \ldots, k\}$$

and then defining

$$\mathsf{box}(program\ \ path_1 \ldots path_n\ \ endprogram) = \mathsf{op}_{\Omega_{COSY^{a_1 \ldots a_k}_{m_1 \ldots m_k}}}(\mathsf{op}_{\Omega^n_{\|}}(\mathsf{box}(path_1), \ldots, \mathsf{box}(path_n)))$$

where $a_1, \ldots, a_k$ are the labels occurring in the path program and each m_i is the number of paths within the program in which a_i occurs.

Note that by the above definitions and their surrounding theory, we have defined, all at once, compositional Petri net, step sequence, interleaving (obtained by taking only singleton steps), SOS, and partial order semantics for CCS, TCSP and COSY.

6.5 Operational semantics, soundness and completeness

Following the development of the operational semantics for the general case, to define it for the PBC, we need to introduce: (i) dynamic box expressions, (ii) equations defining the auxiliary similarity relation $\equiv_{\mathsf{OP}}$, and (iii) inference rules defining actual moves for the expressions. We do not have to expend much effort on any of these three steps, as they are automatically derived from the discussion in section 5 and a simple analysis of the markings reachable from the initial markings of the the six PBC operator boxes. (All we have to do is consider two single-token markings for each of $\Omega_{[f]}, \Omega_{\mathrm{rs}\ a}, \Omega_{\mathrm{sy}\ a}$ and $\Omega_{\square}$, three single-token markings for $\Omega_{;}$, and four two-token markings for $\Omega_{\|}$.) We therefore present (i), (ii) and (iii) without detailed explanations.

The dynamic PBC expressions are given by the following syntax rules:

$$D \ ::= \ \overline{E} \mid \underline{E} \mid D; E \mid E; D \mid D \,\square\, E \mid E \,\square\, D \mid D \ \mathsf{rs}\ a \mid D \ \mathsf{sy}\ a \mid D[f]$$

where E stands for an arbitrary (static) PBC expression obeying syntax (19).

The equations used to define the relation $\equiv_{\mathsf{OP}}$ are given in table 1. They are instances of equation schemes (E1)-(E3) of section 5.4. For example, ES1 is an instance of (E1), ES2 is an instance of (E3) and ES3 is an instance of (E2). In addition to these, we have, for every defining equation $K \stackrel{\mathrm{df}}{=} expr$,

$$\begin{aligned} \overline{K} &\equiv_{\mathsf{rec}} \overline{expr} \qquad \text{ER1} \\ \text{and} \quad \underline{K} &\equiv_{\mathsf{rec}} \underline{expr} \qquad \text{ER2} \end{aligned}$$

Tables 2 and 3 specify the inference rules for the two types of step sequence operational semantics defined for the expressions in PBC. The former gives the rules for that based on transition trees, while the latter for that based on labels. In the tables, γ and δ are finite multisets of multiactions, D, H, D' and H' are static and dynamic PBC expressions, and U and V are finite sets of transition trees. The notation $(v, g) \oplus U$, where v is a transition name, $g : \mathsf{Lab}_{\mathrm{PBC}} \to \mathsf{Lab}_{\mathrm{PBC}}$ is a mapping from labels

Operator	*Equation(s)*	
Sequence	$\overline{E;F} = \overline{E};F$	ES1
	$\underline{E};F = E;\overline{F}$	ES2
	$E;\underline{F} = \underline{E;F}$	ES3
Choice	$\overline{E \square F} = \overline{E} \square F$	EC1
	$\overline{E \square F} = E \square \overline{F}$	EC2
	$\underline{E} \square F = \underline{E \square F}$	EC3
	$E \square \underline{F} = \underline{E \square F}$	EC4
Concurrent composition	$\overline{E \| F} = \overline{E} \| \overline{F}$	ECC1
	$\underline{E} \| \underline{F} = \underline{E \| F}$	ECC2
Synchronisation	$\overline{E \ \mathsf{sy}\ a} = \overline{E} \ \mathsf{sy}\ a$	ESy1
	$\underline{E} \ \mathsf{sy}\ a = \underline{E \ \mathsf{sy}\ a}$	ESy2
Restriction	$\overline{E \ \mathsf{rs}\ a} = \overline{E} \ \mathsf{rs}\ a$	ERs1
	$\underline{E} \ \mathsf{rs}\ a = \underline{E \ \mathsf{rs}\ a}$	ERs2
Basic relabelling	$\overline{E[f]} = \overline{E}[f]$	ERl1
	$\underline{E}[f] = \underline{E[f]}$	ERl2

Table 1: Equations for the basic PBC.

Operator	*Inference rule(s)*	
Basic action	$\overline{\alpha} \xrightarrow{\{\overset{\bullet}{\alpha}\}} \underline{\alpha}$	RB
Sequence	$\dfrac{D \xrightarrow{U} D' \;,\; H \xrightarrow{V} H'}{D;H \xrightarrow{(1_;,id)\oplus U \,\cup\, (2_;,id)\oplus V} D';H'}$	RS
Choice	$\dfrac{D \xrightarrow{U} D' \;,\; H \xrightarrow{V} H'}{D \,\Box\, H \xrightarrow{(1_\Box,id)\oplus U \,\cup\, (2_\Box,id)\oplus V} D' \,\Box\, H'}$	RC
Concurrent composition	$\dfrac{D \xrightarrow{U} D' \;,\; H \xrightarrow{V} H'}{D \Vert H \xrightarrow{(1_\Vert,id)\oplus U \,\cup\, (2_\Vert,id)\oplus V} D' \Vert H'}$	RCC
Basic relabelling	$\dfrac{D \xrightarrow{U} H}{D[f] \xrightarrow{(1_{[f]},f)\oplus U} H[f]}$	RRe
Restriction	$\dfrac{D \xrightarrow{U} H \;,\; \mathsf{lab}\langle U\rangle \in \mathsf{mult}(\mathsf{mult}(\mathsf{A}\setminus\{a,\widehat{a}\}))}{D \;\mathsf{rs}\; a \xrightarrow{(1\,\mathrm{rs}\;a,id)\oplus U} H \;\mathsf{rs}\; a}$	RRs
Synchronisation	$\dfrac{D \xrightarrow{U_1 \uplus \ldots \uplus U_k} H \;,\; (\mathsf{lab}\langle U_i\rangle, \alpha_i) \in \rho_{\,\mathrm{sy}\;a}\; (1 \le i \le k)}{D \;\mathsf{sy}\; a \xrightarrow{\{(1\,\mathrm{sy}\;a,\alpha_1)\lhd U_1, \ldots, (1\,\mathrm{sy}\;a,\alpha_k)\lhd U_k\}} H \;\mathsf{sy}\; a}$	RSy

Table 2: Inference rules based on transition trees.

Operator	*Inference rule(s)*	
Basic action	$\overline{\alpha} \xrightarrow{\langle\alpha\rangle} \underline{\alpha}$	RB
Sequence	$\dfrac{D \xrightarrow{\gamma} D' \;,\; H \xrightarrow{\delta} H'}{D;H \xrightarrow{\gamma+\delta} D';H'}$	RS
Choice	$\dfrac{D \xrightarrow{\gamma} D' \;,\; H \xrightarrow{\delta} H'}{D \,\Box\, H \xrightarrow{\gamma+\delta} D' \,\Box\, H'}$	RC
Concurrent composition	$\dfrac{D \xrightarrow{\gamma} D' \;,\; H \xrightarrow{\delta} H'}{D \Vert H \xrightarrow{\gamma+\delta} D' \Vert H'}$	RCC
Basic relabelling	$\dfrac{D \xrightarrow{\gamma} H}{D[f] \xrightarrow{f(\gamma)} H[f]}$	RRe
Restriction	$\dfrac{D \xrightarrow{\gamma} H \;,\; \gamma \in \mathsf{mult}(\mathsf{mult}(\mathsf{A} \setminus \{a, \widehat{a}\}))}{D \;\mathsf{rs}\; a \xrightarrow{\gamma} H \;\mathsf{rs}\; a}$	RRs
Synchronisation	$\dfrac{D \xrightarrow{\gamma_1+\cdots+\gamma_k} H \;,\; (\gamma_i, \alpha_i) \in \rho_{\,\mathsf{sy}\; a} \; (1 \leq i \leq k)}{D \;\mathsf{sy}\; a \xrightarrow{\langle\alpha_1,\ldots,\alpha_k\rangle} H \;\mathsf{sy}\; a}$	RSy

Table 3: Inference rules based on labels.

to labels and $U = \{\tau_1, \ldots, \tau_k\}$ is a finite set of transition trees, denotes the set of transition trees $\{(v, g(\mathsf{lab}(\tau_1))) \lhd \tau_1, \ldots, (v, g(\mathsf{lab}(\tau_k))) \lhd \tau_k\}$. Moreover, *id* is the identity on the set of PBC labels. We also use the inaction rule based on the similarity relation $\equiv$.

We can state two fundamental consistency results for the PBC algebra which are formulated exactly as theorems 5.1 and 5.2 under the proviso that all expressions involved are PBC expressions as defined in this section. Both results were stated in [25] without proof; here their correctness follows directly from their general counterparts. Recall that theorems 5.1 and 5.2 are based on their 'syntax-free' counterparts, theorems 3.2 and 3.3. Recall also that the latter have a basic consequence showing consistency of operator boxes, namely theorem 3.4 which can also be specialised in the context of this section (it follows directly from lemma C.1):

Corollary 6.1 If E is a static PBC expression then all markings reachable from the initial marking of $\mathsf{box}(E)$ are safe and clean. $\square$

This corollary can be found, with a different proof, in [14]. Finally, we would like to point out that other results established in section 5, in particular, theorems 5.4 and 5.6, hold also for the PBC algebra.

6.6 Examples

We first show derivations based on the inference rules and equations in tables 1 and 3, indicating which inference rules have been applied and how the application of the inaction rule can be justified. In the examples, each singleton multiset $\langle a \rangle \in \mathsf{Lab}_{\mathrm{PBC}}$ will be denoted by a. We first deduce that the expression $\overline{(a;b) \square c}$ can do an a-move, and thereafter a b-move and terminate, i.e.,

$$\overline{(a;b) \square c} \xrightarrow{\langle a \rangle \langle b \rangle} \underline{(a;b) \square c}.$$

The above step sequence can be obtained in the following way:

$$\begin{array}{rcll}
\overline{(a;b) \square c} & \equiv & \overline{(a;b)} \square c & \text{EC1} \\
 & \equiv & (\overline{a};b) \square c & \text{ES1} \\
 & \xrightarrow{\langle a \rangle} & (\underline{a};b) \square c & \text{RB RS RC} \\
 & \equiv & (a;\overline{b}) \square c & \text{ES2} \\
 & \xrightarrow{\langle b \rangle} & (a;\underline{b}) \square c & \text{RB RS RC} \\
 & \equiv & \underline{(a;b)} \square c & \text{ES3} \\
 & \equiv & \underline{(a;b) \square c} & \text{EC3}
\end{array}$$

As an application of the rule for synchronisation, we show that

$$\overline{E} \xrightarrow{\langle \langle b,c \rangle \rangle} \underline{E}$$

where $E = ((\langle a,b \rangle \| \langle \widehat{a}, \widehat{a} \rangle) \| \langle a,c \rangle)\ \mathsf{sy}\ a$. A detailed derivation for this is provided below:

$$\begin{array}{rcll}
\overline{E} & \equiv & (\overline{(\langle a,b \rangle \| \langle \widehat{a}, \widehat{a} \rangle) \| \langle a,c \rangle})\ \mathsf{sy}\ a & \text{ESy1} \\
 & \equiv & (\overline{(\langle a,b \rangle \| \langle \widehat{a}, \widehat{a} \rangle)} \| \overline{\langle a,c \rangle})\ \mathsf{sy}\ a & \text{ECC1} \\
 & \equiv & ((\overline{\langle a,b \rangle} \| \overline{\langle \widehat{a}, \widehat{a} \rangle}) \| \overline{\langle a,c \rangle})\ \mathsf{sy}\ a & \text{ECC1} \\
 & \xrightarrow{\langle \langle b,c \rangle \rangle} & ((\underline{\langle a,b \rangle} \| \underline{\langle \widehat{a}, \widehat{a} \rangle}) \| \underline{\langle a,c \rangle})\ \mathsf{sy}\ a & \text{RB}(\times 3)\ \text{RCC}(\times 2)\ \text{RSy} \\
 & \equiv & \underline{E} & \text{ECC2}(\times 2)\ \text{ESy2}
\end{array}$$

As the next example we show that $\overline{K}$, for which the defining equation is $K \stackrel{\mathrm{df}}{=} (a;K) \Box b$, can first make an a-move, then a b-move and terminate, i.e.,

$$\overline{K} \xrightarrow{\langle a\rangle\langle b\rangle} \underline{K}$$

We first replace the defining equation by a set of two equations, $K \stackrel{\mathrm{df}}{=} M \Box b$ and $M \stackrel{\mathrm{df}}{=} a;K$. Then the detailed derivation can be carried out as shown in figure 14. Note that $\overline{K}$ can make an infinite sequence of a-moves since figure 14 shows that

$$\overline{K} \xrightarrow{\langle a\rangle} (a;\overline{K}) \Box b.$$

$$\begin{array}{lcll}
\overline{K} & \equiv & \overline{M \Box b} & \mathrm{ER1} \\
 & \equiv & \overline{M} \Box b & \mathrm{EC1} \\
 & \equiv & \overline{a;K} \Box b & \mathrm{ER1} \\
 & \equiv & (\overline{a};K) \Box b & \mathrm{ES1} \\
 & \xrightarrow{\langle a\rangle} & (\underline{a};K) \Box b & \mathrm{RB\ RS\ RC} \\
 & \equiv & (a;\overline{K}) \Box b & \mathrm{ES2} \\
 & \equiv & (a;\overline{(M \Box b)}) \Box b & \mathrm{ER1} \\
 & \equiv & (a;(M \Box \overline{b})) \Box b & \mathrm{EC2} \\
 & \xrightarrow{\langle b\rangle} & (a;(M \Box \underline{b})) \Box b & \mathrm{RB\ RC(\times 2)\ RS} \\
 & \equiv & (a;\underline{(M \Box b)}) \Box b & \mathrm{EC4} \\
 & \equiv & (a;\underline{K}) \Box b & \mathrm{ER2} \\
 & \equiv & \underline{(a;K)} \Box b & \mathrm{ES3} \\
 & \equiv & \underline{M} \Box b & \mathrm{ER2} \\
 & \equiv & \underline{M \Box b} & \mathrm{EC3} \\
 & \equiv & \underline{K} & \mathrm{ER2}
\end{array}$$

Figure 14: A derivation for $\overline{K}$ defined by $K \stackrel{\mathrm{df}}{=} M \Box b$ and $M \stackrel{\mathrm{df}}{=} a;K$.

The next example is similar to that discussed in [1] and shows a non-tail-end recursive expression not directly expressible in CCS. Figure 15 shows that the dynamic expression $\overline{K}$, for which the defining equation is $K \stackrel{\mathrm{df}}{=} (K;a) \Box a$, can make two successive a-moves, i.e.,

$$\overline{K} \xrightarrow{\langle a\rangle\langle a\rangle} \underline{K}.$$

Note that we first transformed the equation into the format required by the scheme developed in section 5, i.e., $K \stackrel{\mathrm{df}}{=} M \Box a$ and $M \stackrel{\mathrm{df}}{=} K;a$. By induction, it can be shown that $\overline{K}$ can do n successive a-moves, for every n. But the argument cannot be used to show that it can do an infinite sequence of a-moves; in fact it cannot. This may be contrasted with the previous example.

$$
\begin{array}{lll l}
\overline{K} & \equiv & \overline{M \Box a} & \text{ER1} \\
 & \equiv & \overline{M} \Box a & \text{EC1} \\
 & \equiv & \overline{(K;a)} \Box a & \text{ER1} \\
 & \equiv & (\overline{K};a) \Box a & \text{ES1} \\
 & \equiv & ((\overline{M \Box a});a) \Box a & \text{ER1} \\
 & \equiv & ((M \Box \overline{a});a) \Box a & \text{EC2} \\
 & \xrightarrow{\langle a \rangle} & ((M \Box \underline{a});a) \Box a & \text{RB RC2}(\times\ 2)\ \text{RS} \\
 & \equiv & (\underline{(M \Box a)};a) \Box a & \text{EC4} \\
 & \equiv & (\underline{K};a) \Box a & \text{ER2} \\
 & \equiv & (K;\overline{a}) \Box a & \text{ES2} \\
 & \xrightarrow{\langle a \rangle} & (K;\underline{a}) \Box a & \text{RB RS RC} \\
 & \equiv & \underline{(K;a)} \Box a & \text{ES3} \\
 & \equiv & \underline{M} \Box a & \text{ER2} \\
 & \equiv & \underline{M \Box a} & \text{EC3} \\
 & \equiv & \underline{K} & \text{ER2}
\end{array}
$$

Figure 15: A derivation for $\overline{K}$ defined by $K \stackrel{\mathrm{df}}{=} M \Box a$ and $M \stackrel{\mathrm{df}}{=} K;a$.

As the next example, we prove by induction that the dynamic expression $\overline{K}$, for which the defining equation is $K \stackrel{\mathrm{df}}{=} a \| K$, can do a γ_n-move, for every $\gamma_n = n \cdot \langle a \rangle$, $n \geq 1$. That is, we want to show that for every $n \geq 1$ there is D_n such that

$$\overline{K} \xrightarrow{\quad \gamma_n \quad} D_n .$$

In the base case, $n = 1$, the above follows from the following derivation:

$$\begin{array}{ccll} \overline{K} & \equiv & \overline{a \| K} & \text{ER} \\ & \equiv & \overline{a} \| \overline{K} & \text{ECC1} \\ & \xrightarrow{\langle a \rangle} & \underline{a} \| \overline{K} & \text{RB RCC} \end{array}$$

In the induction step, one can see that D_{n+1} exists, in the following way. From the induction hypothesis it follows that there are D and D_n such that $\overline{K} \equiv D \xrightarrow{\gamma_n} H$ (note that the empty step move between two expressions implies structural similarity). Then

$$\begin{array}{ccll} \overline{K} & \equiv & \overline{a \| K} & \text{ER} \\ & \equiv & \overline{a} \| \overline{K} & \text{ECC1} \\ & \equiv & \overline{a} \| D & \text{Induction hypothesis} \\ & \xrightarrow{\langle a \rangle + \gamma_n} & \underline{a} \| H & \text{RB Induction hypothesis RCC} \end{array}$$

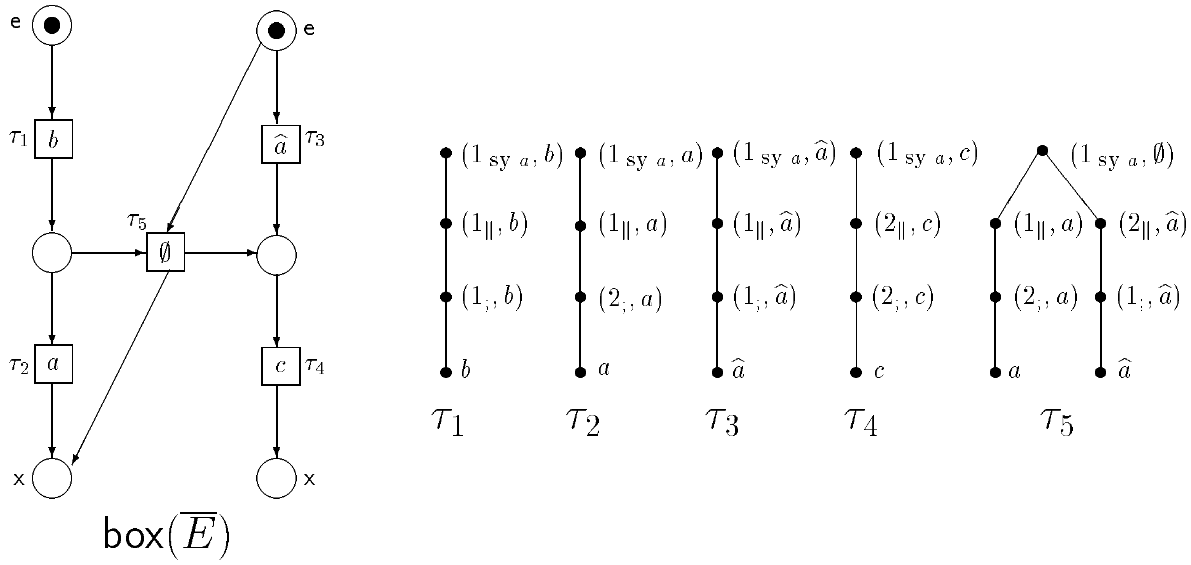


Figure 16: The box semantics of $\overline{E} = \overline{((b; a) \| (\widehat{a}; c)) \ \mathsf{sy}\ a}$.

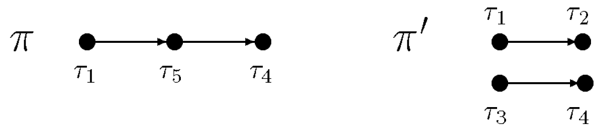
We now turn to the partial order semantics based on transition trees and the derived partial order semantics. Let E be an expression defined as $((b; a) \| (\widehat{a}; c))$ sy a. Figure 16 shows $\mathsf{box}(\overline{E})$ together with the transition trees which serve as the actual names of transitions. Note that there are four pairs of independent transition trees in the sense of the relation ind defined in the proof of proposition 5.5, namely (τ_1, τ_3), (τ_1, τ_4), (τ_2, τ_3) and (τ_2, τ_4). It is possible to show that $\overline{E}$ can generate the following two step sequences,

$$\overline{E} \xrightarrow{\{\tau_1\}\{\tau_5\}\{\tau_4\}} \underline{E} \quad \text{and} \quad \overline{E} \xrightarrow{\{\tau_1\}\{\tau_2, \tau_3\}\{\tau_4\}} \underline{E}.$$

The following is a detailed derivation for the first one:

$$
\begin{array}{llllllll}
\overline{E} & \equiv & \overline{((b;a)\|(\widehat{a};c))}\ \mathsf{sy}\ a & \equiv & (\overline{(b;a)}\|\overline{(\widehat{a};c)})\ \mathsf{sy}\ a & \equiv & ((\overline{b};a)\|(\overline{\widehat{a}};c))\ \mathsf{sy}\ a \\
 & \xrightarrow{\{\tau_1\}} & ((\underline{b};a)\|(\overline{\widehat{a}};c))\ \mathsf{sy}\ a & \equiv & ((b;\overline{a})\|(\overline{\widehat{a}};c))\ \mathsf{sy}\ a & \xrightarrow{\{\tau_5\}} & ((b;\underline{a})\|(\underline{\widehat{a}};c))\ \mathsf{sy}\ a \\
 & \equiv & ((b;a)\|(\widehat{a};\overline{c}))\ \mathsf{sy}\ a & \xrightarrow{\{\tau_4\}} & (\underline{(b;a)}\|(\widehat{a};\underline{c}))\ \mathsf{sy}\ a & \equiv & (\underline{(b;a)}\|\underline{(\widehat{a};c)})\ \mathsf{sy}\ a \\
 & \equiv & \underline{((b;a)\|(\widehat{a};c))}\ \mathsf{sy}\ a & \equiv & \underline{E} & &
\end{array}
$$

As a result, we obtain that $\overline{E} \xrightarrow{\pi} \underline{E}$ and $\overline{E} \xrightarrow{\pi'} \underline{E}$, where π and π' are the following labelled partial orders:



As the last example, note that $\overline{K}$, where $K \stackrel{\text{df}}{=} K[\rho_{id}]$, cannot make any nonempty move at all, although it can make infinitely many 'empty moves' (but no silent move, nor any other move):

$$\overline{K} \xrightarrow{\emptyset} \overline{K[\rho_{id}]} \xrightarrow{\emptyset} \overline{K}[\rho_{id}] \xrightarrow{\emptyset} \overline{K[\rho_{id}]}[\rho_{id}] \xrightarrow{\emptyset} \cdots$$

This, too, is consistent with the box semantics where $\mathsf{box}(K)$ is an ex box having only one entry place, one exit place, and no transitions.

7 Concluding Remarks

This paper describes two groups of results. The first group shows that recursive equations have, in general, Petri net solutions. These results have been given a fixpoint rendering. The second group gives soundness and completeness of an operational semantics including the recursive case. These results have been given an algebra-independent rendering, and then also a phrasing in terms of an algebra and both its step sequence and its partial order semantics.

Among the directions for further research we would mention the following four possible extensions of the current model:

- Multisets rather than sets could be used in the definition of the transitions of a refinement. As we already mentioned, such a definition would only make sense if we were to allow non-safe boxes.

- The safeness of boxes in the current model could be relaxed to either n-boundedness or unboundedness. Note that allowing boxes to be unsafe means that the operator boxes need no longer be pure.

- Entry places could be allowed to have incoming arcs, and exit places outgoing arcs, provided that the cleanness[11] of reachable markings remained a valid property.

- The net refinement could be generalised by allowing the operator net Ω to contain tokens contributing to the marking of the result, $\mathsf{op}_\Omega(\Sigma)$. In such a case, the class of operator boxes could be extended to include a wider set of nets, for example, the N-box from section 3.2.

In addition, it may be worthwhile to explore stronger connections with general domain theory [31].

[11] Actually, one would also need to ensure that the initial marking of a box could not be reached from itself by a nonempty step sequence, and that the terminal marking was deadlocked.

Acknowledgments

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 non-pure nets), and to Javier Esparza who did a formalisation for guardedly recursive expressions [15]. Both also gave useful comments on previous versions of this paper, as gave the anonymous reviewers. 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] J.W. de Bakker: Denotational Semantics of Unguarded Recursion. Unpublished Manuscript (1995/96).

[3] E. Best, R. Devillers: Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* Vol. 55/1, 87–136 (1988).

[4] E. Best, R. Devillers, J. Esparza: General Refinement and Recursion Operators for the Petri Box Calculus. *Proc. STACS'93*, Springer-Verlag, Lecture Notes in Computer Science Vol. 665, 130–140 (1993).

[5] 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).

[6] E. Best, C. Fernández: *Nonsequential Processes. A Petri Net View.* Springer-Verlag, EATCS Monographs on Theoretical Computer Science Vol. 13 (1988).

[7] E. Best, M. Koutny: A Refined View of the Box Algebra. *Proc. Petri Net Conference'95*, G. De Michelis, M. Diaz (eds). Springer-Verlag, Lecture Notes in Computer Science Vol. 935, 1–20 (1995).

[8] E. Best, M. Koutny: Solving Recursive Net Equations. *Proc. ICALP'95*, Z. Fülöp and F. Gécseg (eds). Springer-Verlag, Lecture Notes in Computer Science Vol. 944, 605–623 (1995).

[9] G. Boudol, I. Castellani: Flow Models of Distributed Computations: Event Structures and Nets. Rapport de Recherche, INRIA, Sophia Antipolis (July 1991).

[10] B.A. Davey, H.A. Priestley: *Introduction to Lattices and Order.* Cambridge Mathematical Textbook (1990).

[11] R. Devillers: Towards a General Relabelling Operator for the Petri Box Calculus. Technical Report LIT-274, Laboratoire d'Informatique Théorique, Université Libre de Bruxelles (1993).

[12] R. Devillers: Construction of S-invariants and S-components for Refined Petri Boxes. *Proc. Petri Net Conference'93*, Chicago, Springer-Verlag, Lecture Notes in Computer Science 691 (1993).

[13] R. Devillers: The Synchronisation Operator Revisited for the Petri Box Calculus. Technical Report LIT-290, Laboratoire d'Informatique Théorique, Université Libre de Bruxelles (1994).

[14] R. Devillers: S-invariant Analysis of Recursive Petri Boxes. *Acta Informatica* Vol. 32, 313-345 (1995).

[15] J. Esparza: Fixpoint Definition of Recursion in the Box Algebra. Memorandum, Universität Hildesheim (December 1991).

[16] W. Frączak, H. Klaudel: A Multi-action Synchronisation Scheme and its Application to the Petri Box Calculus. *Proc. ESDA (Engineering Systems Design and Analysis Conference)*, London 91-100, (1994).

[17] 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).

[18] U. Goltz, R. van Glabbeek: Refinement of Actions in Causality Based Models. Springer-Verlag, Lecture Notes in Computer Science Vol. 430, 267-300 (1989).

[19] M.B. Hennessy: *Algebraic Theory of Processes*. The MIT Press (1988).

[20] C.A.R. Hoare: *Communicating Sequential Processes*. Prentice Hall (1985).

[21] R. Janicki, M. Koutny: Structure of Concurrency. *Theoretical Computer Science* Vol. 112, 5-52 (1993).

[22] R. Janicki, P.E. Lauer: *Specification and Analysis of Concurrent Systems - the COSY Approach*. Springer-Verlag, EATCS Monographs on Theoretical Computer Science (1992).

[23] K. Jensen: *Coloured Petri Nets. Basic Concepts*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1992).

[24] M. Koutny: Partial Order Semantics of Box Expressions. *Proc. Petri Net Conference'94*, R. Valette (ed.), Springer-Verlag, Lecture Notes in Computer Science Vol. 815, 318-337 (1994).

[25] M. Koutny, J. Esparza, E. Best: Operational Semantics for the Petri Box Calculus. *Proc. CONCUR'94*, B. Jonsson and J.Parrow (eds), Springer-Verlag, Lecture Notes in Computer Science Vol. 836, 210-225 (1994).

[26] 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).

[27] A. Mazurkiewicz: Trace Theory. *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Advances in Petri Nets 1986, Part II, W. Brauer, W. Reisig and G. Rozenberg (eds). Springer-Verlag, Lecture Notes in Computer Science Vol. 255, 279–324 (1987).

[28] R. Milner: *Communication and Concurrency*. Prentice Hall (1989).

[29] E.R. Olderog: *Nets, Terms and Formulas*. Cambridge Tracts in Theoretical Computer Science 23 (1991).

[30] G. Plotkin: *A Structural Approach to Operational Semantics*. DAIMI Technical Report FN-19, Computer Science Department, University of Århus (1981).

[31] G. Plotkin: *Domains*. Department of Computer Science, University of Edinburgh (1983).

[32] W. Reisig: *Petri Nets. An Introduction*. Springer-Verlag, EATCS Monographs on Theoretical Computer Science Vol. 3, (1985).

[33] D. Taubner: *Finite Representation of CCS and TCSP Programs by Automata and Petri Nets*. Springer-Verlag, Lecture Notes in Computer Science Vol. 369 (1989).

A Appendix: Proofs of sections 2 and 3

Our goal here is to prove theorems 3.2, 3.3 and 3.4. To this end we assume, unless stated otherwise, that $\Omega = (S, T, W, \lambda, \emptyset)$ is an n–ary operator box, $\phi = (\phi_1, \phi_2)$ is a factorisation of a marking reachable from ${}^\circ\Omega$ and $\mathbf{\Sigma} \in \mathsf{dom}_\Omega^\phi$ is an Ω–tuple of boxes. In addition to the notation introduced in sections 2.2 and 3, we will use the following.

(1) For every $s \in S$, ${}^\diamond s = \{(v, q) \mid v \in {}^\bullet s \wedge q \in \Sigma_v^\circ\}$ and $s^\diamond = \{(v, q) \mid v \in s^\bullet \wedge q \in {}^\circ\Sigma_v\}$.

(2) For every $p \in S_0$ and $v \in T$, we define $\mathsf{tree}^v(p) = \bot$ (where $\bot \notin \mathsf{P}_{\mathsf{tree}}$) unless one of the following holds:

   (a) $p = v \lhd q \in \mathsf{ST}_{\mathsf{new}}^v$. Then we define $\mathsf{tree}^v(p) = q$.

   (b) $p = s \lhd \{v_1 \lhd q_1, \ldots, v_k \lhd q_k, v_{k+1} \lhd q_{k+1}, \ldots, v_{k+m} \lhd q_{k+m}\} \in \mathsf{SP}_{\mathsf{new}}^s$ and $v = v_i$, for some $i \leq k + m$. Then we define $\mathsf{tree}^v(p) = q_i$.

(3) $\Sigma_0 = \mathsf{op}_\Omega(\mathbf{\Sigma})$.

Note that since Ω is pure there can be at most one i in (2b) such that $v = v_i$, so $\mathsf{tree}^v(p)$ is well-defined. Moreover, in (2b), $(v, \mathsf{tree}^v(p)) \in {}^\diamond s \cup s^\diamond$.

A.1 Basic properties of net refinement

The first two lemmata present some useful results about Σ_0.

Lemma A.1 Let $s \in S$ and $v, t \in T$ $(v \neq t)$.

(1) $S_0 \cap T_0 = \emptyset$, $S_0 \subseteq \mathsf{P}_{\mathsf{tree}}$ and $T_0 \subseteq \mathsf{T}_{\mathsf{tree}}$.

(2) ${}^\bullet(\mathsf{T}_{\mathsf{new}}^v) \subseteq \bigcup\{\mathsf{SP}_{\mathsf{new}}^s \mid s \in {}^\bullet v\} \cup \mathsf{ST}_{\mathsf{new}}^v$ and $(\mathsf{T}_{\mathsf{new}}^v)^\bullet \subseteq \bigcup\{\mathsf{SP}_{\mathsf{new}}^s \mid s \in v^\bullet\} \cup \mathsf{ST}_{\mathsf{new}}^v$.

(3) $\mathsf{SP}_{\mathsf{new}}^s$ is a non-empty set.

(4) If $(v, q) \in {}^\diamond s \cup s^\diamond$ then there is $p \in \mathsf{SP}_{\mathsf{new}}^s$ such that $q = \mathsf{tree}^v(p)$.

(5) If $q \in S_v \backslash ({}^\circ\Sigma_v \cup \Sigma_v^\circ)$ then $v \lhd q \in \mathsf{ST}_{\mathsf{new}}^v$.

(6) If $q \in S_v$ then there is $p \in S_0$ such that $q = \mathsf{tree}^v(p)$.

(7) If $(v, q), (t, r) \in {}^\diamond s \cup s^\diamond$ then there is $p \in \mathsf{SP}_{\mathsf{new}}^s$ such that $q = \mathsf{tree}^v(p)$ and $r = \mathsf{tree}^t(p)$.

(8) $\lfloor \Sigma_0 \rfloor$ is a box.

Proof. (1,2) Follow directly from the definition of net refinement and $\mathbf{\Sigma}$ being ex–directed. (3,4) Follow from each Σ_v being ex–restricted and the fact that s is not an isolated place (in the case of (3)). (5) Follows directly from the definition of refinement. (6) Follows from (4) and (5). (7) Follows from each Σ_v being ex–restricted and $v \neq t$.

(8) By (1), what we need to show is that Σ_0 is ex-restricted, ex-directed and T-restricted. Σ_0 is ex–restricted by (3) and Ω being ex–restricted. Suppose $s \in {}^\circ\Omega$, $p \in \mathsf{SP}_{\mathsf{new}}^s$, $t \in \mathsf{T}_{\mathsf{new}}^v$ and $W_0(t, p) > 0$. Then $\mathsf{tree}^v(p) \neq \bot$ and $W_v(w, \mathsf{tree}^v(p)) > 0$, for some $w \in \mathsf{trees}(t)$. By ${}^\bullet s = \emptyset$ (since Ω is ex–directed), $(v, \mathsf{tree}^v(p)) \in s^\diamond$. Hence $w \in {}^\bullet({}^\circ\Sigma_v)$, contradicting Σ_v being ex–directed. Thus ${}^\bullet({}^\circ\Sigma_0) = \emptyset$. Since $(\Sigma_0^\circ)^\bullet = \emptyset$ can be shown in a similar way, Σ_0 is ex–directed.

To prove that Σ_0 is T–restricted, suppose $t \in \mathsf{T}_{\mathsf{new}}^v$. Take any $u \in \mathsf{trees}(t)$. Since Σ_v is T–restricted, there is $q \in S_v$ such that $W_v(q, u) > 0$. By (6), there is $p \in S_0$ such that $q = \mathsf{tree}^v(p)$. Hence $p \in {}^\bullet t$. Similarly, we may show that $t^\bullet \neq \emptyset$. ■ A.1

Lemma A.2 Let $t \in \mathsf{T}^v_{\mathsf{new}}$, $u \in \mathsf{T}^w_{\mathsf{new}}$ and $(F, F'), (G, G') \in \{({}^\bullet(.), {}^\circ(.)), ((.)^\bullet, (.)^\circ)\}$.

(1) If $v = w$ then $F(t) \cap G(u) \neq \emptyset$ if and only if $F(\mathsf{trees}(t)) \cap G(\mathsf{trees}(u)) \neq \emptyset$.

(2) If $v \neq w$ then $F(t) \cap G(u) \neq \emptyset$ if and only if

$$F(\mathsf{trees}(t)) \cap F'(\Sigma_v) \neq \emptyset \text{ and } G(\mathsf{trees}(t)) \cap G'(\Sigma_w) \neq \emptyset \text{ and } F(v) \cap G(w) \neq \emptyset.$$

(3) $F(t) \cap F'(\Sigma_0) \neq \emptyset$ if and only if

$$F(v) \cap F'(\Omega) \neq \emptyset \text{ and } F(\mathsf{trees}(t)) \cap F'(\Sigma_v) \neq \emptyset.$$

Note: We here treat ${}^\bullet(.)$, $(.)^\bullet$, ${}^\circ(.)$ and $(.)^\circ$ as generic operators – the actual domain and codomain is determined by the context. For example, an istance of (3) is: ${}^\bullet t \cap {}^\circ\Sigma_0 \neq \emptyset$ if and only if ${}^\bullet v \cap {}^\circ\Omega \neq \emptyset$ and ${}^\bullet\mathsf{trees}(t) \cap {}^\circ\Sigma_v \neq \emptyset$.

Proof. Without loss of generality, we assume $F = G = {}^\bullet(.)$ and $F' = G' = {}^\circ(.)$.

(1) Suppose $p \in {}^\bullet t \cap {}^\bullet u$. Then, there are $x \in \mathsf{trees}(t)$ and $y \in \mathsf{trees}(u)$ such that $W_v(\mathsf{tree}^v(p), x) > 0$ and $W_v(\mathsf{tree}^v(p), y) > 0$. Hence ${}^\bullet\mathsf{trees}(t) \cap {}^\bullet\mathsf{trees}(u) \neq \emptyset$. The reverse implication holds by lemma A.1(6).

(2) Suppose $p \in {}^\bullet t \cap {}^\bullet u$. Then there are $x \in \mathsf{trees}(t)$ and $y \in \mathsf{trees}(u)$ such that $W_v(\mathsf{tree}^v(p), x) > 0$ and $W_w(\mathsf{tree}^w(p), y) > 0$. Thus $p \in \mathsf{SP}^s_{\mathsf{new}}$ for some $s \in S$, and $(v, \mathsf{tree}^v(p)) \in {}^\diamond s \cup s^\diamond$. Moreover, by ex–directedness of Σ_v, $\mathsf{tree}^v(p) \notin \Sigma_v^\circ$. Hence $(v, \mathsf{tree}^v(p)) \in s^\diamond$. Similarly, $(w, \mathsf{tree}^w(p)) \in s^\diamond$. Hence the $(\Rightarrow)$ implication holds. The reverse one follows from lemma A.1(7).

(3) Suppose $p \in {}^\bullet t \cap {}^\circ\Sigma_0$. Then, from the e-labelling of p it follows that there is $s \in {}^\circ\Omega$ such that $p \in \mathsf{SP}^s_{\mathsf{new}}$. Moreover, there is $x \in \mathsf{trees}(t)$ such that $W_v(\mathsf{tree}^v(p), x) > 0$. Hence, by lemma A.1(2) and $s \in {}^\circ\Omega$, $(v, \mathsf{tree}^v(p)) \in s^\diamond$. Thus the $(\Rightarrow)$ implication holds. The reverse one follows from lemma A.1(4).

■ A.2

A.2 Properties of factorisations

For every factorisation $\mu = (\mu_1, \mu_2)$ of a set of places of Ω, we will use M_μ to denote ${}^\bullet\mu_1 \cup \mu_2^\bullet$. Moreover, for every $s \in S$, μ_s will be defined as the transition v satisfying: $v \in ({}^\bullet s \cup s^\bullet) \cap \mu_1$ or $v \in {}^\bullet s \cap \mu_2$. If such a v does not exist, $\mu_s = \bot$.

Lemma A.3 Let $\mu = (\mu_1, \mu_2)$ be a factorisation of a marking $M \in [{}^\circ\Omega\rangle$.

(1) If $t \in T$ and ${}^\bullet t \subseteq M$ then $t^\bullet \cap M = \emptyset$.

(2) $\mu_1 \cap \mu_2 = \emptyset$ and $\mu_1^\bullet \cap \mu_2^\bullet = \emptyset$ and $\mu_1 \in \mathsf{Ind}_\Omega$.

(3) ${}^\circ\Omega \subseteq M$ implies ${}^\bullet\mu_1 = M = {}^\circ\Omega$ and $\mu_2 = \emptyset$.

(4) $\Omega^\circ \subseteq M$ implies $\mu_2^\bullet = M = \Omega^\circ$ and $\mu_1 = \emptyset$.

(5) μ_s is well defined, and if $\mu_s \neq \bot$ then $(({}^\bullet s \cup s^\bullet) \cap \widetilde{\mu}) \backslash \{\mu_s\} \subseteq s^\bullet \cap \mu_2$.

(6) If $\eta = (\eta_1, \eta_2)$ and $\psi = (\psi_1, \psi_2)$ are two factorisations such that $M_\eta = M_\psi$ and $\eta_1 \subseteq \mu_1$ and $\eta_2 \subseteq \mu_2$ then $\kappa = ((\mu_1 \backslash \eta_1) \cup \psi_1, (\mu_2 \backslash \eta_2) \cup \psi_2)$ is a factorisation of M.

Proof. (1) If ${}^\bullet t \subseteq M$ and $t^\bullet \cap M \neq \emptyset$ then, since Ω is simple, we can execute step $\{t\}$, and thus obtain a contradiction with Ω being a pure net and $[{}^\circ\Omega\rangle$ being safe markings.

(2) The first two properties follow from (1) and Ω being T–restricted. The third one is shown by observing that μ_1 is a step enabled at M. We then use proposition 3.1.

(3) Since $^\circ\Omega$ is clean, $^\circ\Omega = M$. This and Ω being ex–directed and T–restricted, means that $\mu_2 = \emptyset$ and ${}^\bullet\mu_1 = M$. (4) can be shown in a similar way.

(5) Part (2) and the definition of factorisation (the disjoint union condition) imply that there can be at most one v satisfying the definition of μ_s. The second part also follows from the definition of factorisation and (2).

(6) Since μ, η and ψ are factorisations, κ will also be one, provided that:

$$({}^\bullet(\mu_1\backslash\eta_1) \cup (\mu_2\backslash\eta_2)^\bullet) \cap ({}^\bullet\psi_1 \cup \psi_2^\bullet) = \emptyset.$$

Hence, since $M_\eta = M_\psi$, it suffices to show

$$({}^\bullet(\mu_1\backslash\eta_1) \cup (\mu_2\backslash\eta_2)^\bullet) \cap ({}^\bullet\eta_1 \cup \eta_2^\bullet) = \emptyset.$$

But this holds since μ is a factorisation and $\eta_1 \subseteq \mu_1$ and $\eta_2 \subseteq \mu_2$. We show $M = M_\mu = M_\kappa$ thus:

$${}^\bullet(\mu_1\backslash\eta_1) \cup {}^\bullet\psi_1 \cup (\mu_2\backslash\eta_2)^\bullet \cup \psi_2^\bullet =_{\text{(i)}} {}^\bullet(\mu_1\backslash\eta_1) \cup {}^\bullet\eta_1 \cup (\mu_2\backslash\eta_2)^\bullet \cup \eta_2^\bullet =_{\text{(ii)}} {}^\bullet\mu_1 \cup \mu_2^\bullet.$$

In the above, (i) holds since $M_\eta = M_\psi$, and (ii) holds since $\eta_1 \subseteq \mu_1$ and $\eta_2 \subseteq \mu_2$. ■ A.3

After proving some basic properties of place factorisations, we may show that the sub–domain of dom_Ω to which $\mathbf{\Sigma}$ belongs, may be chosen in a rather special way.

Lemma A.4 There is $\psi = (\psi_1, \psi_2) \in \mathsf{fact}_\Omega$ such that $\mathbf{\Sigma} \in \mathsf{dom}_\Omega^\psi$ and $\psi_2 = \{t \mid \Sigma_t \in \mathsf{Box}^x\}$.

Proof. Let $Z = \{t \mid \Sigma_t \in \mathsf{Box}^d\backslash\mathsf{Box}^x\}$, $V = \{t \mid \Sigma_t \in \mathsf{Box}^x\}$ and $U = V \cap \phi_1$. By the definition of dom_Ω^ϕ and lemma A.3(2), $\phi_1 = Z \cup U$ and $\phi_2 = V\backslash U$. By ${}^\bullet U \subseteq M_\phi$ and lemma A.3(2) and Ω being simple, U is a step of transitions enabled at $M = M_\phi$ and $M\ [U\rangle\ M'$, where $M' = (M\backslash{}^\bullet U) \cup U^\bullet$.

Define $\psi = (Z, V)$. Since ϕ is a factorisation, the sets ${}^\bullet t$ (for $t \in Z$) are mutually disjoint, and (also due to lemma A.3(2)), the sets $t^\bullet$ (for $t \in V$) are mutually disjoint. Suppose, ${}^\bullet t \cap u^\bullet \neq \emptyset$, where $t \in Z$ and $u \in V$. Then, since ϕ is a factorisation, $u \in U$. But this contradicts lemma A.3(2). Hence ψ is a factorisation. We now observe that the following holds:

$$\begin{array}{rcl}
M' & = & (M\backslash{}^\bullet U) \cup U^\bullet \\
 & = & (({}^\bullet(Z \cup U) \cup (V\backslash U)^\bullet)\backslash{}^\bullet U) \cup U^\bullet \\
 & = & (({}^\bullet Z \cup {}^\bullet U \cup (V\backslash U)^\bullet)\backslash{}^\bullet U) \cup U^\bullet \\
 & =_{\text{(i)}} & {}^\bullet Z \cup (V\backslash U)^\bullet \cup U^\bullet \\
 & =_{\text{(ii)}} & {}^\bullet Z \cup (V^\bullet\backslash U^\bullet) \cup U^\bullet \\
 & =_{\text{(iii)}} & {}^\bullet Z \cup V^\bullet.
\end{array}$$

In the above: (i) follows from $({}^\bullet Z \cup (V\backslash U)^\bullet) \cap {}^\bullet U = \emptyset$ which in turn follows from the definition of factorisation and $Z \cap U = \emptyset$; (ii) follows from $U^\bullet \cap (V\backslash U)^\bullet = \emptyset$ (see lemma A.3(2)); and (iii) follows from $U \subseteq V$. Hence ψ is a factorisation of $M' \in [{}^\circ\Omega\rangle$. Clearly, $\mathbf{\Sigma} \in \mathsf{dom}_\Omega^\psi$. ■ A.4

A.3 Properties of markings

We are now in a position to relate the marking of Σ_0 to those of the nets in $\mathbf{\Sigma}$ and factorisation ϕ.

Lemma A.5 Let $p \in S_0$ and $s \in S$.

(1) M_0 is a safe marking, and $M_0(p)$ is given by:

$$M_0(p) = \begin{cases} M_v(\mathsf{tree}^v(p)) & \text{if} \quad p \in \mathsf{ST}^v_{\mathsf{new}} \text{ and } v \in T \\ M_{\phi_s}(\mathsf{tree}^{\phi_s}(p)) & \text{if} \quad p \in \mathsf{SP}^s_{\mathsf{new}} \text{ and } \phi_s \neq \bot \\ 0 & \text{if} \quad p \in \mathsf{SP}^s_{\mathsf{new}} \text{ and } \phi_s = \bot. \end{cases}$$

(2) If $1 \in M_0(\mathsf{SP}^s_{\mathsf{new}})$ then $\phi_s \neq \bot$. Moreover, if $0 \notin M_0(\mathsf{SP}^s_{\mathsf{new}})$ then

$$\begin{array}{lll} \text{either} & \phi_s \in {}^\bullet s & \text{and} \quad \Sigma_{\phi_s} \in \mathsf{Box}^x \\ \text{or} & \phi_s \in s^\bullet & \text{and} \quad \Sigma_{\phi_s} \in \mathsf{Box}^e. \end{array}$$

Proof. (1) If $p \in \mathsf{ST}^v_{\mathsf{new}}$ then the $M_0(p) = M_v(\mathsf{tree}^v(p))$ follows directly from the definition of net refinement. Suppose $p \in \mathsf{SP}^s_{\mathsf{new}}$ and $\phi_s \neq \bot$. Then, by lemma A.3(5), $(({}^\bullet s \cup s^\bullet) \cap \phi) \setminus \{\phi_s\} \subseteq s^\bullet \cap \phi_2$. As a result, $M_t(\mathsf{tree}^t(p)) = 0$ for all $t \in ({}^\bullet s \cup s^\bullet) \setminus \{\phi_s\}$. Hence, $M_0(p) = M_{\phi_s}(\mathsf{tree}^{\phi_s}(p))$. Suppose $p \in \mathsf{SP}^s_{\mathsf{new}}$ and $\phi_s = \bot$. Then $M_t(\mathsf{tree}^t(p)) = 0$ for all $t \in {}^\bullet s \cup s^\bullet$, and we obtain $M_0(p) = 0$. Finally, $M_0(p) \in \{0, 1\}$ in all three cases since the markings of $\mathbf{\Sigma}$ are safe.

(2) The first part follows from (1). The second part can be proved thus. By (1) and the first part, $\phi_s \neq \bot$. Suppose $\phi_s \in {}^\bullet s$ and $\Sigma_{\phi_s} \notin \mathsf{Box}^x$. Then, since Σ_{ϕ_s} has a safe and clean marking, there is $r \in \Sigma^\circ_{\phi_s}$ such that $M_{\phi_s}(r) = 0$. Hence, by lemma A.1(4), there is $p \in \mathsf{SP}^s_{\mathsf{new}}$ such that $r = \mathsf{tree}^{\phi_s}(p)$. Thus, by (1), $M_0(p) = M_{\phi_s}(r) = 0$, a contradiction. The case when $\phi_s \in s^\bullet$ can be dealt with in a similar way. ■ A.5

Lemma A.6 Let $v \in T$.

(1) If $\Sigma_v \in \mathsf{Box}^e$ then

$$M_0\Big(\bigcup_{s \in {}^\bullet v} \mathsf{SP}^s_{\mathsf{new}}\Big) = \{1\} \text{ and } M_0\Big(\mathsf{ST}^v_{\mathsf{new}} \;\cup\; \bigcup_{s \in v^\bullet} \mathsf{SP}^s_{\mathsf{new}}\Big) = \{0\}.$$

(2) If $\Sigma_v \in \mathsf{Box}^x$ then

$$M_0\Big(\bigcup_{s \in v^\bullet} \mathsf{SP}^s_{\mathsf{new}}\Big) = \{1\} \text{ and } M_0(\mathsf{ST}^v_{\mathsf{new}}) = \{0\}.$$

Moreover, if $v \in \phi_1$ then

$$M_0\Big(\bigcup_{s \in {}^\bullet v} \mathsf{SP}^s_{\mathsf{new}}\Big) = \{0\}.$$

(3) M_0 is a clean marking.

Proof. (1,2) Follow from lemma A.5(1). Note that in (1), $v = \phi_s$ for all $s \in {}^\bullet v \cup v^\bullet$; and in (2), $v = \phi_s$ for all $s \in v^\bullet$. Moreover, if in (2) $v \in \phi_1$ then $v = \phi_s$ for all $s \in {}^\bullet v$.

(3) Suppose ${}^\circ\Sigma_0 \subseteq M_0$. Take any $s \in {}^\circ\Omega$. Then, by lemma A.5(2) and Ω being ex-directed, $\phi_s \in s^\bullet$ and $\Sigma_{\phi_s} \in \mathsf{Box}^e$. The latter implies $\phi_s \in \phi_1$. Hence ${}^\circ\Omega \subseteq M_\phi$ so, by lemma A.3(3), $M_\phi = {}^\circ\Omega = {}^\bullet\phi_1$ and $\phi_2 = \emptyset$. Moreover, by ${}^\circ\Omega \subseteq \bigcup_{s \in {}^\circ\Omega} {}^\bullet\phi_s$, $\phi_1 = \{\phi_s \mid s \in {}^\circ\Omega\}$. This and $\Sigma_{\phi_s} \in \mathsf{Box}^e$ (for $s \in {}^\circ\Omega$) and lemma A.5(1) yields $M_0 = {}^\circ\Sigma_0$. The case $\Sigma_0^\circ \subseteq M_0$ can be dealt with similarly after assuming $\phi_2 = \{t \mid \Sigma_t \in \mathsf{Box}^x\}$ (which can be done by lemma A.4), in order to be able to apply lemma A.3(4). ■ A.6

A.4 Properties of enabled transitions

Lemma A.7 Let $t \in \mathsf{T}^v_{\mathsf{new}}$ be enabled at M_0, i.e., ${}^\bullet t \subseteq M_0$.

(1) If $v \notin \phi_1$ then ${}^\bullet\mathsf{trees}(t) \subseteq {}^\circ\Sigma_v$ and $\bigcup_{s \in {}^\bullet v} \mathsf{SP}^s_{\mathsf{new}} \subseteq M_0$.

(2) $\Sigma_v \notin \mathsf{Box}^x$.

Proof. (1) Suppose $q \in {}^\bullet\mathsf{trees}(t) \cap (S_v \backslash ({}^\circ\Sigma_v \cup \Sigma_v^\circ))$. By lemma A.1(5) and A.5(1),we have $p = v \lhd q \in \mathsf{ST}^v_{\mathsf{new}}$ and $M_0(p) = M_v(q) = 0$. This, however, contradicts ${}^\bullet t \subseteq M_0$ and $p \in {}^\bullet t$. Hence ${}^\bullet\mathsf{trees}(t) \subseteq {}^\circ\Sigma_v$ since Σ_v is ex–directed.

Suppose $s \in {}^\bullet v$. Since Σ_v is T–restricted and ${}^\bullet\mathsf{trees}(t) \subseteq {}^\circ\Sigma_v$ and lemma A.1(4) holds, there is $p \in \mathsf{SP}^s_{\mathsf{new}}$ such that $p \in {}^\bullet t$ and so $M_0(p) \neq 0$. Hence, by lemma A.5(2), $\phi_s \neq \bot$. Clearly, $\phi_s \neq v$ since we assumed $v \notin \phi_1$ and $s \in {}^\bullet v$. Suppose $0 \in M_0(\mathsf{SP}^s_{\mathsf{new}})$. Then, by lemma A.5(1), there is $q \in S_{\phi_s}$ such that $M_{\phi_s}(q) = 0$ and $(\phi_s, q) \in {}^\diamond s \cup s^\diamond$. Take $r \in {}^\bullet\mathsf{trees}(t)$. Then $(v, r) \in s^\diamond$ and, by lemma A.1(7), there is $p' \in \mathsf{SP}^s_{\mathsf{new}}$ such that $\mathsf{tree}^{\phi_s}(p') = q$ and $\mathsf{tree}^v(p') = r$. Moreover, by lemma A.5(1), $M_0(p') = M_{\phi_s}(q) = 0$. But this produces a contradiction with ${}^\bullet t \subseteq M_0$ and $p' \in {}^\bullet t$. Hence $\mathsf{SP}^s_{\mathsf{new}} \subseteq M_0$.

(2) Suppose $\Sigma_v \in \mathsf{Box}^x$. If $v \in \phi_1$ then we obtain a contradiction between ${}^\bullet t \subseteq M_0$, and lemmata A.6(2) and A.1(4,5). Hence $v \in \phi_2$ and, by (1) and lemma A.5(2), for all $s \in {}^\bullet v$, $\phi_s \neq \bot$, and either (i) $\phi_s \in {}^\bullet s$ and $\Sigma_{\phi_s} \in \mathsf{Box}^x$, or (ii) $\phi_s \in s^\bullet$ and $\Sigma_{\phi_s} \in \mathsf{Box}^e$.

Define $U = \{\phi_s \mid s \in {}^\bullet v \wedge s \notin M_\phi\}$. Clearly, $U \subseteq \phi_1 \cap {}^\bullet({}^\bullet v)$ and (by (i)) $\Sigma_u \in \mathsf{Box}^x$, for $u \in U$. Hence, by lemma A.3(2), U is a step enabled at M_ϕ leading to some marking M. By the definition of factorisation, ${}^\bullet U \cap v^\bullet = \emptyset$. Suppose $s \in {}^\bullet u \cap {}^\bullet v$, for some $u \in U$. Then $u = \phi_s$ and (by (ii)) $\Sigma_u \in \mathsf{Box}^e$, a contradiction with $\Sigma_u \in \mathsf{Box}^x$. Hence ${}^\bullet U \cap {}^\bullet v = \emptyset$. Consequently, ${}^\bullet v \subseteq M$ and $v^\bullet \subseteq M$, contradicting lemma A.3(1). ■ A.7

Lemma A.8 If $t \in \mathsf{T}^v_{\mathsf{new}}$, $u \in \mathsf{T}^w_{\mathsf{new}}$, $v \neq w$ and $\{t, u\}$ is a step enabled at M_0 then $\{v, w\} \in \mathsf{Ind}_\Omega$.

Proof. Recall that Ω is a simple net, so it suffices to show that $({}^\bullet u \cup u^\bullet) \cap ({}^\bullet w \cup w^\bullet) = \emptyset$. By lemma A.4, we may assume: (i) $\phi_2 = \{z \mid \Sigma_z \in \mathsf{Box}^x\}$. Hence, by lemma A.7(2), we have: (ii) $v, w \notin \phi_2$. Moreover, if $v \notin \phi_1$ then, by lemma A.7(1), we have: (iii) ${}^\bullet\mathsf{trees}(t) \subseteq {}^\circ\Sigma_v$ and $\mathsf{SP}^s_{\mathsf{new}} \subseteq M_0$ for all $s \in {}^\bullet v$. This yields, by (i) and lemma A.5(2), that: (iv) $v \notin \phi_1$ implies ${}^\bullet v \subseteq M_\phi$. We now consider three cases (note that $v, w \notin \phi_2$ due to (ii)):

Case 1: $v, w \in \phi_1$. Then $\{v, w\} \in \mathsf{Ind}_\Omega$ follows from lemma A.3(2).

Case 2: $v, w \notin \phi$. Then, by (iv), ${}^\bullet v \subseteq M_\phi$ and, by symmetry, ${}^\bullet w \subseteq M_\phi$. Moreover, ${}^\bullet v \cap {}^\bullet w = \emptyset$, which follows from: lemma A.2(2), (iii) (the first part) and its counterpart for w, M_0 beinge safe and $M_0\,[\{t, u\}\rangle$. Hence, by proposition 3.1, $\{v, w\} \in \mathsf{Ind}_\Omega$.

Case 3: $v \notin \phi$ and $w \in \phi_1$. Then, ${}^\bullet w \subseteq M_\phi$, and ${}^\bullet v \subseteq M_\phi$ (the latter by (iv)). Suppose $s \in {}^\bullet v \cap {}^\bullet w$. Then, by lemma A.7(1), $\mathsf{SP}^s_{\mathsf{new}} \subseteq M_0$. Hence, by lemma A.5(2) and $w \in \phi_1$, we have $\phi_s = w$ and $\Sigma_w \in \mathsf{Box}^e$. We then observe that by ${}^\bullet u \subseteq M_0$, lemma A.5(1) and A.1(5), we have ${}^\bullet\mathsf{trees}(u) \subseteq {}^\circ\Sigma_w$. Hence, by lemma A.2(2) and (iii) (the first part), we obtain ${}^\bullet t \cap {}^\bullet u \neq \emptyset$, a contradiction with M_0 being safe and $M_0\,[\{t, u\}\rangle$. Hence ${}^\bullet v \cap {}^\bullet w = \emptyset$ and we obtain $\{v, w\} \in \mathsf{Ind}_\Omega$ by proposition 3.1. ■ A.8

Lemma A.9 Let $v \in \widetilde{\phi}$ and $t \in \mathsf{T}^v_{\mathsf{new}}$.

(1) t is enabled at M_0 if and only if $\mathsf{trees}(t)$ is a step enabled at M_v. Moreover, if t is enabled at M_0 then $\mathsf{trees}(t) \in \mathsf{Ind}_{\Sigma_v}$ and $\{t\} \in \mathsf{Ind}_{\Sigma_0}$.

(2) If $\mathsf{op}_\Omega(\mathbf{\Sigma})\,[\{t\}\rangle\,\Sigma$ and $\Sigma_v\,[\mathsf{trees}(t)\rangle\,\Theta_v$ then $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\Sigma = \mathsf{op}_\Omega(\mathbf{\Theta})$, where $\Theta_w = \Sigma_w$ for all $w \neq v$.

Proof. (1) Suppose t is enabled at M_0. By lemma A.7(2):

(i) $v \in \phi_1$ and $\phi_s = v$, for all $s \in {}^\bullet v \cup v^\bullet$.

Suppose $u \in \mathsf{trees}(t)$ and $q \in {}^\bullet u$. By lemma A.1(2,6), there is $p \in {}^\bullet t$ such that $q = \mathsf{tree}^v(p)$ and $p \in \mathsf{ST}^v_{\mathsf{new}} \cup \{\mathsf{SP}^s_{\mathsf{new}} \mid s \in {}^\bullet v\}$. Hence, since M_0 is safe and t is enabled: (ii) $W_v(q,u) \leq W_0(p,t) \leq M_0(p) = 1$. For the same reasons, if $w \in \mathsf{trees}(t)$ and $w \neq v$ then $W_v(q,w){+}W_v(q,u) \leq 1$. Hence, (iii) ${}^\bullet w \cap {}^\bullet v = \emptyset$. We then observe that, by lemma A.5(1) and (i) and ${}^\bullet t \subseteq M_0$, $M_0(p) = M_v(q) = 1$. Hence ${}^\bullet\mathsf{trees}(t) \subseteq M_v$ which, together with (ii) and (iii), means that $\mathsf{trees}(t)$ is a step enabled at M_v. As a result, by proposition 3.1, $\mathsf{trees}(t) \in \mathsf{Ind}_{\Sigma_v}$. This and the definition of W_0 yields $\{t\} \in \mathsf{Ind}_{\Sigma_0}$.

Suppose now that $\mathsf{trees}(t)$ is enabled at M_v. Then, by proposition 3.1, $\mathsf{trees}(t) \in \mathsf{Ind}_{\Sigma_v}$ and so, by the definition of W_0, $\{t\} \in \mathsf{Ind}_{\Sigma_0}$. Hence to show that t is enabled at M_0 it suffices to prove that ${}^\bullet t \subseteq M_0$. From Σ_v being ex–directed it follows $\Sigma_v \notin \mathsf{Box}^x$. Hence (i) holds again. Suppose $p \in {}^\bullet t$. Then, by (i) and lemma A.5(1), $\mathsf{tree}^v(p) \in {}^\bullet\mathsf{trees}(t)$ and $M_0(p) = M_v(\mathsf{tree}^v(p))$. Hence, since $\mathsf{trees}(t)$ is enabled at M_v, $M_0(p) = 1$.

(2) We first observe that $\Sigma_v \notin \mathsf{Box}^x$, hence (i) holds again. By $v \in \phi_1$, $\mathbf{\Theta} \in \mathsf{dom}^\phi_\Omega$. To prove $\Sigma = \mathsf{op}_\Omega(\mathbf{\Theta})$ it suffices to show $M_1(p) = M_2(p)$, for every $p \in S_0$, where M_1 and M_2 are respectively the markings of Σ and $\mathsf{op}_\Omega(\mathbf{\Theta})$. Suppose

$$p \notin R \;=\; \mathsf{ST}^v_{\mathsf{new}} \cup \bigcup_{s \in {}^\bullet v \cup v^\bullet} \mathsf{SP}^s_{\mathsf{new}}.$$

Then $M_1(p) = M_2(p)$ follows from lemmata A.1(2) and A.5(1). If $p \in R$ then the thesis follows from (i), the second part of (1), lemma A.5(1) and

$$\begin{array}{rcl} p \in {}^\bullet t & \Leftrightarrow & \mathsf{tree}^v(p) \cap {}^\bullet\mathsf{trees}(t) \neq \emptyset \\ \text{and} \quad p \in t^\bullet & \Leftrightarrow & \mathsf{tree}^v(p) \cap \mathsf{trees}(t)^\bullet \neq \emptyset, \end{array}$$

for all $t \in \mathsf{T}^v_{\mathsf{new}}$. ■ A.9

Lemma A.10 If $\mathsf{op}_\Omega(\mathbf{\Sigma})\ [U\rangle\ \Sigma$ and $(\Omega : \mathbf{\Sigma}) \xrightarrow{U} (\Omega : \mathbf{\Theta})$ then $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Theta}) = \Sigma$.

Proof. (By induction on $|U|$.) The base case with $|U| = 0$ is obvious. In the induction step, suppose $t \in U$ and $V = U \backslash \{t\}$. We have $\mathsf{op}_\Omega(\mathbf{\Sigma})\ [\{t\}\rangle\ \Sigma'\ [V\rangle\ \Sigma$ and $(\Omega : \mathbf{\Sigma}) \xrightarrow{\{t\}} (\Omega : \mathbf{\Gamma})$. From lemma A.9(2) it follows that $\mathbf{\Gamma} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Gamma}) = \Sigma'$. Moreover, since the formula for the move of Σ_v in the definition of $\xrightarrow{U}$ employs a disjoint union of independent steps, we have $(\Omega : \mathbf{\Gamma}) \xrightarrow{V} (\Omega : \mathbf{\Theta})$. Thus, by applying the induction hypothesis for V and $\mathbf{\Gamma}$, we get $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Theta}) = \Sigma$. ■ A.10

Lemma A.11 If $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$ then $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Sigma}) = \mathsf{op}_\Omega(\mathbf{\Theta})$.

Proof. Let $eta = (\eta_1, \eta_2)$ and $\psi = (\psi_1, \psi_2)$ be two disjoint factorisations from the definition of $\equiv_\Omega$. By lemma A.4, we may assume that $\phi_2 = \{t \mid \Sigma_t \in \mathsf{Box}^x\}$. Hence $\eta_1 \subseteq \phi_1$ and $\eta_2 \subseteq \phi_2$. Let $\kappa = ((\phi_1 \setminus \eta_1) \cup \psi_1, (\phi_2 \setminus \eta_2) \cup \psi_2)$. By lemma A.3(6), κ is a factorisation of M_ϕ and, clearly, $\mathbf{\Theta} \in \mathsf{dom}^\kappa_\Omega$. We have $\lfloor \mathsf{op}_\Omega(\mathbf{\Sigma}) \rfloor = \lfloor \mathsf{op}_\Omega(\mathbf{\Theta}) \rfloor$, so in order to prove $\mathsf{op}_\Omega(\mathbf{\Sigma}) = \mathsf{op}_\Omega(\mathbf{\Theta})$, we only need to show $M_0 = M_1$, where M_1 is the marking of $\mathsf{op}_\Omega(\mathbf{\Theta})$. By lemma A.5(1) and $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$, if $p \in \mathsf{ST}^v_{\mathsf{new}}$ (for $v \in T$) or $p \in \mathsf{SP}^s_{\mathsf{new}}$ (for $s \in S$ satisfying $\phi_s = \kappa_s$), then $M_0(p) = M_1(p)$.

Define $R = {}^\bullet\eta_1 \cup \eta_2^\bullet = {}^\bullet\psi_1 \cup \psi_2^\bullet$, and suppose that $s \in S$ and $p \in \mathsf{SP}^s_{\mathsf{new}}$, where $\phi_s \neq \kappa_s$. By lemma A.5(1) and $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$, we have:

(i) $s \in R \Rightarrow M_0(p) = M_1(p) = 1$.

(ii) $(s \notin R \wedge \phi_s \in \widetilde{\eta}) \Rightarrow M_0(p) = 0$.

(iii) $(s \notin R \wedge \kappa_s \in \widetilde{\psi}) \Rightarrow M_1(p) = 0$.

Suppose $\phi_s \neq \bot = \kappa_s$. Then $M_1(p) = 0$ (by lemma A.5(1)) and $\phi_s \in \widetilde{\eta}$ and $s \notin R$. Hence, by (ii), $M_0(p) = 0$. The case $\phi_s = \bot \neq \kappa_s$ is symmetric. Suppose $\bot \neq \phi_s \neq \kappa_s \neq \bot$. Then $\phi_s \in \eta$ and $\kappa_s \in \psi$. If $s \in R$ then, by (i), $M_0(p) = M_1(p) = 1$. Otherwise, $M_0(p) = M_1(p) = 0$, by (ii) and (iii). ■ A.11

A.5 Proofs of three compositionality theorems

Proof of Theorem 3.2. First note that $\mathbf{\Theta} \in \mathsf{dom}_\Omega^\phi$ since if $\Sigma_v \in \mathsf{Box}^x$ then $\Theta_v = \Sigma_v$ as no move is possible for Σ_v. Suppose that U is the union of sets U_v, for $v \in T$, such that $U_v = \{u_v^1, \ldots, u_v^{k_v}\}$ and $\Sigma_v\ [V_v\rangle\ \Theta_v$, where $V_v = \mathsf{trees}(u_v^1) \uplus \cdots \mathsf{trees}(, u_v^{k_v})$. From proposition 3.1 it follows that $V_v \in \mathsf{Ind}_{\Sigma_v}$, for $v \in T$. Since V_v is the disjoint union, by lemma A.2(1) and the definition of net refinement, we obtain $U_v \in \mathsf{Ind}_{\Sigma_0}$ for every $v \in T$. Suppose $\{u_v^j, u_w^k\} \notin \mathsf{Ind}_{\Sigma_0}$ for some $v \neq w$. Then, by lemma A.2(2), there is $s \in S$ such that $v, w \in {}^\bullet s \cup s^\bullet$. On the other hand, since V_v and V_w are non-empty enabled steps, $\Sigma_v, \Sigma_w \in \mathsf{Box}^d \backslash \mathsf{Box}^x$. Hence $v, w \in \phi_1$ and we obtain a contradiction with lemma A.3(2). Hence $\{u_v^j, u_w^k\} \in \mathsf{Ind}_{\Sigma_0}$ which, together with $U_v \in \mathsf{Ind}_{\Sigma_0}$, for all v, and U being finite, yields $U \in \mathsf{Ind}_{\Sigma_0}$. Moreover, by lemma A.9(1), ${}^\bullet U \subseteq M_0$. Hence U is a step enabled at M_0 and by lemma A.10 we obtain $\mathsf{op}_\Omega(\mathbf{\Sigma})\ [U\rangle\ \mathsf{op}_\Omega(\mathbf{\Theta})$. ■ 3.2

Proof of Theorem 3.3. Define the following sets of transitions of Ω:

$$\begin{array}{rcl} V^0 &=& \{v \mid U \cap \mathsf{T}_{\mathsf{new}}^v \neq \emptyset \wedge \Sigma_v \in \mathsf{Box}^s\} \\ V^1 &=& \{v \mid U \cap \mathsf{T}_{\mathsf{new}}^v \neq \emptyset \wedge \Sigma_v \notin \mathsf{Box}^s\} \\ V^2 &=& \{v \mid \Sigma_v \in \mathsf{Box}^x\} \\ V^3 &=& \{v \mid U \cap \mathsf{T}_{\mathsf{new}}^v = \emptyset \wedge \Sigma_v \in \mathsf{Box}^e\} \\ V^4 &=& \{v \mid U \cap \mathsf{T}_{\mathsf{new}}^v = \emptyset \wedge \Sigma_v \notin \mathsf{Box}^s \cup \mathsf{Box}^e \cup \mathsf{Box}^x\}. \end{array}$$

The five sets are mutually disjoint ($V^1 \cap V^2 = \emptyset$ follows from lemma A.7(2)). Moreover, by lemmata A.4 and A.7(2), we may assume that:

(i) $\phi_1 = V^1 \cup V^3 \cup V^4$ and $\phi_2 = V^2$ and $\Sigma_v \notin \mathsf{Box}^x$ for $v \in V^1$.

Let $V' = V^0 \cup V^1 \cup V^4$. We will show that there are $\kappa = (\kappa_1, \kappa_2)$ and $\mathbf{\Theta}$ such that $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$ and $V' \subseteq \kappa_1$ and $\mathbf{\Theta} \in \mathsf{dom}_\Omega^\kappa$.

We first show ${}^\bullet V' \subseteq M_\phi$. Clearly, ${}^\bullet V^1 \cup {}^\bullet V^4 \subseteq M_\phi$. Suppose $v \in V^0$ and $s \in {}^\bullet v$. By lemma A.7(1), $\mathsf{SP}_{\mathsf{new}}^s \subseteq M_0$. Hence, by lemma A.5(2) and (i), there is w such that $w \in ({}^\bullet s \cap \phi_2) \cup (s^\bullet \cap \phi_1)$. Thus $M_\phi(s) = 1$.

We next prove $V' \subseteq \mathsf{Ind}_\Omega$. By lemma A.8, $V^0 \cup V^1 \in \mathsf{Ind}_\Omega$, and by (i) and lemma A.3(2), $V^1 \cup V^4 \subseteq \mathsf{Ind}_\Omega$. Let $t \in V^0$ and $u \in V^4$. Suppose $s \in {}^\bullet t \cap {}^\bullet u$. Then, by lemma A.7(1), $\mathsf{SP}_{\mathsf{new}}^s \subseteq M_0$. Hence, by (i) and lemma A.5(2), $\Sigma_u \in \mathsf{Box}^e$, a contradiction. Hence ${}^\bullet t \cap {}^\bullet u = \emptyset$. Thus, by ${}^\bullet V' \subseteq M_\phi$ and proposition 3.1, $\{t, u\} \in \mathsf{Ind}_\Omega$. As a result, $V' \in \mathsf{Ind}_\Omega$. Hence since Ω is factorisable, there is a factorisation $\kappa = (\kappa_1, \kappa_2)$ such that $V' \subseteq \kappa_1$ and $M_\kappa = M_\phi$. From the definition of factorisation and $M_\kappa = M_\phi$ it follows that

$$\begin{array}{rcccccl} \eta &=& (\eta_1, \eta_2) &=& (\phi_1 \backslash (\phi_1 \cap \kappa_1), \phi_2 \backslash (\phi_2 \cap \kappa_2)) &=& (\phi_1 \backslash \kappa_1, \phi_2 \backslash \kappa_2) \\ \text{and}\quad \psi &=& (\psi_1, \psi_2) &=& (\kappa_1 \backslash (\phi_1 \cap \kappa_1), \kappa_2 \backslash (\phi_2 \cap \kappa_2)) &=& (\kappa_1 \backslash \phi_1, \kappa_2 \backslash \phi_2) \end{array}$$

are factorisations of the same set of places. We further observe that $\kappa_1 \cap \phi_2 = \kappa_2 \cap \phi_1 = \emptyset$, by lemma A.3(1), so $\widetilde{\eta} = \widetilde{\phi} \backslash \widetilde{\kappa}$ and $\widetilde{\psi} = \widetilde{\kappa} \backslash \widetilde{\phi}$. In particular, this implies $\widetilde{\eta} \cap \widetilde{\psi} = \emptyset$, so η and ψ are disjoint factorisations. Define $\mathbf{\Theta}$ by $\lfloor \mathbf{\Theta} \rfloor = \lfloor \mathbf{\Sigma} \rfloor$ and $\Theta_v = \Sigma_v$ for $v \notin \widetilde{\psi} \cup \widetilde{\eta}$ and:

$$\begin{array}{rrcl} & \mathbf{\Theta}_{\psi_1} &\subseteq& \mathsf{Box}^e \\ & \mathbf{\Theta}_{\psi_2} &\subseteq& \mathsf{Box}^x \\ \text{and} & \mathbf{\Theta}_{\widetilde{\eta}} &\subseteq& \mathsf{Box}^s. \end{array}$$

We have the following: if $v \in \eta_1$ then (by $V' \subseteq \kappa_1$) $v \in V^3$, so $\Sigma_v \in \mathsf{Box}^e$; if $v \in \eta_2$ then $v \in V^2$, so $\Sigma_v \in \mathsf{Box}^x$; and if $v \in \widetilde{\psi}$ then $v \in \widetilde{\kappa} \backslash \widetilde{\phi}$, so $\Sigma_v \in \mathsf{Box}^s$. Hence $\mathbf{\Sigma} \equiv_\Omega \mathbf{\Theta}$ and so, by lemma A.11, $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Sigma}) = \mathsf{op}_\Omega(\mathbf{\Theta})$. For every $v \in V_0 \cup V_1$, let $U_v = U \cap \mathsf{T}_{\mathsf{new}}^v$. Since each U_v is a step, by lemma A.2(1)

$$V_v = \bigcup_{t \in U_v} \mathsf{trees}(t) = \biguplus_{t \in U_v} \mathsf{trees}(t) \in \mathsf{Ind}_{\Sigma_v}.$$

Moreover, by lemma A.9(1), V_v is enabled at Θ_v. Hence $(\Omega : \mathbf{\Theta}) \xrightarrow{U} (\Omega : \mathbf{\Gamma})$ for some $\mathbf{\Gamma}$, which, by lemma A.10, yields $\mathbf{\Gamma} \in \mathsf{dom}_\Omega$ and $\mathsf{op}_\Omega(\mathbf{\Gamma}) = \Sigma$. ■ 3.3

Proof of Theorem 3.4. In view of lemmata A.1(8), A.5(1) and A.6(3), to prove that all markings reachable from M_0 are safe and clean, it suffices to show that if U is a step such that $\mathsf{op}_\Omega(\mathbf{\Sigma})\ [U\rangle\ \Sigma$ then there is $\mathbf{\Theta} \in \mathsf{dom}_\Omega$ satisfying $\Sigma = \mathsf{op}_\Omega(\mathbf{\Theta})$, which holds by theorem 3.3 and lemma A.11. Note that we still need to show that $\lfloor \mathsf{op}_\Omega(\mathbf{\Sigma}) \rfloor$ is a static box, but this follows from the considerations below. ■ 3.4

For all three theorems we still need to consider the case when $\mathbf{\Sigma} \in (\mathsf{Box}^s)^n$. Clearly, for theorems 3.2 and 3.3, the thesis trivially holds since all the nets involved have empty markings. For theorem 3.4 we proceed thus:

We need to show that all markings reachable from the initial marking of $\mathsf{op}_\Omega(\mathbf{\Sigma})$ are safe and clean. We first observe that, since Ω is an operator box, there is a factorisation $\phi = (\phi_1, \phi_2)$ of ${}^\circ\Omega$ and, by lemma A.3(3), $\phi_2 = \emptyset$. Let $\mathbf{\Theta}$ be such that $\Theta_t = \overline{\Sigma_t}$ for $t \in \phi_1$, and $\Theta_t = \Sigma_t$ otherwise. Clearly, $\mathbf{\Theta} \in \mathsf{dom}_\Omega^\phi$ and, by lemma A.6(1) and ${}^\bullet\phi_1 = {}^\circ\Omega$, we have $\mathsf{op}_\Omega(\mathbf{\Theta}) = \overline{\mathsf{op}_\Omega(\mathbf{\Sigma})}$. We now can apply the part of the theorem which we have already shown to hold.

B Appendix: Proofs of section 4

B.1 Properties of the limit construction

We first formulate an auxiliary lemma.

Lemma B.1 Let $\chi = (\Sigma_0, \Sigma_1, \ldots)$ be a chain in Box^{rec}, and Σ and Θ be boxes in Box^{rec}.

(1) If $\Sigma \sqsubseteq \bigsqcup \chi$ is a finite box then there is $k \geq 0$ such that $\Sigma \sqsubseteq \Sigma_k$.

(2) Suppose that $\Sigma \sqsubseteq \Theta$ and $\overline{\Theta}\ [\sigma\rangle\ \Phi$ and all the transitions occurring in step sequence σ belong to Σ. Then $\overline{\Sigma}\ [\sigma\rangle\ \Psi$, for some Ψ. Moreover, $M_\Psi(s) = M_\Phi(s)$ for each place s in Σ.

Proof. (1) For every place or transition of Σ there is Σ_i in χ which contains it. Define k as the maximum of all such i.

(2) Directly by the definitions of inclusion order and step sequence semantics. ■ B.1

Proof of Theorem 4.2. Let $\chi = (\Sigma_0, \Sigma_1, \ldots)$. It is clear that $\Sigma = \bigsqcup \chi$ is a box. What we need to show is that if $\overline{\Sigma}\ [\sigma\rangle\ \Theta$ then M_Θ is a safe and clean marking. To the contrary, suppose that $M_\Theta(s) > 1$. Let U be the set of transitions occurring in the step sequence σ. Since $\{s\} \cup U$ is a finite set, there is a finite box $\Psi \sqsubseteq \Sigma$ such that $\{s\} \cup U$ belong to Ψ. Hence, by lemma B.1, there is $k \geq 0$ such that $\overline{\Sigma}_k\ [\sigma\rangle\ \Phi$ and $M_\Phi(s) = M_\Psi(s) > 1$. This, however, contradicts Σ_k being a static box. Suppose now that M_Θ is not clean. Then, since Σ is ex–directed and T-restricted and safe, $\Sigma^\circ \subseteq M_\Theta$ and $M_\Theta(s) > 0$, for some $s \notin \Sigma^\circ$. By proceeding as before, we can obtain a contradiction with the fact that all markings reachable from the markings ${}^\circ\Sigma_k$ are clean. ■ 4.2

Proof of Theorem 4.3. That F is monotonic follows directly from the definition of the net refinement and the inclusion order. To show that it is also continuous, suppose $\chi = (\Sigma^0, \Sigma^1, \ldots)$ is a chain in Box^{rec}. Let $\Sigma = \bigsqcup \chi$ and $\Theta = \bigsqcup(\mathsf{F}(\Sigma^0), \mathsf{F}(\Sigma^1), \ldots)$. We need to show that $\Theta = \mathsf{F}(\Sigma)$.

Since $\Sigma^i \sqsubseteq \Sigma$ and F is monotonic, $\mathsf{F}(\Sigma^i) \sqsubseteq \mathsf{F}(\Sigma)$, for all $i \geq 0$, and thus $\Theta \sqsubseteq \mathsf{F}(\Sigma)$. To show the reverse inclusion, we first observe that the set of transitions of $\mathsf{F}(\Sigma)$ is included in Θ. Indeed, if $(v, a) \lhd Q \in T_{\mathsf{F}(\Sigma)}$ then there are only finitely many transitions in Q, and hence there is $i \geq 0$ such that all the transitions in Q belong to Σ^i. As a result, $(v,a) \lhd Q \in T_{\mathsf{F}(\Sigma^i)}$ and thus $T_{\mathsf{F}(\Sigma)} \subseteq T_\Theta$. To demonstrate that a similar inclusion holds for the sets of places we observe that to define any $p \in S_{\mathsf{F}(\Sigma)}$ we only need finitely many places and therefore there is $i \geq 0$ such that all these places belong to Σ^i and hence $p \in S_{\mathsf{F}(\Sigma^i)}$. Thus $S_{\mathsf{F}(\Sigma)} \subseteq S_\Theta$. Clearly, the labels and markings of the transitions and places common to $\mathsf{F}(\Sigma)$ and Θ are the same. Showing that the weight functions are also consistent is rather straightforward. Suppose that $u \in T_{\mathsf{F}(\Sigma)}$ and $p \in S_{\mathsf{F}(\Sigma)}$. Then, from what we have already proved, it follows that there is $i \geq 0$ such that $u \in T_{\mathsf{F}(\Sigma^i)}$ and $p \in S_{\mathsf{F}(\Sigma^i)}$. Let A and B denote the sets of transition and place trees of Σ and Σ^i which have been used to construct respectively u and p. By the definition of net refinement, these sets are exactly the same in the case of both Σ and Σ^i. Hence $W_{\mathsf{F}(\Sigma)}(p,u) = W_{\mathsf{F}(\Sigma^i)}(p,u)$ and $W_{\mathsf{F}(\Sigma)}(u,p) = W_{\mathsf{F}(\Sigma^i)}(u,p)$ which follows directly from the way the weight function is calculated in the definition of net refinement. Thus $W_{\mathsf{F}(\Sigma)} \subseteq W_\Theta$. ■ 4.3

We now prove that all solutions of equation (4) have the same sets of transitions, and that the inclusion order $\sqsubseteq$ on the set of all possible solutions is equivalent to the ordering resulting from the ordinary set inclusion on their entry and exit places.

Lemma B.2 Let Σ and Θ be solutions of equation (4).

(1) The sets of transitions of Σ and Θ are the same.

(2) The following are equivalent:

(a) ${}^\circ\Sigma \subseteq {}^\circ\Theta$ and $\Sigma^\circ \subseteq \Theta^\circ$

(b) $\Sigma \sqsubseteq \Theta$.

Proof. Let $\Sigma = (S_0, T_0, W_0, \lambda_0)$ and $\Theta = (S_1, T_1, W_1, \lambda_1)$.
(1) Suppose $T_0 \not\subseteq T_1$. Then, by $\Sigma = \mathsf{op}_\Omega(\mathbf{\Delta}^\Sigma)$, there is $t = (v,a) \lhd Q$ which is a smallest transition tree in $T_0 \backslash T_1$ (we can assume that t exists since all transition trees are finite). If Δ_v is a box then Q is a set of transitions of Δ_v and, since $\Delta_v^\Sigma = \Delta_v^\Theta$ and $\Theta = \mathsf{op}_\Omega(\mathbf{\Delta}^\Theta)$, we have $t \in T_1$, leading to a contradiction. If $\Delta_v = X$ then $Q \subseteq T_0$. If $Q \subseteq T_1$ then, by $\Theta = \mathsf{op}_\Omega(\mathbf{\Delta}^\Theta)$ and $\Delta_v^\Theta = \Theta$, $t \in T_1$ contradicting the choice of t. Hence there is $u \in Q$ such that $u \notin T_1$. Moreover, u is a proper sub-tree of t, which produces a contradiction with the minimality of t. Hence $T_0 \subseteq T_1$. By symmetry, $T_1 \subseteq T_0$. Thus $T_0 = T_1$.
(2) Since the (a) $\Leftarrow$ (b) implication is obvious, we only prove that (a) $\Rightarrow$ (b) holds. We first show that the places of Σ are also places of Θ. Since every place is a tree of the form $t_1 \lhd \cdots \lhd t_k \lhd \tau$, where $t_1, \ldots, t_k \in \mathsf{T_{root}}$ and τ is a tree with the root labelled with a primitive name in $\mathsf{P_{root}}$ (see section 2.2.2), we can proceed by induction on k. For $k = 0$, we have $p = \tau$. Thus, by the definition of net refinement and $\Sigma = \mathsf{op}_\Omega(\mathbf{\Delta}^\Sigma)$, p is as in the part (b) of the definition of net refnement in section 2.2.5. Since we have ${}^\circ(\Delta_v^\Sigma) \subseteq {}^\circ(\Delta_v^\Theta)$ and $(\Delta_v^\Sigma)^\circ \subseteq (\Delta_v^\Theta)^\circ$, for all $v \in T_\Omega$, p is also a place in $\mathsf{op}_\Omega(\mathbf{\Delta}^\Theta) = \Theta$ with the same label as in Σ (i.e., $\lambda_\Omega(s)$). The induction step for $k > 0$ is similar to that made for transitions as in part (1). And so is the proof that $W_0 \subseteq W_1$. ■ B.2

Proof of Theorem 4.5. (1) Follows from lemma B.2(2) and $\sqsubseteq$ being a partial order.
(2) Clearly, Σ^0 is a static box. Moreover, by $\Sigma = \mathsf{F}(\Sigma)$,

$$\begin{array}{rl} & {}^\circ(\Sigma^0) = {}^\circ(\Sigma^1) = \ldots = {}^\circ\Sigma \\ \text{and} & (\Sigma^0)^\circ = (\Sigma^1)^\circ = \ldots = \Sigma^\circ. \end{array}$$

Hence $\Sigma^0 \sqsubseteq \Sigma^1$ and Σ^0 is a seed box. What is more,

$${}^\circ(\bigsqcup \mathsf{chain}(\Sigma^0)) = {}^\circ\Sigma \text{ and } (\bigsqcup \mathsf{chain}(\Sigma^0))^\circ = \Sigma^\circ.$$

Hence, by lemma B.2(2), $\bigsqcup \mathsf{chain}(\Sigma^0) = \Sigma$. ■ 4.5

B.2 Properties of seed boxes

We first show that (10) is sufficient to characterise all seed boxes.

Lemma B.3 Let Σ^0 be as in (8). Then the following are equivalent:

(1) Σ^0 is a seed box.

(2) The sets $\mathsf{P_e}$ and $\mathsf{P_x}$ are nonempty and satisfy (10).

Proof. If $\Sigma^0 \sqsubseteq \Sigma^1 = \mathsf{F}(\Sigma^0)$ then, by the definition of $\sqsubseteq$, ${}^\circ(\Sigma^0) \subseteq {}^\circ(\Sigma^1)$ and $(\Sigma^0)^\circ \subseteq (\Sigma^1)^\circ$. Hence, by the definition of net refinement, $\mathsf{P_e} \subseteq \mathsf{f_e}(\mathsf{P_e})$ and $\mathsf{P_x} \subseteq \mathsf{f_x}(\mathsf{P_x})$. The reverse implication holds by the definition of net refinement. ■ B.3

Lemma B.4 The functions $\mathsf{f_e}$ and $\mathsf{f_x}$ defined by (9) are monotonic and continuous.

Proof. We only show that the result holds for $\mathsf{f_e}$, the other case being symmetric. Monotonicity follows directly from the definition of $\mathsf{f_e}$ and the fact that $\cup$ and $\lhd$ (in the second argument) are monotonic in the domain $(2^{\mathsf{P_{tree}}}, \subseteq)$. To show $\mathsf{f_e}$ is continuous, let $P_0 \subseteq P_1 \subseteq \dots$ be a chain in $(2^{\mathsf{P_{tree}}}, \subseteq)$. The inclusion

$$\mathsf{f_e}(\bigcup_{i=0}^{\infty} P_i) \supseteq \bigcup_{i=0}^{\infty} \mathsf{f_e}(P_i)$$

follows from monotonicity. As for the other inclusion, suppose that

$$p = s_i \lhd \{v_i^1 \lhd p_i^1, \dots, v_i^{k_i} \lhd p_i^{k_i}\} \in \mathsf{f_e}(\bigcup_{i=0}^{\infty} P_i).$$

Since there are only finitely many trees p_i^l in the definition of p, from the definition of chain it follows that there is $m \geq 1$ such that each p_i^l belongs either to P_m (if $\Delta_{v_i^l} = X$), or to $P_i^l = {}^\circ(\Delta_{v_i^l})$ (if $\Delta_{v_i^l}$ is a box). Hence $p \in \mathsf{f_e}(P_m)$. ■ B.4

We finally formulate and prove three properties of fixpoints of the two functions, $\mathsf{f_e}$ and $\mathsf{f_x}$, defined by (9). The lemma is formulated for $\mathsf{f_e}$, and the results for $\mathsf{f_x}$ follow by symmetry.

Lemma B.5 The following are satisfied

(1) $\mathsf{min_e}$ is a countable set.

(2) $\mathsf{min_e}$ contains only finite place trees if the place trees appearing in the nets of $\mathbf{\Delta}$ in (4) are finite.

(3) $\mathsf{max_e}$ is a nonempty set.

Proof. (1,2) Define $P_0 = \emptyset$ and $P_{i+1} = \mathsf{f_e}(P_i)$ for $i \geq 0$. As $\mathsf{f_e}$ is continuous, $\mathsf{min_e} = \bigcup_{i=0}^{\infty} P_i$. If every box in the recursive equation contains only finite place trees, so does $\mathsf{min_e}$. Moreover, $\mathsf{min_e}$ is countable because it is the countable union of finite sets.

(3) Construct $\mathsf{spine_e}$ as in construction 4.6. Then $\{\mathsf{spine_e}\} \subseteq \mathsf{f_e}(\{\mathsf{spine_e}\})$. Hence $\mathsf{spine_e} \in \mathsf{max_e}$. ■ B.5

Proof of Theorem 4.8. In the guarded case, both $\mathsf{f_e}$ and $\mathsf{f_x}$ are constant functions. Hence, if Σ and Θ are two solutions of (4) then ${}^\circ\Sigma = {}^\circ\Theta$ and $\Sigma^\circ = \Theta^\circ$. Thus, by theorem 4.5(1), $\Sigma = \Theta$. ■ 4.8

C Appendix: Proofs of section 5

C.1 Static properties of the $\mathsf{box}(.)$ mapping

Lemma C.1 (1) $\mathsf{box}(\mathsf{Expr}^\delta) \subseteq \mathsf{Box}^\delta$, for $\delta \in \{s,d,e,x\}$.

(2) $\mathsf{box}(\mathsf{Dom}_\Omega) \subseteq \mathsf{dom}_\Omega$ and $\mathsf{box}(\mathsf{Dom}_\Omega^\mu) \subseteq \mathsf{dom}_\Omega^\mu$ for all $\mu \in \mathsf{fact}_\Omega$.

Proof. (1) For Expr^s the result follows from theorem 3.4 and $\mathsf{box}(\mathsf{Lab} \cup \mathsf{var}) \subseteq \mathsf{Box}^s$. For Expr^e, Expr^x and Expr^d we proceed by induction on the structure of a dynamic expression H. The base case is $H = \overline{E}$ or $H = \underline{E}$, where $E \in \mathsf{Expr}^s$. We only deal with the former case. Then, it follows from the definition of the syntax, that $H \notin \mathsf{Expr}^x$. Moreover, by $\mathsf{box}(\mathsf{Expr}^s) \subseteq \mathsf{Box}^s$, $\mathsf{box}(H) = \overline{\mathsf{box}(E)} \in \overline{\mathsf{Box}^s} = \mathsf{Box}^e \subseteq \mathsf{Box}^d$. In the induction step, we take $H = \mathsf{op}_\Omega(\mathbf{D})$ and consider three cases. If $H \in \mathsf{Expr}^e$ then, by applying the induction hypothesis to the components of $\mathbf{D}$ and using lemmata A.3(3) and A.5(1), one can see that $\mathsf{box}(H) \in \mathsf{Box}^e \subseteq \mathsf{Box}^d$. The case $H \in \mathsf{Expr}^x$ is similar. Suppose now that $\mathbf{D} \in \mathsf{Dom}_\Omega^\mu$ is as in (iv) of the syntax. By applying the induction hypothesis to the components of $\mathbf{D}$, one can see that $\mathsf{box}(\mathbf{D}) \in \mathsf{dom}_\Omega^\mu$. Hence $\mathsf{box}(H) \in \mathsf{Box}^d$, by theorem 3.4.
(2) Follows directly from (1). ■ C.1

The inclusions in the last lemma can be reversed.

Lemma C.2 (1) $\mathsf{box}^{-1}(\mathsf{Expr}^\delta) \subseteq \mathsf{Box}^\delta$, for $\delta \in \{s,d,e,x\}$.

(2) $\mathsf{box}^{-1}(\mathsf{dom}_\Omega) \subseteq \mathsf{Dom}_\Omega$ and $\mathsf{box}^{-1}(\mathsf{dom}_\Omega^\mu) \subseteq \mathsf{Dom}_\Omega^\mu$, for all $\mu \in \mathsf{fact}_\Omega$.

Proof. (1) For Box^s and Box^d the result follows from lemma C.1(1) and $\mathsf{Box}^s \cap \mathsf{Box}^d = \emptyset$. For Box^e, it can be proved by induction on the structure of a dynamic expression H for which we assume $\mathsf{box}(H) \in \mathsf{Box}^e$. The base case is $H = \overline{E}$ or $H = \underline{E}$, where $E \in \mathsf{Expr}^s$. The latter is clearly impossible (since $\mathsf{box}(\underline{E}) \in \mathsf{Box}^x$), and in the former $H \in \mathsf{Expr}^e$. In the inductive step, suppose $H = \mathsf{op}_\Omega(\mathbf{D})$. Then, by proceeding as in the proof of lemma A.6(3), one can show that $\mathbf{D} \in \mathsf{Dom}_\Omega^{(\phi_1,\phi_2)}$, where ${}^\circ\Omega = {}^\bullet\phi_1$, $\phi_2 = \emptyset$ and $\mathsf{box}(\mathbf{D}_{\phi_1}) \subseteq \mathsf{Box}^e$. By the induction hypothesis, $\mathbf{D}_{\phi_1} \subseteq \mathsf{Expr}^e$. Hence $\mathsf{op}_\Omega(\mathbf{D}) \in \mathsf{Expr}^e$. For Box^x the proof is similar.
(2) Follows from (1). ■ C.2

C.2 Similarity relations on boxes and expressions are compatible

The first result clarifies the relationship between two equivalence notions defined for process expressions, namely the structural similarity relation, and the semantical equivalence induced by the box mapping. Below we use the notation $\mathbf{E}^{[\mu]}$, for an Ω-tuple $\mathbf{E}$ of expressions in Expr^{all} and a factorisation $\mu = (\mu_1, \mu_2)$ of a set of places in Ω, in the same way as it was used for Ω-tuples of variables in the definitions of equations (E1)-(E3). I.e., $\mathbf{E}^{[\mu]}$ is $\mathbf{E}$ with each E_v changed to $\overline{E_v}$, for $v \in \mu_1$, and each E_v changed to $\underline{E_v}$, for $v \in \mu_2$.

Lemma C.3 If D is a process expression and $D \equiv H$ then H is a process expression such that $\mathsf{box}(D) = \mathsf{box}(H)$.

Proof. If D is a static expression then $D = H$ and there is nothing to prove, so we assume that $D \in \mathsf{Expr}^d$. We first observe that if $D \equiv_{\mathsf{rec}} H$ or $H \equiv_{\mathsf{rec}} D$ then the lemma holds, which follows from lemma C.1(1) and $\mathsf{box}(K) = \mathsf{op}_\Omega(\mathsf{box}(\mathbf{L}))$, for every defining equation $K \overset{\mathrm{df}}{=} \mathsf{op}_\Omega(\mathbf{L})$. We next show that the result holds if $D \equiv_{\mathsf{OP}} H$ or $H \equiv_{\mathsf{OP}} D$.
Suppose $\mu = (\mu_1, \mu_2)$ is a factorisation of ${}^\circ\Omega$ and, by equation (E1),

$$D = \overline{\mathsf{op}_\Omega(\mathbf{E})} \equiv_{\mathsf{OP}} \mathsf{op}_\Omega(\mathbf{E}^{[\mu]}) = H.$$

Since $D \in \mathsf{Expr}^d$, we have $\mathbf{E} \subseteq \mathsf{Expr}^s$, and so $\mathbf{E}^{[\mu]} \in \mathsf{Dom}_\Omega^\mu$. Hence $D, H \in \mathsf{Expr}^e$. Moreover, $\lfloor D \rfloor = \lfloor H \rfloor$. Thus, by lemma C.1(1), $\mathsf{box}(D) = \mathsf{box}(H)$. Suppose now that, by equation (E1),

$$H = \overline{\mathsf{op}_\Omega(\mathbf{E})} \equiv_{\mathsf{OP}} \mathsf{op}_\Omega(\mathbf{E}^{[\mu]}) = D.$$

Since $D \in \mathsf{Expr}^d$, there is $\zeta = (\zeta_1, \zeta_2) \in \mathsf{fact}_\Omega$ such that $\mathbf{E}^{[\mu]} \in \mathsf{Dom}_\Omega^\zeta$. Since $E_v^{[\mu]} \notin \mathsf{Box}^s \cup \mathsf{Box}^x$, for $v \in \mu_1$, we have $\mu_1 \subseteq \zeta_1$. Thus, by lemma A.3(3), $\mu_1 = \zeta_1$ and $\mu_2 = \zeta_2 = \emptyset$. Hence $\mathbf{E} \subseteq \mathsf{Expr}^s$ and we can proceed as before.

If κ is a factorisation of Ω°, as in equation (E2), we proceed similarly as above. Consider now equation (E3) and assume that

$$D = \mathsf{op}_\Omega(\mathbf{E}^{[\eta]}) \equiv_{\mathsf{OP}} \mathsf{op}_\Omega(\mathbf{E}^{[\psi]}) = H$$

where $\eta = (\eta_1, \eta_2)$ and $\psi = (\psi_1, \psi_2)$. (Note that this case is symmetric since η and κ can be interchanged.) From lemma C.1(2) it follows that $\mathsf{box}(\mathbf{E}^{[\eta]}) \in \mathsf{dom}_\Omega$. Hence, by lemma A.4, there is $\zeta = (\zeta_1, \zeta_2) \in \mathsf{fact}_\Omega$ such that $\mathsf{box}(\mathbf{E}^{[\eta]}) \in \mathsf{dom}_\Omega^\zeta$ and $\zeta_2 = \{v \mid \mathsf{box}(E_v^{[\eta]}) \in \mathsf{Box}^x\}$. Thus, by lemma C.2, $\mathbf{E}^{[\eta]} \in \mathsf{Dom}_\Omega^\zeta$ and $\zeta_2 = \{v \mid \mathbf{E}_v^{[\eta]} \in \mathsf{Expr}^x\}$. This and $\mathbf{E}_{\widetilde{\eta}} \subseteq \mathsf{Expr}^s$ (follows from $\mathbf{E}^{[\eta]} \in \mathsf{Dom}_\Omega$) yields $\eta_1 \subseteq \zeta_1$ and $\eta_2 \subseteq \zeta_2$. Thus, by lemma A.3(6),

$$\beta = ((\zeta_1 \backslash \eta_1) \cup \psi_1, (\zeta_2 \backslash \eta_2) \cup \psi_2)$$

is a factorisation of the same set of places as ζ. Suppose $w \in \psi_1 \cap \widetilde{\zeta}$. Then, since β is a factorisation and $\widetilde{\eta} \cap \widetilde{\psi} = \emptyset$, $w \in \zeta_1$. But this contradicts the definition of factorisation (for ζ) and $^\bullet\eta_1 \cup \eta_2^\bullet = {}^\bullet\psi_1 \cup \psi_2^\bullet$. Hence $\psi_1 \cap \widetilde{\zeta} = \emptyset$. One can show in a similar way that $\psi_2 \cap \widetilde{\zeta} = \emptyset$, so we have $\widetilde{\psi} \cap \widetilde{\zeta} = \emptyset$. As a result, $\mathbf{E}_{\widetilde{\psi}} \subseteq \mathsf{Expr}^s$ and $\mathbf{E}_{\widetilde{\eta}} \subseteq \mathsf{Expr}^s$. Hence

$$\begin{array}{rcl} \mathbf{E}_{\eta_1}^{[\eta]} \cup \mathbf{E}_{\psi_1}^{[\psi]} & \subseteq & \mathsf{Expr}^e \\ \text{and} \quad \mathbf{E}_{\eta_2}^{[\eta]} \cup \mathbf{E}_{\psi_2}^{[\psi]} & \subseteq & \mathsf{Expr}^x. \end{array}$$

Moreover, since $\widetilde{\eta} \cap \widetilde{\psi} = \emptyset$, $\mathbf{E}_{\widetilde{\psi}}^{[\eta]} \subseteq \mathsf{Expr}^s$ and, by $\mathbf{E}_{\widetilde{\eta}} \subseteq \mathsf{Expr}^s$, $\mathbf{E}_{\widetilde{\eta}}^{[\psi]} \subseteq \mathsf{Expr}^s$. Finally, it is clear that $\mathbf{E}^{[\eta]}$ and $\mathbf{E}^{[\psi]}$ are the same for all elements indexed by $v \notin \widetilde{\eta} \cup \widetilde{\psi}$.Thus, by lemma C.1(1), $\mathsf{box}(\mathbf{E}^{[\eta]}) \equiv_\Omega \mathsf{box}(\mathbf{E}^{[\psi]})$. Hence, by lemmata A.11, $\mathsf{box}(\mathbf{E}^{[\psi]}) \in \mathsf{dom}_\Omega$ and

$$\mathsf{op}_\Omega(\mathsf{box}(\mathbf{E}^{[\eta]})) = \mathsf{op}_\Omega(\mathsf{box}(\mathbf{E}^{[\psi]})).$$

Thus, by lemma C.2(2) and the definition of box, $\mathbf{E}^{[\psi]} \in \mathsf{Dom}_\Omega$ and

$$\mathsf{box}(D) = \mathsf{box}(\mathsf{op}_\Omega(\mathbf{E}^{[\eta]})) = \mathsf{box}(\mathsf{op}_\Omega(\mathbf{E}^{[\psi]})) = \mathsf{box}(H).$$

To summarise, we have shown that if a pair (D, H) belongs to the relation $\equiv_{\mathsf{aux}}^0$ equal to

$$\equiv_{\mathsf{rec}} \;\cup\; \equiv_{\mathsf{rec}}^{-1} \;\cup\; \equiv_{\mathsf{OP}} \;\cup\; \equiv_{\mathsf{OP}}^{-1}$$

and $D \in \mathsf{Expr}^\delta$, for $\delta \in \{s, d\}$, then $H \in \mathsf{Expr}^d$ and $\mathsf{box}(D) = \mathsf{box}(H)$. Moreover, we can extend this to $\delta \in \{e, x\}$. Indeed, this follows immediately from $\mathsf{box}(D) = \mathsf{box}(H)$ and lemmata C.1(1) and C.2(1).

Define $\equiv_{\mathsf{aux}}^1$ as the set of all $(D, H) \in \mathsf{Expr}^{all}$ for which there are $D', H' \in \mathsf{Expr}^{all}$ such that $D' \equiv_{\mathsf{aux}}^0 H'$, D' is a subexpression of D which does not occur within the scope of application of the $\overline{(.)}$ and $\underline{(.)}$ operators, and H is D with D' replaced by H'. From what we have already shown, and by simple induction on the structure of D, it follows that the lemma holds if we replace $\equiv$ by $\equiv_{\mathsf{aux}}^1$ in its formulation. Moreover, by the definition of $\equiv$ and $\equiv_{\mathsf{aux}}^1$, we have $\equiv_{\mathsf{aux}}^1 \subseteq \equiv$. Let $\equiv_{\mathsf{aux}}$ be the transitive reflexive closure of $\equiv_{\mathsf{aux}}^1$. Clearly, the lemma also holds if we replace $\equiv$ by $\equiv_{\mathsf{aux}}$ in its formulation. Thus, to complete the proof it suffices to show that $\equiv_{\mathsf{aux}}$ is equal to $\equiv$.

We have $\equiv_{\mathsf{aux}} \subseteq \equiv$ since both $\equiv_{\mathsf{aux}}$ and $\equiv$ are equivalence relations and $\equiv_{\mathsf{aux}}^1 \subseteq \equiv$. Hence, from the minimality of $\equiv$ it follows that it is enough to show that if $\Omega \in \mathsf{OP}$ and $\mathbf{D}, \mathbf{H}$ are Ω-tuples of expressions in Expr^{all} such that $D_v \equiv_{\mathsf{aux}} H_v$ for all $v \in T_\Omega$, then $\mathsf{op}_\Omega(\mathbf{D}) \equiv_{\mathsf{aux}} \mathsf{op}_\Omega(\mathbf{H})$. Suppose $\mathbf{D} = (D_1, \ldots, D_n)$ and $\mathbf{H} = (H_1, \ldots, H_n)$. Then

$$\begin{array}{rcl} \mathsf{op}_\Omega(\mathbf{D}) & = & \mathsf{op}_\Omega(D_1, \ldots, D_n) \\ & \equiv_{\mathsf{aux}} & \mathsf{op}_\Omega(H_1, D_2, \ldots, D_n) \\ & \equiv_{\mathsf{aux}} & \mathsf{op}_\Omega(H_1, H_2, D_3 \ldots, D_n) \\ & \ldots & \\ & \equiv_{\mathsf{aux}} & \mathsf{op}_\Omega(H_1, H_2, \ldots, H_n) \\ & = & \mathsf{op}_\Omega(\mathbf{H}) \end{array}$$

which completes the proof. ■ C.3

The next result is a partial converse of the last lemma.

Lemma C.4 Let D be a dynamic expression.

(1) If $\mathsf{box}(D) \in \mathsf{Box}^e$ then $D \equiv \overline{\lfloor D \rfloor}$.

(2) If $\mathsf{box}(D) \in \mathsf{Box}^x$ then $D \equiv \underline{\lfloor D \rfloor}$.

Proof. We only prove (1), by induction on the structure of D. The base case $D = \overline{\lfloor D \rfloor}$ is obvious (note that $D = \underline{\lfloor D \rfloor}$ can be rejected by lemma C.2(1)). Suppose $D = \mathsf{op}_\Omega(\mathbf{D})$. By lemma C.2(1), $D \in \mathsf{Expr}^e$. Hence there is a factorisation $\mu = (\mu_1, \emptyset)$ of ${}^\circ\Omega$ such that $\mathbf{D}_{\mu_1} \subseteq \mathsf{Expr}^e$ and $\mathbf{D} \backslash \mathbf{D}_{\mu_1} \subseteq \mathsf{Expr}^s$. By the induction hypothesis, $D_v \equiv \overline{\lfloor D_v \rfloor}$ for $v \in \mu_1$. Hence $D \equiv \mathsf{op}_\Omega(\mathbf{F})$, where $F_v = \overline{\lfloor D_v \rfloor}$ if $v \in \mu_1$ and $F_v = D_v \in \mathsf{Expr}^s$ otherwise. Thus, by equation (E1),

$$\mathsf{op}_\Omega(\mathbf{D}) \equiv \mathsf{op}_\Omega(\mathbf{F}) \equiv_{\mathsf{OP}} \overline{\mathsf{op}_\Omega(\lfloor \mathbf{F} \rfloor)} = \overline{\lfloor D \rfloor}.$$

■ C.4

We can also clarify the relationship between $\equiv_\Omega$ and $\equiv$.

Lemma C.5 Let $\mathsf{op}_\Omega(\mathbf{D})$ be a dynamic expression and $\mathbf{\Sigma}$ be an Ω–tuple of boxes such that $\mathsf{box}(\mathbf{D}) \equiv_\Omega \mathbf{\Sigma}$. Then there is $\mathbf{F} \in \mathsf{Dom}_\Omega$ such that $\mathsf{box}(\mathbf{F}) = \mathbf{\Sigma}$ and $\mathsf{op}_\Omega(\mathbf{D}) \equiv \mathsf{op}_\Omega(\mathbf{F})$.

Proof. From $\mathbf{D} \in \mathsf{Dom}_\Omega$ and lemma C.1(2) it follows that $\mathbf{\Theta} = \mathsf{box}(\mathbf{D}) \in \mathsf{dom}_\Omega$. Hence there is a factorisation $\mu = (\mu_1, \mu_2) \in \mathsf{fact}_\Omega$ such that $\mathbf{\Theta} \in \mathsf{dom}_\Omega^\mu$. Moreover, by lemma A.4, we may assume $\mu_2 = \{v \mid \Theta_v \in \mathsf{Box}^x\}$. From $\mathbf{\Theta} \equiv_\Omega \mathbf{\Sigma}$ it follows that there are factorisations, $\eta = (\eta_1, \eta_2)$ and $\psi = (\psi_1, \psi_2)$, of the same set of places such that $\lfloor \mathbf{\Theta} \rfloor = \lfloor \mathbf{\Sigma} \rfloor$, and $\Theta_v = \Sigma_v$ for $v \notin \widetilde{\eta} \cup \widetilde{\psi}$, and

$$\begin{array}{rrcl} & \mathbf{\Theta}_{\eta_1} \cup \mathbf{\Sigma}_{\psi_1} & \subseteq & \mathsf{Box}^e \\ & \mathbf{\Theta}_{\eta_2} \cup \mathbf{\Sigma}_{\psi_2} & \subseteq & \mathsf{Box}^x \qquad (*) \\ \text{and} & \mathbf{\Theta}_{\widetilde{\psi}} \cup \mathbf{\Sigma}_{\widetilde{\eta}} & \subseteq & \mathsf{Box}^s. \end{array}$$

We first observe that, by the choice of μ, $\eta_1 \subseteq \mu_1$ and $\eta_2 \subseteq \mu_2$, so η and ψ are factorisations of a set of places included in a marking reachable from ${}^\circ\Omega$. We also observe that, by lemma A.3(1) and the fact that η and ψ are factorisations of the same set of places, $\eta_1 \cap \psi_2 = \emptyset = \eta_2 \cap \psi_1$. Hence the above also holds for η redefined as $(\eta_1 \backslash \psi_1, \eta_2 \backslash \psi_2)$ and ψ as $(\psi_1 \backslash \eta_1, \psi_2 \backslash \eta_2)$. As a result, we may assume that η and ψ are disjoint factorisations, which makes them satisfy the conditions in the formulation of equation (E3).

Define an Ω-tuple of expressions $\mathbf{E}$ by $E_v = D_v$, for $v \notin \widetilde{\eta}$, and $E_v = \lfloor D_v \rfloor$, for $v \in \widetilde{\eta}$. Then $D_v \equiv \mathbf{E}_v^{[\eta]}$, for all $v \in T_\Omega$, which follows from lemma C.4 and (*). Define $\mathbf{F} = \mathbf{E}^{[\psi]}$. Then, by equation (E3),

$$\mathsf{op}_\Omega(\mathbf{D}) \equiv \mathsf{op}_\Omega(\mathbf{E}^{[\eta]}) \equiv_{\mathsf{OP}} \mathsf{op}_\Omega(\mathbf{E}^{[\psi]}) = \mathsf{op}_\Omega(\mathbf{F}).$$

Hence, by lemma C.3, $\mathbf{F} \in \mathsf{Dom}_\Omega$. Moreover, $\mathbf{\Sigma} = \mathsf{box}(\mathbf{F})$ follows from (*).

■ C.5

C.3 Proofs of soundness and completeness of operational semantics

Proof of Theorem 5.1. If $U = \emptyset$ then $D \equiv H$. Hence the thesis holds by lemma C.3. If $U \neq \emptyset$ then we proceed by induction on the structure of D. Suppose $D = \overline{E}$, for some $E \in \mathsf{Expr}^s$. Then $E = a$, for some $a \in \mathsf{Lab}$, and the thesis obviously holds. The case $D = \underline{E}$, for some $E \in \mathsf{Expr}^s$, can be excluded since no non-empty move is then possible. Suppose

$$D = \mathsf{op}_\Omega(\mathbf{D}) \xrightarrow{U} \mathsf{op}_\Omega(\mathbf{H}) = H.$$

Then, by the induction hypothesis and the $U = \emptyset$ case,

$$(\Omega : \mathsf{box}(\mathbf{D})) \xrightarrow{U} (\Omega : \mathsf{box}(\mathbf{H})).$$

Hence, by theorem 3.2, $\mathsf{box}(\mathbf{H}) \in \mathsf{dom}_\Omega$ and

$$\mathsf{op}_\Omega(\mathsf{box}(\mathbf{D}))\ [U\rangle\ \mathsf{op}_\Omega(\mathsf{box}(\mathbf{H})).$$

Moreover, by lemma C.2(2), $\mathbf{H} \in \mathsf{Dom}_\Omega$, so H is a dynamic expression. Thus

$$\mathsf{box}(D) = \mathsf{op}_\Omega(\mathsf{box}(\mathbf{D}))\ [U\rangle\ \mathsf{op}_\Omega(\mathsf{box}(\mathbf{H})) = \mathsf{box}(H).$$

■ 5.1

Proof of Theorem 5.2. By the inaction rule, it suffices to show $D \equiv H \xrightarrow{U} J$. If $U = \emptyset$ then the thesis holds with $D = H = J$. If $U \neq \emptyset$ then we proceed by induction on the maximal height h of the transition trees[12] in U (this is possible since U is a finite set of finite trees). We first observe that one can ignore the case when $D = \underline{E}$, for some $E \in \mathsf{Expr}^s$, since then $\mathsf{box}(D) \in \mathsf{Box}^x$ and no non-empty step is enabled. Similarly, one can ignore the case when $D = \overline{E}$, for some $E \in \mathsf{Expr}^s \setminus \mathsf{Lab}$. Indeed, we can always use the equation (E1) for the initial marking, the relation $\equiv_{\mathsf{rec}}$, the fact that $^\circ\Omega$ hasat least onefactorisation, and lemma C.3to transform D into another expression D' of the form $\mathsf{op}_\Omega(\mathbf{D})$ and satisfying $D \equiv D'$ and $\mathsf{box}(D) = \mathsf{box}(D')$.

Suppose $h = 1$. Then, by the above observation, we may assume $D = \overline{a}$, for $a \in \mathsf{Lab}$, and the lemma holds with $D = H$ and $J = \underline{a}$. Suppose $h > 1$. Then, again by the observation we made, we may assume $D = \mathsf{op}_\Omega(\mathbf{D})$. Hence $\mathsf{op}_\Omega(\mathsf{box}(\mathbf{D}))\ [U\rangle\ \Sigma$. From theorem 3.3 it follows that there are $\mathbf{\Theta}, \mathbf{\Gamma} \in \mathsf{dom}_\Omega$ such that $\mathsf{box}(\mathbf{D}) \equiv_\Omega \mathbf{\Theta}$ and $\Sigma = \mathsf{op}_\Omega(\mathbf{\Gamma})$ and

$$(\Omega : \mathbf{\Theta}) \xrightarrow{U} (\Omega : \mathbf{\Gamma}).$$

From lemma C.5 it follows that there is $\mathbf{F} \in \mathsf{Dom}_\Omega$ such that $\mathsf{box}(\mathbf{F}) = \mathbf{\Theta}$ and $\mathsf{op}_\Omega(\mathbf{D}) \equiv \mathsf{op}_\Omega(\mathbf{F})$. Suppose $U = \bigcup_{v \in T} U_v$, where U_v are transitions in $\mathsf{T}^v_{\mathsf{new}}$ (see the definition of net refinement with $\mathsf{box}(\mathbf{D})$ playing the role of $\mathbf{\Sigma}$). Let v be a transition in Ω and $U_v = \{u_1, \ldots, u_k\}$ $(k \geq 0)$. We have

$$\Theta_v\ [\mathsf{trees}(u_1) \uplus \cdots \uplus \mathsf{trees}(u_k)\rangle\ \Gamma_v$$

so, by the induction hypothesis (recall that all transition trees have finite height) and the $U = \emptyset$ case, there are process expressions H_v and G_v such that $\mathsf{box}(G_v) = \Gamma_v$ and

$$F_v \equiv H_v \xrightarrow{\mathsf{trees}(u_1) \uplus \cdots \uplus \mathsf{trees}(u_k)} G_v.$$

Hence we have

$$\mathsf{op}_\Omega(\mathbf{D}) \equiv \mathsf{op}_\Omega(\mathbf{F}) \equiv \mathsf{op}_\Omega(\mathbf{H}) \xrightarrow{U} \mathsf{op}_\Omega(\mathbf{G})$$

and (by lemma C.3 and theorem 5.1) $\mathsf{op}_\Omega(H)$ and $\mathsf{op}_\Omega(G)$ are process expressions and $\mathsf{box}(\mathsf{op}_\Omega(\mathbf{G})) = \mathsf{op}_\Omega(\mathbf{\Gamma}) = \Sigma$, which completes the proof. ■ 5.2

[12] The height of a tree is defined as the number of nodes on the longest directed path from the root to a leaf node.

C.4 Independence relation is compositional

Proof of proposition 5.5. We prove a stronger result. Namely, we will also show that there are two sets of transition trees in $\mathsf{Proc_{tree}}$, in and out, such that for every process expression D and every transition t in $\mathsf{box}(D)$ the following hold:

$$\begin{array}{rcl} {}^\bullet t \cap {}^\circ\mathsf{box}(D) \neq \emptyset & \Leftrightarrow & t \in \mathsf{in} \\ \text{and} \quad t^\bullet \cap \mathsf{box}(D)^\circ \neq \emptyset & \Leftrightarrow & t \in \mathsf{out}. \end{array}$$

Define $\mathsf{ind} \subseteq \mathsf{Proc_{tree}} \times \mathsf{Proc_{tree}}$ and $\mathsf{in}, \mathsf{out} \subseteq \mathsf{Proc_{tree}}$ as the smallest sets satisfying the following two conditions:

(1) For every $a \in \mathsf{Lab}$, $t \in \mathsf{in}$ and $t \in \mathsf{out}$, where t is the only transition in $\mathsf{box}(a)$.

(2) For every operator box $\Omega \in \mathsf{OP}$, the following hold (below $(v, a) \lhd Q$ etc are transition trees in $\mathsf{Proc_{tree}}$, and v, w are transitions in Ω):

  (a) If ${}^\bullet v \cap {}^\circ\Omega \neq \emptyset$ and $Q \cap \mathsf{in} \neq \emptyset$ then $(v, a) \lhd Q \in \mathsf{in}$.

  (b) If $v^\bullet \cap \Omega^\circ \neq \emptyset$ and $Q \cap \mathsf{out} \neq \emptyset$ then $(v, a) \lhd Q \in \mathsf{out}$.

  (c) If $v \neq w$ and none of the following holds:

$$\begin{array}{lllll} {}^\bullet v \cap {}^\bullet w \neq \emptyset & \wedge & Q \cap \mathsf{in} \neq \emptyset & \wedge & R \cap \mathsf{in} \neq \emptyset \\ {}^\bullet v \cap w^\bullet \neq \emptyset & \wedge & Q \cap \mathsf{in} \neq \emptyset & \wedge & R \cap \mathsf{out} \neq \emptyset \\ v^\bullet \cap {}^\bullet w \neq \emptyset & \wedge & Q \cap \mathsf{out} \neq \emptyset & \wedge & R \cap \mathsf{in} \neq \emptyset \\ v^\bullet \cap w^\bullet \neq \emptyset & \wedge & Q \cap \mathsf{out} \neq \emptyset & \wedge & R \cap \mathsf{out} \neq \emptyset \end{array}$$

  then $((v, a) \lhd Q, (w, b) \lhd R) \in \mathsf{ind}$.

  (d) If $Q \times R \subseteq \mathsf{ind}$ then $((v, a) \lhd Q, (v, b) \lhd R) \in \mathsf{ind}$.

Using lemma A.2 and induction on the height of transition trees, one can show that ind, in and out satisfy the required properties. Note that the assumption that the operator boxes in OP have disjoint sets of transitions is needed to guarantee the correctness of the definition of ind, in and out, by providing a unique association of $(v, a) \lhd Q$ with Ω. ■ 5.5