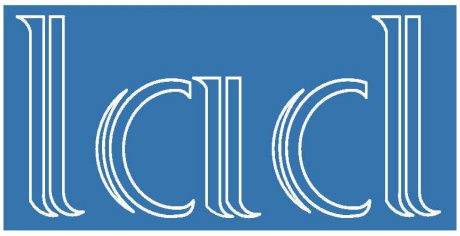
Asynchronous Box Calculus with multi-way communication

Cécile Bui Thanh, Hanna Klaudel, Franck Pommereau



Asynchronous Box Calculus with Multi-way Communication

Bui Thanh, C. and Klaudel, H. and Pommereau, F.

Technical Report TR-2002-16

Laboratory of Algorithms, Complexity and Logic
University of Paris XII, Val-de-Marne
61, avenue du Général de Gaulle
F-94010 CRÉTEIL Cedex – FRANCE
Tel: +33 (0)1 45 17 16 47
Fax: +33 (0)1 45 17 66 01

Asynchronous Box Calculus with Multi-way Communication

Cécile Bui Thanh, Hanna Klaudel, and Franck Pommereau

LACL, Université Paris 12
61, avenue du général de Gaulle
94010 Créteil, France
{bui,klaudel,pommereau}@univ-paris12.fr

Abstract. The starting point of this paper is the *Asynchronous Box Calculus* (or ABC), a formalism suitable for modelling compositionally distributed systems using both synchronous and asynchronous communication (through *actions* and *links* respectively). ABC is composed of two semantically consistent models: an algebra of Petri nets and an associated algebra of process expressions whose constants and operators directly corresponds to the Petri net ones. In this paper, we extend the Petri net algebra of ABC by allowing the use of multisets of actions as well as that of multisets of links. The ABC process expressions and their associated structured operational semantics (SOS) are extended in the same way. The obtained framework, called MBC (for *Multi-way communication Box Calculus*), allows to express processes which could not be modelled with ABC; for instance, processes with multi-way synchronisation. As it was the case for ABC, the resulting algebra of expressions is consistent with the net algebra in the sense that an expression and the corresponding net generate isomorphic transition systems.
Keywords. Petri nets, process algebra, synchronous and asynchronous communication, structured operational semantics.

1 Introduction

This paper is set in the framework of the *Petri Box Calculus* (PBC [1]), which has been designed with the aim of allowing a compositional Petri net semantics of nondeterministic and concurrent programming languages [4]. It was later extended into a more generic *Petri Net Algebra* (PNA [2, 3]), and afterwards to an *Asynchronous Box Algebra* (ABC [5]) which introduced asynchronous communication links [6]. The ABC model is composed of an algebra of process expressions (*box expressions*) with a fully compositional translation into labelled Petri nets (*boxes*). Synchronous communications are modelled by *actions* and asynchronous communications by *links*. A basic ABC event can perform at most one synchronous communication and one asynchronous link. This may be seen as a limitation because it makes impossible, for instance, the multi-way synchronisations to be built compositionally. In this paper, we remove this restriction and introduce multisets of actions (called *multiactions*) for synchronous communications and multisets of links (called *multilinks*) for asynchronous ones. The resulting model is called *Multi-way communication Box Calculus* (*MBC* for short).

We will now introduce informally the aspects of this model which are needed for our extension.

1.1 An algebra of nets and process expressions

The variant of the ABC model relevant for this paper considers the following operators : *sequence* $E_1; E_2$ (the execution of the process E_1 is followed by that of E_2); *choice* $E_1 \,\square\, E_2$ (either E_1 or E_2 can be executed); *parallel composition* $E_1 \| E_2$ (E_1 and E_2 can be executed concurrently); *iteration* $E_1 \circledast E_2$ (E_1 can be executed an arbitrary number of times and is followed by E_2); and *buffer restriction* $E \,\mathsf{tie}\, b$ (the buffer b and the related asynchronous links become private to the expression E). The ABC operator of *scoping* $E \,\mathsf{sc}\, a$ enforces all *multi-way* synchronisations (instead of binary

ones) between the sets of multiactions involving a or $\widehat{a}$, but forbidding any independent execution of such actions.

A basic process expression is composed of a multiaction and a multilink where one or both of them may be empty. An example of a basic expression is be, for instance, $\{a_r, \widehat{a_c}\}\{b_a^-, b_d^+\}$, where a_r and $\widehat{a_c}$ are synchronous actions used for scoping while b_a^- and b_d^+ are asynchronous links expressing the receiving of a resource from the buffer b_a and a sending to the buffer b_d. Such an expression may be executed if it is in a context with sufficiently many resources in buffers in order to satisfy all the receiving links (in our case, we should have at least one ressource in the buffer b_a). Its execution removes then a resource from the buffer b_a, produces one in the buffer b_d, and emits the multiaction $\{a_r, \widehat{a_c}\}$. A basic net corresponding to such an expression has one transition labelled $\{a_r, \widehat{a_c}\}$ connected to one input and one output place and to two buffer places labelled b_a and b_d. The receiving link b_a^- is represented by an arc going from the buffer place b_a to the transition and the sending link b_d^+ is represented by an arc from the transition to the buffer place labelled b_d.

We illustrate these constructs using an example based on three process expressions:

$$\begin{aligned}
\textsc{Employee} &\stackrel{\text{df}}{=} (\{\widehat{a_r}, \widehat{a_r}\}\{\} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\})\,\mathsf{tie}\,b_d\ ,\\
\textsc{Boss} &\stackrel{\text{df}}{=} ((\{a_r, \widehat{a_y}\}\{\}\ ;\ \{\}\{b_a^+\})\ \square\ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}\ ,\\
\textsc{Friend} &\stackrel{\text{df}}{=} \{a_r\}\{\} \circledast \{a_c\}\{\}
\end{aligned}$$

modelling an employee, his boss and his friend. The employee repetitively requests his boss for a pay rise, which is modelled by the synchronous action $\widehat{a_r}$ in the first part of the iteration. At the same time, the employee receives encouragements from his friend, which is also modelled by the action $\widehat{a_r}$. Notice that this first step needs a three-way communication, which was impossible to model compositionally in ABC. The boss may answer the request by yes or no which is modelled by the actions $\widehat{a_y}$ and $\widehat{a_n}$, respectively. If he accepts, then an amend to the contract is sent to the employee, which is modelled by the sending link b_a^+. If not, the employee continues asking until the boss agrees or terminates, in which case the system is deadlocked. The employee receiving an amend to the contract (modelled by the receiving link b_a^-) keeps it with his personal documents (modelled by the sending link b_d^+) and is congratulated by his friend (through the synchronous action $\widehat{a_c}$). Since the buffer b_d is used to keep private documents, it is encapsulated into Employee thanks to the buffer restriction $\mathsf{tie}\,b_d$.

Assuming that the binary operators associate to the left, the system where these processes operate in parallel is:

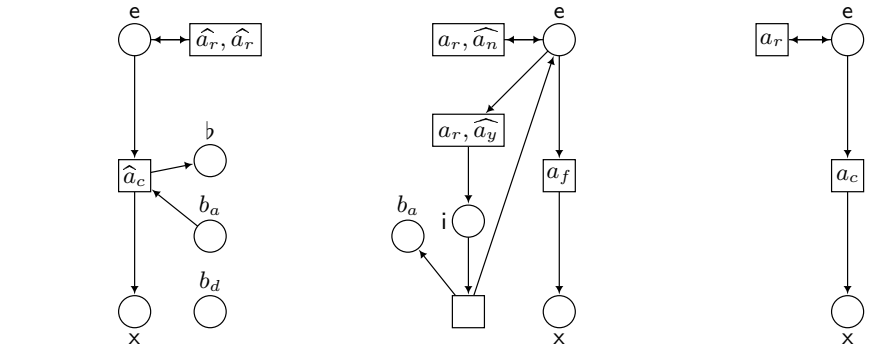
$$\textsc{Par} \stackrel{\text{df}}{=} \textsc{Friend} \parallel \textsc{Employee} \parallel \textsc{Boss}.$$

However, this system does not allow yet for synchronous communication between the processes (through the synchronous communication actions a_r and $\widehat{a_r}$, a_c and $\widehat{a_c}$). This may be achieved by applying the scoping w.r.t. a_r and a_c, leading to the expression:

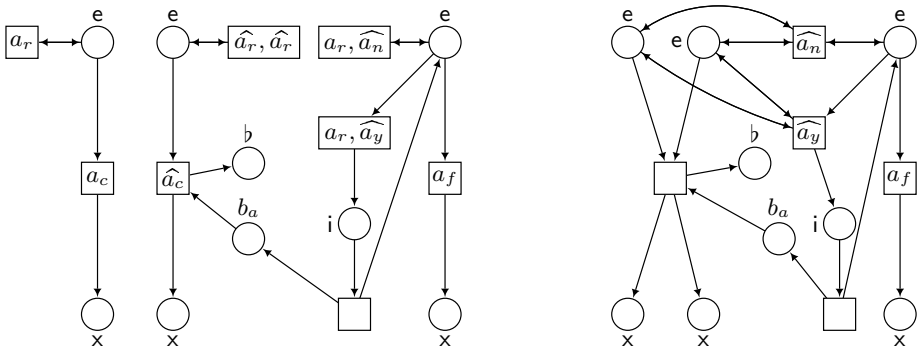
$$\textsc{PayRise} \stackrel{\text{df}}{=} (\textsc{Friend} \parallel \textsc{Employee} \parallel \textsc{Boss})\ \mathsf{sc}\ a_r\ \mathsf{sc}\ a_c.$$

The nets (called the *boxes*) corresponding to Employee, Boss and Friend are represented on the top of figure 1. The boxes of Par and PayRise are presented in the middle of the same figure. They are all the translation of their process expressions (themselves called the *box expressions*).

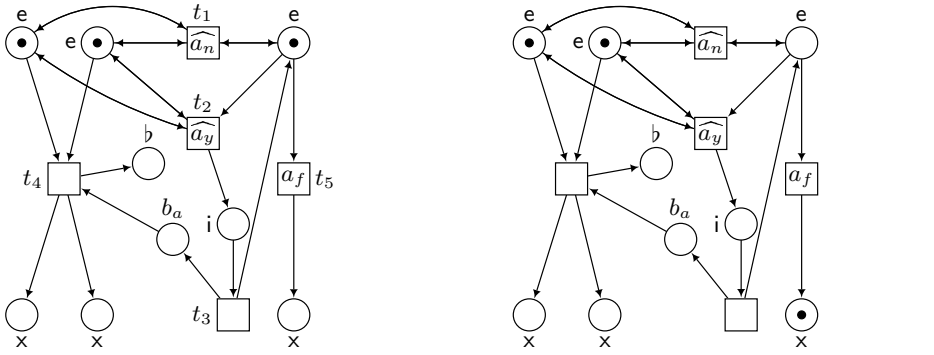
Notice that, in a box, the places are labelled by their status (e for entry, x for exit, i for internal places, $\flat$ for restricted buffer places and the name of the buffer for each unrestricted buffer place) while the transitions are labelled by multiactions. The multilinks present in the expression are represented in the box as arcs between a transition and the corresponding buffer places. Each binary operator merges the unrestricted buffer places having the same label making asynchronous communication effective. For instance, the two buffer places labelled by b_a coming from Employee and Boss are merged when the parallel composition is applied, producing a box with only one b_a-labelled place. The application of the buffer restriction w.r.t. b_d changes the status of buffer place



EMPLOYEE BOSS FRIEND



The box of PAR The box of PAYRISE



Σ_0 — the box of $\overline{\text{PAYRISE}}$ Σ_2 — the box of $\overline{\text{PAYRISE}}$ after the firing of t_5

Fig. 1. Boxes used in the example of the pay rise request. By convention, annotations inside the transitions represent the corresponding multiactions, and the one outside next to the transitions (like in the box of $\overline{\text{PAYRISE}}$) represent their names (identities).

b_d to $\flat$ denoting that it is now a private place which cannot be merged anymore. The scoping w.r.t. a_r and a_c applied to the box of PAR produces the new transitions labelled $\widehat{a_y}$ and $\widehat{a_n}$ (visible in the box of PAYRISE) and removes from the net all the transitions involving a_r, $\widehat{a_r}$, a_c and $\widehat{a_c}$ in their labels.

In this paper, synchronous actions are always denoted by the letter a (possibly with some subscripts), asynchronous links are denoted by the letter b (also with subscripts). Moreover, empty multiactions are omitted in the figures, as well as disconnected unmarked buffer places.

1.2 Structured operational semantics

The operational semantics of box expressions is given through SOS rules in Plotkin's style [7]. However, instead of expressing the evolutions through rules modifying the structure of the expressions, like $a.E \xrightarrow{a} E$ in CCS, the idea here is to represent the current state of the evolution using overbars and underbars, corresponding respectively to the initial and final states of (sub)expressions. This is illustrated by the bottom left net of the figure 1, which represents the initial state of the system specified by PAYRISE, *i.e.*, that corresponding to the box expression $\overline{\text{PAYRISE}}$.

There are two kinds of SOS rules: equivalence rules specifying when two distinct expressions denote the very same state, *e.g.*, one can derive that

$$\overline{\text{FRIEND}\|\text{EMPLOYEE}\|\text{BOSS}} \equiv \overline{\text{FRIEND}\|\text{EMPLOYEE}}\|\overline{\text{BOSS}} \equiv \overline{\text{FRIEND}}\|\overline{\text{EMPLOYEE}}\|\overline{\text{BOSS}}\ ,$$

and evolution rules specifying when we may have a state change, *e.g.*, one can derive that

$$\overline{\{a_r, \widehat{a_y}\}\{\}}\ ;\ \{\}\{b_a^+\} \xrightarrow{\{a_r, \widehat{a_y}\}} \underline{\{a_r, \widehat{a_y}\}\{\}}\ ;\ \{\}\{b_a^+\}$$

1.3 Execution examples

In order to provide more intuition about the way the MBC algebra is used, we describe in this section two system evolutions (or execution scenarios) of the system PAYRISE presented in the previous section. We use labelled step sequences as a formal device for capturing concurrent behaviours,[1] and we assume that the system starts from its implicit initial state. Thus, for example, we consider $\overline{\text{PAYRISE}}$ rather than PAYRISE.

Scenario I. Consider the box Σ_0 of $\overline{\text{PAYRISE}}$ shown in figure 1, and the following evolution:

- the employee, encouraged by his friend, requests for his pay rise but the boss refuses (this is modelled by the transition t_1);
- the employee, still encouraged, asks again and now the boss accepts (transition t_2);
- the boss sends the amend to the contract (transition t_3, which puts a token in the buffer place labelled by b_a);
- the employee receives the amend, puts it into his private documents and is also congratulated by his friend (transition t_4); simultaneously, the boss terminates (transition t_5).

Such a scenario corresponds to the step sequence $\{t_1\}\,\{t_2\}\,\{t_3\}\,\{t_4, t_5\}$ leading from Σ_0 to a net Σ_1, where Σ_1 is Σ_0 with one token in the buffer place $\flat$, one token in each of the x-labelled places, and no token elsewhere. This is denoted by:

$$\Sigma_0\,[\{t_1\}\,\{t_2\}\,\{t_3\}\,\{t_4, t_5\}\rangle\,\Sigma_1\ ,$$

In terms of labelled step sequences, the scenario is represented by

$$\Sigma_0\,[\{\{\widehat{a_n}\}\}\,\{\{\widehat{a_y}\}\}\,\{\{\}\}\,\{\{a_f\}, \{\}\}\rangle\,\Sigma_1\ .$$

Since each x-labelled place holds exactly one token, we will say that Σ_1 is in a final marking (or final state).

[1] A labelled step is a multiset of multiactions, while a step is here a set of transitions.

Scenario II. Consider now the following execution of Σ_0:

- the employee asks for his pay rise but the boss refuses (transition t_1);
- the boss terminates (transition t_5);

This scenario corresponds to $\Sigma_0 \, [\{\{\widehat{a_n}\}\} \, \{\{a_f\}\}\rangle \, \Sigma_2$, where Σ_2 is the box represented in figure 1. Notice that Σ_2 is *not* in a final marking. Moreover, the transition t_4, which comes from the synchronisation of the employee and his friend on action a_c is now dead (it will never have a token in the b_a-labelled place in order to fire).

Notice that the same labelled step sequences may be obtained at the box expression level by the application of the operational rules (SOS) to $\overline{\text{PayRise}}$.

1.4 About this paper

Multi-way communication is introduced through multiactions (multisets of actions replacing the single actions of ABC) and multilinks (multisets of links while ABC has single links). Since one atomic event in MBC can perform arbitrarily many communications, the three kinds of constant expressions and basic boxes in ABC are replaced by infinite families of basic expressions and basic boxes representing all the possibilities. The structured operational semantics is modified in order to take all these changes into account; in particular, the rules for basic moves are replaced with a single general rule. All the proofs were carefully revised and it turned out that in all cases, they could be adapted to the new framework in a quite systematic and straightforward way. This is the reason why they are not included in the present paper.

To start with, section 2 describes the class of labelled nets on which MBC is based. In section 3, we introduce the net algebra part of MBC; in particular, we define a number of operators, either directly, or by using an auxiliary net substitution meta-operator. Section 4 investigates the relationship between the structure and the behaviour of composite nets. In section 5, we introduce an algebra of process expressions which forms the second part of the MBC. In particular, we define a translation from expressions to nets, and an operational semantics of process expressions both in terms of steps of transitions of the corresponding boxes and in terms of their labels. We also present there our main result that a box expression and the corresponding box generate isomorphic transition systems.

2 Preliminaries

In this section, we present a number of definitions used throughout the paper.

2.1 Multisets

A *multiset* over a set X is a function $\mu : X \to \mathbb{N}$. We denote by $\mathsf{mult}(X)$ the set of all finite multisets μ over X, *i.e.*, those satisfying $\sum_{x \in X} \mu(x) < \infty$. We will write $\mu \leq \mu'$ if the domain X of μ is included in that of the multiset μ', and $\mu(x) \leq \mu'(x)$, for all $x \in X$. An element $x \in X$ belongs to μ, denoted $x \in \mu$, if $\mu(x) > 0$. The sum and difference of multisets, and the multiplication by a non-negative integer are respectively denoted by $+$, $-$ and $\cdot$ (the difference will only be applied when the second argument is smaller or equal to the first one). A subset of X may be treated as a multiset over X, by identifying it with its characteristic function, and a singleton set can be identified with its sole element.

A finite multiset μ over X may be written as $\sum_{x \in X} \mu(x) \cdot x$ or $\sum_{x \in X} \mu(x) \cdot \{x\}$, as well as in extended set notation, *e.g.*, $\{a_1, a_1, a_2\}$ denotes a multiset μ such that $\mu(a_1) = 2$, $\mu(a_2) = 1$ and $\mu(x) = 0$ for all $x \in X \setminus \{a_1, a_2\}$.

In this paper, in particular for steps or in the annotations of transitions, we may denote by $\{\}$ instead of $\varnothing$ empty multisets of actions or links.

2.2 Actions, buffers and synchronisation

We assume that there is a set $\mathbb{A}$ of *actions* representing synchronous interface activities used, in particular, to model handshake communication. We will also assume that, for every $a \in \mathbb{A}$, $\widehat{a}$ is an action in $\mathbb{A}$ such that $\widehat{\widehat{a}} = a$. In addition to the set of atomic actions, there is a finite set $\mathbb{B}$ of *buffer symbols* (or buffers) for asynchronous interprocess communications.[2] In the following, we will use the notation $\mathsf{m}\mathbb{B} \stackrel{\text{df}}{=} \mathsf{mult}(\mathbb{B})$. Communications through a buffer $b \in \mathbb{B}$ are represented with *asynchronous links* of the form b^+ for sending or b^- for receiving. Thus, we denote by $\mathbb{L} \stackrel{\text{df}}{=} \{b^+, b^- \mid b \in \mathbb{B}\}$ the set of all possible links.

Since an atomic action can perform simultaneously several synchronous and asynchronous communications, it will have the form $\alpha\beta$ where $\alpha \in \mathsf{m}\mathbb{A} \stackrel{\text{df}}{=} \mathsf{mult}(\mathbb{A})$ is a *multiaction*, representing its synchronous communications (actions), and $\beta \in \mathsf{m}\mathbb{L} \stackrel{\text{df}}{=} \mathsf{mult}(\mathbb{L})$ is a *multilink*, representing its asynchronous communications (links).

We need a device to express formally that, *e.g.*, concurrent actions should synchronise. To this end, we will use partial functions $\varphi : \mathsf{mult}(\mathsf{m}\mathbb{A}) \to \mathsf{m}\mathbb{A}$, called interface functions, and interpreted as a subset of $\mathsf{mult}(\mathsf{m}\mathbb{A}) \times \mathsf{m}\mathbb{A}$. For $(\Gamma, \alpha) \in \varphi$, we define $\varphi(\Gamma) = \alpha$.

An example of such a function is the identity $\varphi_{id} \stackrel{\text{df}}{=} \{(\{\alpha\}, \alpha) \mid \alpha \in \mathsf{m}\mathbb{A}\}$. This function is used when no synchronous interface change has to be performed.

The interface function $\varphi_{\mathsf{sy}\,a}$ is used in order to specify when concurrent events labelled by multiactions $\alpha_1, \cdots, \alpha_k$ can synchronise w.r.t $a \in \mathbb{A}$. Formally, $\varphi_{\mathsf{sy}\,a}$ is defined as the smallest set containing φ_{id} and such that if $(\Gamma_1, \alpha_1 + \{a\})$ and $(\Gamma_2, \alpha_2 + \{\widehat{a}\})$ belong to $\varphi_{\mathsf{sy}\,a}$, then $(\Gamma_1 + \Gamma_2, \alpha_1 + \alpha_2)$ belongs to $\varphi_{\mathsf{sy}\,a}$. For instance, we have $\varphi_{\mathsf{sy}\,a}(\{a_1, a_1, a_2\}, \{\widehat{a_1}, a_3\}, \{\widehat{a_1}\}) = \{a_2, a_3\}$, which means that the multiactions $\{a_1, a_1, a_2\}$, $\{\widehat{a_1}, a_3\}$ and $\{\widehat{a_1}\}$ may lead to a multi-way synchronisation represented by the multiaction $\{a_2, a_3\}$, while the fact that $\varphi_{\mathsf{sy}\,a}(\{a_1, \widehat{a_1}\}, \{a_2\})$ is not defined means that multiactions $\{a_1, \widehat{a_1}\}$ and $\{a_2\}$ cannot synchronise together.

Another useful interface function is $\varphi_{\mathsf{sc}\,a}$, for $a \in \mathbb{A}$, which allows to setup all synchronous communications w.r.t. a but forbids the execution of events labelled by multiactions still involving a. Formally: $\varphi_{\mathsf{sc}\,a} \stackrel{\text{df}}{=} \{(\Gamma, \alpha) \in \varphi_{\mathsf{sy}\,a} \mid a, \widehat{a} \notin \alpha\}$. Such a function will be used to enforce CCS-like synchronisations, but with no limitation on the number of simultaneously performed synchronisations.

2.3 Labelled nets

A *(marked) labelled net* is, in the present framework, a tuple $\Sigma \stackrel{\text{df}}{=} (S, T, W, \lambda, M)$ such that:

- S and T are disjoint sets of *places* and *transitions*, respectively.
- W is a *weight function* from the set $(S \times T) \cup (T \times S)$ to $\mathbb{N}$.
- λ is a *labelling* for places and transitions such that $\lambda(s)$ is a symbol in $\{\mathsf{e}, \mathsf{i}, \mathsf{x}, \flat\} \uplus \mathbb{B}$, for every place $s \in S$; and $\lambda(t)$ is an interface function φ or a multiaction in $\mathsf{m}\mathbb{A}$, for every transition $t \in T$.
- M is a *marking*, *i.e.*, a multiset over S (in other words, a mapping from the set of places S to $\mathbb{N}$).

Moreover, Σ is *finite* if both S and T are finite sets, and it is *simple* if W always returns 0 or 1.

If the labelling of a place s is e, i or x, then s is an *entry*, *internal* or *exit* place, respectively. Collectively, these places are called *control (flow)* places. If the labelling is $\flat$ then s is a *closed buffer* place, and if it is a buffer symbol $b \in \mathbb{B}$, then s is an *open buffer* place. The set of all entry (resp. exit) places will be denoted by ${}^\circ\Sigma$ (resp. Σ°). We shall also use M^{ctr} and M^{opb} to denote M restricted to the control places and to the open buffer places, respectively; finally, Σ^{ctr} is Σ with all its buffer places and adjacent arcs removed (intuitively, this also amounts to putting an infinite marking on each closed or open buffer place of Σ).

[2] The finiteness of $\mathbb{B}$ is not essential, but it will allow us to consider only finite nets. We also assume that $\mathsf{e}, \mathsf{x}, \mathsf{i}, \flat \notin \mathbb{B}$.

We adopt the standard rules about representing nets as directed graphs; moreover, double-headed arrows will represent self-loops. To avoid ambiguity, we will sometimes decorate the various components of Σ with the index Σ; *e.g.*, T_Σ will denote the set of transitions of Σ. In order to simplify diagrams, we will omit disconnected unmarked buffer places.

For every place (resp. transition) x, we use ${}^\bullet x$ to denote its pre-set, *i.e.*, the set of all transitions (resp. places) y such that there is an arc from y to x, *i.e.*, $W(y,x) > 0$. 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.*, ${}^\bullet R \stackrel{\text{df}}{=} \bigcup_{r\in R} {}^\bullet r$.

2.4 Step sequences

A finite step sequence semantics of a labelled net Σ captures the potential concurrency in the behaviour of the system modelled by Σ. A finite multiset of transitions U, called a *step*, is *enabled* by Σ if for every place $s \in S$,

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

We denote by $\mathsf{enabled}(\Sigma)$ the set of all steps enabled by Σ; notice that we always have $\varnothing \in \mathsf{enabled}(\Sigma)$. A step $U \in \mathsf{enabled}(\Sigma)$ can be *executed*, leading to a marking M' given, for every place $s \in S$, by

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

We will denote this by $\Sigma[U\rangle\Sigma'$, where Σ' is Σ with the marking changed to M'. Transition labelling may be extended to steps, through the formula

$$\lambda(U) \stackrel{\text{df}}{=} \sum_{t\in U} U(t)\cdot\lambda(t) \in \mathsf{mult}(\mathsf{m}\mathbb{A})\ .$$

In particular, we will denote $\Sigma\,[\Gamma\rangle_\lambda\,\Sigma'$ whenever there is a multiset of transitions U such that $\Sigma\,[U\rangle\,\Sigma'$ and $\Gamma = \lambda(U)$. This allows one to translate various behavioural notions defined in terms of multisets of transitions into notions based on multisets of transition labels (or *labelled steps*). Although we will use the same term 'step' to refer both to a finite multiset of transitions and to a finite multiset of labels, it will always be clear from the context which one is meant. It may happen that two different transition steps correspond to the same labelled step, when different transitions have the same label.

A *step sequence* of Σ is a (possibly empty) sequence of steps, $\omega = U_1 \ldots U_k$, such that there are nets $\Sigma_1, \ldots, \Sigma_k$ satisfying $\Sigma\,[U_1\rangle\,\Sigma_1\,[U_2\rangle\,\Sigma_2 \cdots [U_k\rangle\,\Sigma_k$. We will denote this by $\Sigma\,[\omega\rangle\,\Sigma_k$ or $\Sigma_k \in [\Sigma\rangle$, and call Σ_k *derivable* from Σ and its marking M_{Σ_k} *reachable* from M_Σ (with the convention that $\Sigma\,[\omega\rangle\,\Sigma$ if $k = 0$, *i.e.*, if ω is the empty step sequence). The definition of a *labelled step sequence* of Σ is similar.

2.5 Clean, ac-free and quasi-safe markings

A marking M of Σ is:

- *clean* if $M^{ctr} \geq {}^\circ\Sigma \Rightarrow M^{ctr} = {}^\circ\Sigma$ and $M^{ctr} \geq \Sigma^\circ \Rightarrow M^{ctr} = \Sigma^\circ$.
  If $M^{ctr} = {}^\circ\Sigma$, we will say that Σ is in an *initial* state (or marking), and if $M^{ctr} = \Sigma^\circ$, we will say that Σ is in a *final* state (or marking).
- *ac-free* if, for every transition t, there is a control place $s \in {}^\bullet t$ such that $M(s) < 2\cdot W(s,t)$, meaning that the marking of the control places does not allow *auto-concurrency*.
- *quasi-safe* if, for every transition t, there is a control place $s \in {}^\bullet t$ such that $M(s) \leq 1$; note that this implies ac-freeness.

3 An algebra of boxes

To model concurrent systems, we will use a class of labelled nets called *asynchronous boxes*. To model operations on such nets, we will use another class of labelled nets, called *operator boxes*, and the net substitution meta-operator (called also refinement [2]), which allows one to substitute transitions in an operator box by possibly complex asynchronous boxes.

3.1 Asynchronous boxes

An *asynchronous box* (or, shortly, a box) is a labelled net Σ such that each transition is labelled by a multiaction in $\mathsf{m}\mathbb{A}$, and Σ itself is:

- *ex-restricted*: there is at least one entry place and at least one exit place;
- $\mathbb{B}$*-restricted*: for every $b \in \mathbb{B}$, there is exactly one b-labelled place;
- *control-restricted*: for every transition t there is at least one control place in ${}^\bullet t$, and at least one control place in $t^\bullet$.

A box Σ is *static* (resp. *dynamic*) if $M_\Sigma^{ctr} = \varnothing$ (resp. $M_\Sigma^{ctr} \neq \varnothing$) and all the markings reachable from M_Σ^{ctr}, ${}^\circ\Sigma$ or Σ° in the box Σ^{ctr} are both clean and ac-free. The asynchronous boxes, static boxes and dynamic boxes will respectively be denoted by abox, abox^{stc} and abox^{dyn}. In what follows, we will only consider finite boxes and operator boxes.

The basic building blocks, from which other static and dynamic boxes of the MBC will be constructed, are the boxes $\Sigma_{\alpha\beta}$, for $\alpha \in \mathsf{m}\mathbb{A}$ and $\beta \in \mathsf{m}\mathbb{L}$. Each $\Sigma_{\alpha\beta}$ is defined as follows. Its set of places is composed of one entry place e, one exit place x, and one place s_b labelled b for each $b \in \mathbb{B}$. It has only one transition $v^{\alpha\beta}$ labelled by α, which has one incoming arc of weight 1 from e and one outgoing arc of weight 1 to x. The other arcs correspond to the links in β and we have: $W_{\Sigma_{\alpha\beta}}(s_b, v^{\alpha\beta}) \stackrel{\text{df}}{=} \beta(b^-)$ and $W_{\Sigma_{\alpha\beta}}(v^{\alpha\beta}, s_b) \stackrel{\text{df}}{=} \beta(b^+)$, for each $b \in \mathbb{B}$. The marking of $\Sigma_{\alpha\beta}$ is empty. Several examples of such boxes are given in figure 2.

Proposition 1 *Let Σ be a box and $\Sigma[U\rangle\Sigma'$.*

1. *If Σ is static, then $U = \{\}$ and $\Sigma = \Sigma'$.*
2. *If Σ is dynamic, then U is a set of transitions* [3] *and Σ' is a dynamic box.* □

Hence being a dynamic box is invariant over any possible evolution.

We will use the following *marking operators*, which modify the marking of a box Σ, where $b \in \mathbb{B}$ and $B \in \mathsf{m}\mathbb{B}$:

- $\Sigma.B$ adds $B(b)$ tokens to the b-labelled open buffer place of Σ, for each $b \in \mathbb{B}$; in particular, $\Sigma.b \stackrel{\text{df}}{=} \Sigma.\{b\}$ adds one token to the b-labelled place of Σ. This operation will be called *buffer stuffing*.
- $\overline{\Sigma}$ (resp. $\underline{\Sigma}$) is Σ with one additional token in each entry (resp. exit) place, *i.e.*, $M_{\overline{\Sigma}} \stackrel{\text{df}}{=} M_\Sigma + {}^\circ\Sigma$ (resp. $M_{\underline{\Sigma}} \stackrel{\text{df}}{=} M_\Sigma + \Sigma^\circ$).
- $\lfloor\Sigma\rfloor$ is Σ with all the tokens in its control places removed, and $\lVert\Sigma\rVert$ is Σ with the empty marking. Both notations extend component-wise to tuples of boxes.

Proposition 2 *Let Σ be a box and $B, B' \in \mathsf{m}\mathbb{B}$.*

1. *$\underline{\Sigma}$ is static* iff *$\Sigma.B$ is static, and Σ is dynamic* iff *$\Sigma.B$ is dynamic.*
2. *$\overline{\Sigma}$ is dynamic* iff *Σ is static* iff *$\underline{\Sigma}$ is dynamic.*
3. *$\Sigma.\varnothing = \Sigma$, $\Sigma.B.B' = \Sigma.(B + B')$, $\overline{\Sigma}.B = \overline{\Sigma.B}$ and $\underline{\Sigma}.B = \underline{\Sigma.B}$.*
4. *If Σ is static or dynamic, then $\lfloor\Sigma\rfloor$ and $\lVert\Sigma\rVert$ are static boxes.*
5. *If Σ is static, then $\lfloor\Sigma\rfloor = \Sigma$.*
6. *$\lVert\lfloor\Sigma\rfloor\rVert = \lVert\lfloor\Sigma\rfloor\rVert = \lfloor\lVert\Sigma\rVert\rfloor = \lVert\Sigma\rVert$.*
7. *$\lfloor\Sigma\rfloor.B = \lfloor\Sigma.B\rfloor$, $\lfloor\overline{\Sigma}\rfloor = \lfloor\underline{\Sigma}\rfloor = \lfloor\lfloor\Sigma\rfloor\rfloor = \lfloor\Sigma\rfloor$ and $\lVert\Sigma.B\rVert = \lVert\overline{\Sigma}\rVert = \lVert\underline{\Sigma}\rVert = \lVert\Sigma\rVert$.* □

[3] But if $\Sigma\,[\Gamma\rangle_\lambda\,\Sigma'$, then the labelled step Γ may be a true multiset of multiactions.

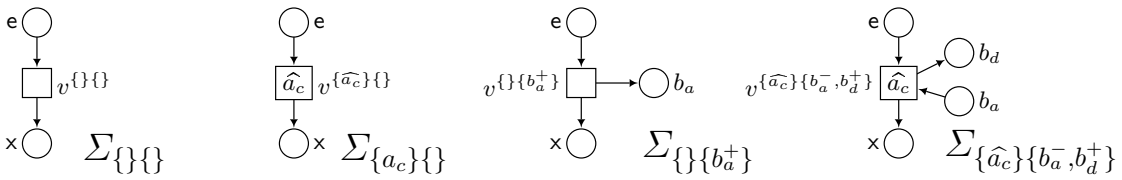


Fig. 2. Four examples of basic boxes.

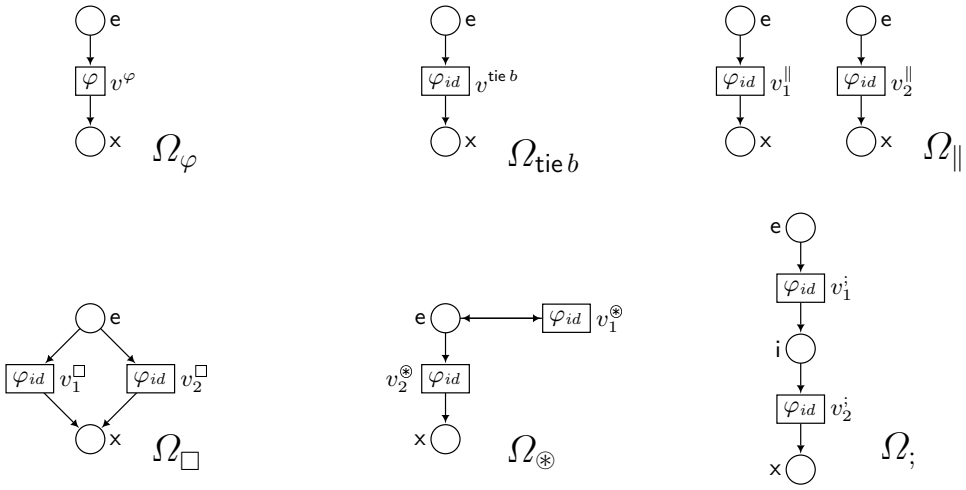


Fig. 3. Operator boxes used in MBC, where $a \in \mathbb{A}$ and $b \in \mathbb{B}$.

3.2 Transition systems

The complete behaviour of a static or dynamic box can be represented by a transition system. And, since we have two kinds of possible steps (based on transitions and on labels), we introduce two kinds of transition systems.

The *full transition system* of a dynamic box Σ is $\mathsf{fts}_\Sigma \stackrel{\text{df}}{=} (V, L, A, \mathit{init})$ where $V \stackrel{\text{df}}{=} [\Sigma\rangle$ is the set of states; $L \stackrel{\text{df}}{=} 2^{T_\Sigma}$ is the set of arc labels; $A \stackrel{\text{df}}{=} \{(\Sigma', U, \Sigma'') \in V \times L \times V \mid \Sigma'\,[U\rangle\,\Sigma''\}$ is the set of arcs; and $\mathit{init} \stackrel{\text{df}}{=} \Sigma$ is the initial state. For a static box Σ, $\mathsf{fts}_\Sigma \stackrel{\text{df}}{=} \mathsf{fts}_{\overline{\Sigma}}$.

The *labelled transition system* of a static or dynamic box Σ, denoted by lts_Σ, is defined as fts_Σ with each arc label U changed to $\lambda_\Sigma(U)$. Note that many different arcs between two states in fts_Σ may lead to a single arc in lts_Σ.

As usual, if (V, L, A, init) and $(V', L', A', \mathit{init}')$ are two transition systems, an *isomorphism* between them is a bijection $\mathsf{iso} : V \to V'$ such that $\mathsf{iso}(\mathit{init}) = \mathit{init}'$ and, for all states $v, w \in V$ and labels $\ell \in L \cup L'$, $(v, \ell, w) \in A$ *iff* $(\mathsf{iso}(v), \ell, \mathsf{iso}(w)) \in A'$. In the consistency results presented in section 5, we construct binary relations which are isomorphisms between transition systems of boxes and the corresponding box expressions.

3.3 Operator boxes

An *operator box* Ω is an unmarked, finite, simple, ex-restricted and control-restricted labelled net with only control places (hence it is not $\mathbb{B}$-restricted) and such that every transition v is labelled by an interface function. For every operator box Ω, we will assume that its transitions $v_1, \ldots, v_n$ are implicitly ordered, and then each n-tuple of nets (or expressions later on) $\boldsymbol{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$, will be referred to as an Ω-*tuple* (or, simply, a *tuple*); we will also use Σ_{v_i} to denote Σ_i, for $i \leq n$. The notation $\boldsymbol{\Sigma}$ will be used in the net substitution operation, denoted by $\Omega(\boldsymbol{\Sigma})$, and defined in the next section.

We will consider four groups of operator boxes for MBC as described below.

Sequential operators. A *sequential* operator box Ω_{sq} is an operator box such that: no place is disconnected; there is exactly one e-labelled place, and exactly one x-labelled place; and, for every transition $v \in T_{\Omega_{sq}}$, $|{}^\bullet v| = |v^\bullet| = 1$ and $\lambda_{\Omega_{sq}}(v) = \varphi_{id}$. That is, Ω_{sq} can be thought of as a *finite automaton* in which each transition will be substituted by a potentially complex box by the net substitution operation. We assume that all the transitions have basic transition identities, and that two distinct operator boxes have disjoint sets of nodes. The domain of application of Ω_{sq} is the set $dom_{\Omega_{sq}}$ comprising all Ω_{sq}-tuples of static and dynamic boxes such that at most one box is dynamic.

Examples of sequential operator boxes are choice $\Omega_\Box$, iteration $\Omega_\circledast$ and sequence $\Omega_;$ depicted in figure 3. They are all binary, with the domain of application $dom_{\Omega_\Box} = dom_{\Omega_\circledast} = dom_{\Omega_;} \stackrel{\text{df}}{=} (\mathsf{abox}^{stc})^2 \cup (\mathsf{abox}^{dyn} \times \mathsf{abox}^{stc}) \cup (\mathsf{abox}^{stc} \times \mathsf{abox}^{dyn})$. We will denote: $\Sigma_1 \,\Box\, \Sigma_2 \stackrel{\text{df}}{=} \Omega_\Box(\Sigma_1, \Sigma_2)$, $\Sigma_1 \circledast \Sigma_2 \stackrel{\text{df}}{=} \Omega_\circledast(\Sigma_1, \Sigma_2)$ and $\Sigma_1 \,;\, \Sigma_2 \stackrel{\text{df}}{=} \Omega_;(\Sigma_1, \Sigma_2)$.

Parallel composition $\Omega_\|$. This is also a binary operator box (see figure 3), but with the domain of application $dom_{\Omega_\|} \stackrel{\text{df}}{=} (\mathsf{abox}^{stc})^2 \cup (\mathsf{abox}^{dyn})^2$, so that its two operands may evolve concurrently. We will denote $\Sigma_1 \| \Sigma_2 \stackrel{\text{df}}{=} \Omega_\|(\Sigma_1, \Sigma_2)$.

Communication interface operators. A unary *communication interface* operator box Ω_φ, shown in figure 3, is parameterised by an interface function $\varphi : \mathsf{mult}(\mathsf{m}\mathbb{A}) \setminus \{\varnothing\} \to \mathsf{m}\mathbb{A}$, and has the domain of application $dom_{\Omega_\varphi} \stackrel{\text{df}}{=} \mathsf{abox}^{stc} \cup \mathsf{abox}^{dyn}$. The role of Ω_φ will be to effect the change of synchronous communication interface specified by φ. It has the domain of application $dom_{\Omega_\varphi} \stackrel{\text{df}}{=} \mathsf{abox}^{stc} \cup \mathsf{abox}^{dyn}$.

Examples of such operators are the scoping operators, which are parameterised by an action $a \in \mathbb{A}$. We will denote $\Sigma\,\mathsf{sc}\,a \stackrel{\text{df}}{=} \Omega_{\mathsf{sc}\,a}(\Sigma)$.

Buffer restriction $\Omega_{\mathsf{tie}\,b}$. Parameterised by a buffer $b \in \mathbb{B}$, this unary operator also has the domain of application $dom_{\Omega_{\mathsf{tie}\,b}} \stackrel{\text{df}}{=} \mathsf{abox}^{stc} \cup \mathsf{abox}^{dyn}$. Buffer restriction will hide the b-labelled open buffer place of the box it is applied to. We will denote $\Sigma\,\mathsf{tie}\,b \stackrel{\text{df}}{=} \Omega_{\mathsf{tie}\,b}(\Sigma)$.

3.4 Net substitution

Throughout the rest of the paper, the identities of transitions in asynchronous boxes will play a key role, especially when defining the SOS semantics of process expressions. For such a model, transition identities will come in the form of finite labelled trees retracing the operators used to construct a box.

We assume that there is a set η of *basic* transition identities and a corresponding set of basic labelled trees with a single node labelled with an element of η. All the transitions in figure 2 and 3 are assumed to be of that kind. To express more complex (unordered) finite trees, or sets of trees, used as transition identities in boxes obtained through net substitution, we will use the following linear notations:

– $v \lhd \mathsf{T}$, where $v \in \eta$ is a basic transition identity and T is a finite set of finite labelled trees, denotes a tree where the trees of the set T are appended to a root labelled with v.
– $v \lhd \mathsf{t}$ denotes the tree $v \lhd \{\mathsf{t}\}$, and $v \blacktriangleleft \mathsf{T}$ denotes the set of trees $\{v \lhd \mathsf{t} \mid \mathsf{t} \in \mathsf{T}\}$.

We assume that place identities may be changed at will to avoid clashes. In particular, when applying net substitution, we will assume that the place sets of the operands are pairwise disjoint; if this is not the case, we rename them in a consistent way. With this assumption, in the following we shall construct new places by grouping the existing ones, *e.g.*, if s_1 and s_2 are places of some operand boxes, then (s_1, s_2) may be the identity of a newly constructed place.

Sequential and parallel operators. Let Ω be a sequential or the parallel operator with transitions $v_1, \ldots, v_n$, and $\boldsymbol{\Sigma} = (\Sigma_1, \ldots, \Sigma_n) = (\Sigma_{v_1}, \ldots, \Sigma_{v_n})$ be a tuple of boxes in dom_Ω. Then $\Omega(\boldsymbol{\Sigma}) = \Phi$ whose components are defined as follows.

The set of transitions of Φ is the set of all trees $v_i \lhd t$ (with $t \in T_{\Sigma_i}$ and $i \in \{1, \ldots, n\}$). The label of each $v_i \lhd t$ is that of t. Each $\mathfrak{i}$-labelled or $\flat$-labelled place $p \in S_{\Sigma_i}$ belongs to S_Φ. Its label and marking are unchanged and for every transition $w \lhd t$, the weight function is given by:

$$W_\Phi(p, w \lhd t) \stackrel{\text{df}}{=} \begin{cases} W_{\Sigma_i}(p,t) & \text{if } w = v_i \\ 0 & \text{otherwise} , \end{cases}$$

and similarly for $W_\Phi(w \lhd t, p)$.

For every place $s \in S_\Omega$ with ${}^\bullet s = \{u_1, \ldots, u_k\}$ and $s^\bullet = \{w_1, \ldots, w_m\}$, we construct in S_Φ all the places of the form $p \stackrel{\text{df}}{=} (x_1, \ldots, x_k, e_1, \ldots, e_m)$, where each x_i is an exit place of Σ_{u_i}, and each e_j is an entry place of Σ_{w_j}. The label of p is that of s, its marking is the sum of the markings of $x_1, \ldots, x_k, e_1, \ldots, e_m$, and for every transition $w \lhd t$, the weight function is given by:

$$W_\Phi(p, w \lhd t) \stackrel{\text{df}}{=} \begin{cases} W_{\Sigma_w}(x_i,t) + W_{\Sigma_w}(e_j,t) & \text{if } w \in {}^\bullet s \cap s^\bullet \text{ and } w = u_i = w_j \\ W_{\Sigma_w}(x_i,t) & \text{if } w \in {}^\bullet s \setminus s^\bullet \text{ and } w = u_i \\ W_{\Sigma_w}(e_j,t) & \text{if } w \in s^\bullet \setminus {}^\bullet s \text{ and } w = w_j \\ 0 & \text{otherwise} , \end{cases}$$

and similarly for $W_{\Omega(\boldsymbol{\Sigma})}(w \lhd t, p)$. We will denote by $\mathbb{P}\langle s \rangle$ the set of all places p as defined above for a given s, and by $\mathbb{P}\langle s, q \rangle$ (resp. $\mathbb{P}\langle s, q, q' \rangle$) the sets of all those $p \in \mathbb{P}\langle s \rangle$ in which q (resp. q and q') is present.

For every $b \in \mathbb{B}$, there is a unique b-labelled place $p^b \stackrel{\text{df}}{=} (p^b_{v_1}, \ldots, p^b_{v_n}) \in S_{\Omega(\boldsymbol{\Sigma})}$, where each $p^b_{v_i}$ is the unique b-labelled place of Σ_i. The marking of p^b is the sum of the markings of the $p^b_{v_i}$'s, and for each transition $w \lhd t$, the weight function is given by:

$$W_{\Omega(\boldsymbol{\Sigma})}(p^b, w \lhd t) \stackrel{\text{df}}{=} W_{\Sigma_w}(p^b_w, t) ,$$

and similarly for $W_{\Omega(\boldsymbol{\Sigma})}(w \lhd t, p^b)$.

Communication interface operators. For a communication interface operator Ω_φ, the intuition behind a multiset Γ in the domain of φ is that some interface change can be applied to any finite multiset of transitions whose labels match the argument, *i.e.*, the non-empty multiset of multiactions Γ. More precisely, such transitions can be synchronised to yield a new transition labelled $\varphi(\Gamma)$. (Note that, since sequential operators as well as the parallel one, use the interface function φ_{id}, no transition label is changed for them.) Hence, the application of a communication interface operator Ω_φ to a box Σ results in a labelled net which is like Σ with the only difference that the set of transitions comprises all trees $t \stackrel{\text{df}}{=} v^\varphi \lhd \{t_1, \ldots, t_l\}$ such that $\{t_1, \ldots, t_l\} \in \mathsf{mult}(T_\Sigma)$ and the multiset $\Lambda \stackrel{\text{df}}{=} \{\lambda_\Sigma(t_1), \ldots, \lambda_\Sigma(t_l)\}$ belongs to the domain of φ. The label of t is $\varphi(\Lambda)$, and for a place p of $\Omega_\varphi(\Sigma)$, the weight function is given by:

$$W_{\Omega_\varphi(\Sigma)}(p,t) \stackrel{\text{df}}{=} \sum_{i=1}^{l} W_\Sigma(p, t_i) ,$$

and similarly for $W_{\Omega_\varphi(\Sigma)}(t,p)$.

Buffer restriction. An application of the buffer restriction operator $\Omega_{\mathsf{tie}\,b}$ to a box Σ results in a labelled net like Σ with the only difference that the identity of each transition $t \in T_\Sigma$ is changed to $v^{\mathsf{tie}\,b} \lhd t$, the label of the only b-labelled place is changed to $\flat$, and a new unmarked disconnected b-labelled place is added to $S_{\Omega_{\mathsf{tie}\,b}(\Sigma)}$.

3.5 Consistency in the box domain

For a constructed net, the property of having an empty control marking is directly linked to the same property about the arguments.

Proposition 3 *Let Ω be any sequential operator box or the parallel composition operator box $\Omega_\|$, $\boldsymbol{\Sigma}$ be any Ω-tuple of boxes, and Σ be any box.*

1. $M^{ctr}_{\Omega(\boldsymbol{\Sigma})} = \varnothing$ iff $M^{ctr}_{\Sigma_{v_i}} = \varnothing$, *for each* $v_i \in T_\Omega$.
2. *If* Σ' *is* $\Omega_\varphi(\Sigma)$ *or* $\Sigma\,\mathsf{tie}\,b$ *or* $\Sigma.B$, *then* $M^{ctr}_{\Sigma'} = \varnothing$ iff $M^{ctr}_{\Sigma} = \varnothing$. □

Moreover, the operation of net substitution always returns a syntactically valid object provided that it is applied to operands belonging to the correct domain. This is shown by the two lemmata and the theorem below.

Lemma 4 *If M is a clean marking of a box Σ such that $M(e) + M(x) \geq 1$, for all $e \in {}^\circ\Sigma$ and $x \in \Sigma^\circ$, then $M^{ctr} \in \{{}^\circ\Sigma, \Sigma^\circ\}$.* □

Lemma 5 *Let $\Omega_{sq}(\boldsymbol{\Sigma})$ be a legal application of a sequential operator box Ω_{sq}, and $z \in T_{\Omega_{sq}}$ be such that $\Sigma = \Sigma_z$ is a dynamic box. Moreover, let s'' be a place in Ω_{sq}, $M \stackrel{\text{df}}{=} M_{\Omega_{sq}(\boldsymbol{\Sigma})}$, ${}^\bullet z = \{s\}$ and $z^\bullet = \{s'\}$.*

1. *If there is a place in $\mathbb{P}\langle s''\rangle$ which is marked at M then $s'' \in \{s, s'\}$.*
2. *If $M^{ctr}_\Sigma = {}^\circ\Sigma$ then $M^{ctr} = \mathbb{P}\langle s\rangle$, and if $M^{ctr}_\Sigma = \Sigma^\circ$ then $M^{ctr} = \mathbb{P}\langle s'\rangle$.*
3. *If $M^{ctr} \geq \mathbb{P}\langle s''\rangle$ then $M^{ctr} = \mathbb{P}\langle s''\rangle$, and one of the following holds:*
   (a) $s'' = s \neq s'$ and $M^{ctr}_\Sigma = {}^\circ\Sigma$.
   (b) $s'' = s' \neq s$ and $M^{ctr}_\Sigma = \Sigma^\circ$.
   (c) $s'' = s = s'$ and $M^{ctr}_\Sigma \in \{{}^\circ\Sigma, \Sigma^\circ\}$. □

Theorem 6. *Let Ω be any operator box and $\boldsymbol{\Sigma} \in dom_\Omega$. Then $\Omega(\boldsymbol{\Sigma})$ is a box with a clean and ac-free marking. Moreover, if all the dynamic boxes (if any) in $\boldsymbol{\Sigma}$ have quasi-safe markings, then the marking of $\Omega(\boldsymbol{\Sigma})$ is also quasi-safe.* □

We finally observe that, if one makes no use of buffer stuffing nor buffer restriction and uses only basic nets of the form $\Sigma_{\alpha\{\}}$, then the net operations described above are similar to those defined in the standard box algebra (see [1–3]), except for the additional b-labelled places, which are all disconnected and unmarked.

4 Relating behaviour and structure of composite boxes

In this section we investigate how the behaviour of composite boxes depends on the behaviours of the boxes being composed.

4.1 Static properties of boxes

An important result from the point of view of developing an algebra of box and box expressions is given next.

Proposition 7 *Let Ω be a sequential operator box, and $\boldsymbol{\Sigma}$ be an Ω-tuple of static boxes.*

1. *If $v \in T_\Omega$ is such that ${}^\circ\Omega = {}^\bullet v$ or ${}^\circ\Omega = v^\bullet$, then $\overline{\Omega(\boldsymbol{\Sigma})} = \Omega(\boldsymbol{\Sigma}')$, where $\boldsymbol{\Sigma}'$ is $\boldsymbol{\Sigma}$ with Σ_v replaced respectively by $\overline{\Sigma_v}$ or $\underline{\Sigma_v}$.*
2. *If $v \in T_\Omega$ is such that $\Omega^\circ = {}^\bullet v$ or $\Omega^\circ = v^\bullet$, then $\underline{\Omega(\boldsymbol{\Sigma})} = \Omega(\boldsymbol{\Sigma}')$, where $\boldsymbol{\Sigma}'$ is $\boldsymbol{\Sigma}$ with Σ_v replaced respectively by $\overline{\Sigma_v}$ or $\underline{\Sigma_v}$.* □

4.2 Structural equivalence

We intend here to capture situations where different applications of a same operator box lead to the same labelled net. Let Ω be an operator box and $\boldsymbol{\Sigma}$, $\boldsymbol{\Theta}$ be Ω-tuples of boxes. We start by defining five auxiliary relations $\equiv^i_\Omega$, in the following way.

- $\boldsymbol{\Sigma} \equiv^0_\Omega \boldsymbol{\Theta}$ if there is a transition $v \in T_\Omega$ and a box Ψ such that ${}^\bullet v = v^\bullet$, $\{\Sigma_v, \Theta_v\} = \{\overline{\Psi}, \underline{\Psi}\}$ and $\Sigma_u = \Theta_u$, for all $u \in T_\Omega \setminus \{v\}$.
- $\boldsymbol{\Sigma} \equiv^1_\Omega \boldsymbol{\Theta}$ if there are two transitions $v \neq w \in T_\Omega$ and two boxes Ψ_v and Ψ_w such that ${}^\bullet v = {}^\bullet w$, $\{(\Sigma_v, \Sigma_w), (\Theta_v, \Theta_w)\} = \{(\overline{\Psi_v}, \Psi_w), (\Psi_v, \overline{\Psi_w})\}$ and $\Sigma_u = \Theta_u$, for all $u \in T_\Omega \setminus \{v, w\}$.
- $\boldsymbol{\Sigma} \equiv^2_\Omega \boldsymbol{\Theta}$ if there are two transitions $v \neq w \in T_\Omega$ and two boxes Ψ_v and Ψ_w such that $v^\bullet = w^\bullet$, $\{(\Sigma_v, \Sigma_w), (\Theta_v, \Theta_w)\} = \{(\underline{\Psi_v}, \Psi_w), (\Psi_v, \underline{\Psi_w})\}$ and $\Sigma_u = \Theta_u$, for all $u \in T_\Omega \setminus \{v, w\}$.
- $\boldsymbol{\Sigma} \equiv^3_\Omega \boldsymbol{\Theta}$ if there are two transitions $v \neq w \in T_\Omega$ and two boxes Ψ_v and Ψ_w such that $v^\bullet = {}^\bullet w$, $\{(\Sigma_v, \Sigma_w), (\Theta_v, \Theta_w)\} = \{(\underline{\Psi_v}, \Psi_w), (\Psi_v, \overline{\Psi_w})\}$ and $\Sigma_u = \Theta_u$, for all $u \in T_\Omega \setminus \{v, w\}$.
- $\boldsymbol{\Sigma} \equiv^4_\Omega \boldsymbol{\Theta}$ if, for each $v \in T_\Omega$, there is a box Ψ_v and two multisets B_v and B'_v over $\mathbb{B}$, such that

$$\sum_{v \in T_\Omega} B_v = \sum_{v \in T_\Omega} B'_v \ ,$$

  and $\Sigma_v = \Psi_v.B_v$ and $\Theta_b = \Psi_v.B'_v$, for all $v \in T_\Omega$.

Then we define $\equiv_\Omega \stackrel{\text{df}}{=} \equiv^4_\Omega \circ \bigcup_{i \in \mathcal{I}} \equiv^i_\Omega$, where $\mathcal{I} = \{0, 1, 2, 3, 5\}$ and $\equiv^5_\Omega \stackrel{\text{df}}{=} id_{\mathsf{abox}}$. Note that $\equiv_{\Omega_\parallel} = \equiv^4_\Omega$ and $\equiv_\Omega = id_{\mathsf{abox}}$, for every unary operator box Ω.

Proposition 8 *Let Ω be an operator box, and $\boldsymbol{\Sigma} \in dom_\Omega$.*

1. *If $\boldsymbol{\Sigma} \equiv_\Omega \boldsymbol{\Theta}$, then $\boldsymbol{\Theta} \in dom_\Omega$ and $[\![\boldsymbol{\Sigma}]\!] = [\![\boldsymbol{\Theta}]\!]$.*
2. *$\equiv_\Omega$ is an equivalence relation on dom_Ω.*
3. *If $\boldsymbol{\Theta} \in dom_\Omega$ and $[\![\boldsymbol{\Sigma}]\!] = [\![\boldsymbol{\Theta}]\!]$, then $\Omega(\boldsymbol{\Sigma}) = \Omega(\boldsymbol{\Theta})$ iff $\boldsymbol{\Sigma} \equiv_\Omega \boldsymbol{\Theta}$.* □

4.3 Dynamic properties of composite nets

In the results presented below, we capture the *behavioural compositionality* of our model, *i.e.*, the way the behaviours of composite nets (in terms of enabled steps) are related to the behaviours of their constituting nets. Basically, we want to establish what steps are enabled by $\Omega(\boldsymbol{\Sigma})$, knowing the steps enabled by the boxes in $\boldsymbol{\Sigma}$.

Proposition 9 *Let $\boldsymbol{\Sigma} \stackrel{\text{df}}{=} (\Sigma_1, \Sigma_2) \in dom_{\Omega_\parallel}$. Then $\mathsf{enabled}(\Omega_\parallel(\boldsymbol{\Sigma}))$ comprises exactly all sets of transitions $U = (v_1^\parallel \blacktriangleleft U_1) \cup (v_2^\parallel \blacktriangleleft U_2)$ such that there is a pair of boxes $\boldsymbol{\Theta}$ satisfying $\boldsymbol{\Theta} \equiv_{\Omega_\parallel} \boldsymbol{\Sigma}$ and $U_i \in \mathsf{enabled}(\Theta_i)$, for $i \in \{1, 2\}$. Moreover, $\Omega_\parallel(\boldsymbol{\Sigma})\,[U\rangle\,\Omega_\parallel(\boldsymbol{\Phi})$, where $\Theta_i\,[U_i\rangle\,\Phi_i$, for $i \in \{1, 2\}$.* □

Proposition 10 *Let Ω be a sequential operator box and $\boldsymbol{\Sigma} \in dom_\Omega$.*

1. *If $\Sigma_i \,[U_i\rangle\, \Theta_i$ (for $i \leq n$), then $\boldsymbol{\Theta} \in dom_\Omega$, and*

$$\Omega(\boldsymbol{\Sigma})\,[(v_1 \blacktriangleleft U_1) \cup \ldots \cup (v_n \blacktriangleleft U_n)\rangle\, \Omega(\boldsymbol{\Theta})\ .$$

2. *If $\Omega(\boldsymbol{\Sigma})\,[U\rangle\,\Theta$, then there are $\boldsymbol{\Psi}, \boldsymbol{\Phi} \in dom_\Omega$ and steps $U_1, \ldots, U_n$ such that $\boldsymbol{\Sigma} \equiv_\Omega \boldsymbol{\Psi}$, $\Psi_i\,[U_i\rangle\,\Phi_i$ (for $i \leq n$), $\Theta = \Omega(\boldsymbol{\Phi})$ and*

$$U = (v_1 \blacktriangleleft U_1) \cup \ldots \cup (v_n \blacktriangleleft U_n)\ .$$

Note: *As a consequence,* $\mathsf{enabled}(\Omega(\boldsymbol{\Sigma}))$ *comprises exactly all sets $(v_1 \blacktriangleleft U_1) \cup \ldots \cup (v_n \blacktriangleleft U_n)$ of transitions such that there is $\boldsymbol{\Psi} \equiv_\Omega \boldsymbol{\Sigma}$ and $U_i \in \mathsf{enabled}(\Psi_i)$ (for $i \leq n$).* □

Notice that for the sequential operators, at most one U_i is non-empty. Otherwise, the property is very similar to that which holds for the parallel operator box.

Proposition 11 *Let Σ be a box, $b \in \mathbb{B}$, $B \in \mathsf{mult}(\{b\})$ and $B' \in \mathsf{m}\mathbb{B}$.*

1. $\mathsf{enabled}(\Sigma\,\mathsf{tie}\,b)$ *comprises exactly all sets of transitions $U = v^{\mathsf{tie}\,b} \blacktriangleleft V$ such that $V \in \mathsf{enabled}(\Sigma)$. Moreover, $\Sigma\,\mathsf{tie}\,b\,[U\rangle\,\Phi\,\mathsf{tie}\,b$, where $\Sigma\,[V\rangle\,\Phi$.*
2. $\mathsf{enabled}((\Sigma\,\mathsf{tie}\,b).B)$ *comprises exactly all sets of transitions $U \in \mathsf{enabled}(\Sigma\,\mathsf{tie}\,b)$. Moreover, $(\Sigma\,\mathsf{tie}\,b).B\,[U\rangle\,\Phi.B$, where $\Sigma\,\mathsf{tie}\,b\,[U\rangle\,\Phi$.*
3. $\mathsf{enabled}(\Sigma)$ *is a subset of* $\mathsf{enabled}(\Sigma.B')$. *Moreover, if $\Sigma\,[U\rangle\,\Phi$ then $\Sigma.B'\,[U\rangle\,\Phi.B'$.* □

That the converse of the last property does not hold may be illustrated by a simple counter-example: the dynamic box $\overline{\Sigma_{\{\}\{b^-\}}}$ only allows the empty step, while $\overline{\Sigma_{\{\}\{b^-\}}.b}\,[\{v^{\{\}\{b^-\}}\}\rangle\,\underline{\Sigma_{\{\}\{b^-\}}}$.

Proposition 12 *Let Σ be a static or dynamic box, and Ω_φ be a communication interface operator box.*

1. *If $\Sigma\,[U_1 \uplus \ldots \uplus U_k\rangle\,\Theta$ and each $\lambda_\Sigma(U_i)$ belongs to the domain of φ, then*

$$\Omega_\varphi(\Sigma)\,[\{v^\varphi \lhd U_1, \ldots, v^\varphi \lhd U_k\}\rangle\,\Omega_\varphi(\Theta)\ .$$

2. *If $\Omega_\varphi(\Sigma)\,[U\rangle\,\Psi$ then there is a box Θ and steps of transitions $U_1, \ldots, U_k$ such that $\Psi = \Omega_\varphi(\Theta)$, $U = \{v^\varphi \lhd U_1, \ldots, v^\varphi \lhd U_k\}$, and $\Sigma\,[U_1 \uplus \ldots \uplus U_k\rangle\,\Theta$.*

Note: *As a consequence,* $\mathsf{enabled}(\Omega_\varphi(\Sigma))$ *comprises exactly all $U = \{v^\varphi \lhd U_1, \ldots, v^\varphi \lhd U_k\}$ such that $U_1 \uplus \ldots \uplus U_k \in \mathsf{enabled}(\Sigma)$ and each $\lambda_\Sigma(U_i)$ belongs to the domain of φ.* □

The behaviours of the basic asynchronous boxes of MBC are captured below.

Proposition 13 *Let $B \in \mathsf{m}\mathbb{B}$ and $\Sigma_{\alpha\beta}$ be one of the basic boxes. If $B(b) \geq \beta(b^-)$, for each $b \in \mathbb{B}$, then the only non empty steps of $\overline{\Sigma_{\alpha\beta}.B}$ are*

$$\overline{\Sigma_{\alpha\beta}.B}\ \ [\{v^{\alpha\beta}\}\rangle\ \ \underline{\Sigma_{\alpha\beta}.\left(B - \sum_{b\in B} \beta(b^-)\cdot b + \sum_{b\in B} \beta(b^+)\cdot b\right)}\ ;$$

otherwise, $\overline{\Sigma_{\alpha\beta}.B}$ has no non-empty step. □

Various important consequences may be derived from the results presented above; in particular that the way static and dynamic boxes are composed in MBC guarantees that the result is a static or dynamic box when the domain of application of the operators is respected.

Proposition 14 *Let Ω be an operator box of MBC and $\boldsymbol{\Sigma} \in dom_\Omega$. Then every net derivable from $\Omega(\boldsymbol{\Sigma})$ is of the form $\Omega(\boldsymbol{\Theta})$, where $\boldsymbol{\Theta} \in dom_\Omega$ and $[\![\boldsymbol{\Theta}]\!] = [\![\boldsymbol{\Sigma}]\!]$. Moreover, if no box in $\boldsymbol{\Sigma}$ is dynamic, then every net derivable from $\overline{\Omega(\boldsymbol{\Sigma})}$ or $\underline{\Omega(\boldsymbol{\Sigma})}$ is of the form $\Omega(\boldsymbol{\Theta})$, where $\boldsymbol{\Theta} \in dom_\Omega$ and $[\![\boldsymbol{\Theta}]\!] = [\![\boldsymbol{\Sigma}]\!]$.* □

Theorem 15. *Every composite net of MBC is a quasi-safe static or dynamic box. Moreover, it is static* iff *the marking operators $\overline{(\cdot)}, \underline{(\cdot)}$ are not used, unless in the scope of the $\lfloor\cdot\rfloor$ or $[\![\cdot]\!]$ operators.* □

5 An algebra of asynchronous box expressions

We consider an algebra of process expressions over the signature:

$$\mathcal{C} \cup \{\, \overline{(\cdot)}\,,\, \underline{(\cdot)}\,\} \cup \{\, \|\,,\ ;\ ,\ \Box\,,\ \circledast\,\} \cup \{\, \mathsf{sc}\, a \mid a \in \mathbb{A}\} \cup \{\, \mathsf{tie}\, b, .b \mid b \in \mathbb{B}\}\ , \tag{1}$$

where $\mathcal{C} \stackrel{\text{df}}{=} \{\alpha\beta \mid \alpha \in \mathsf{m}\mathbb{A}, \beta \in \mathsf{m}\mathbb{L}\}$ are the constants; the binary operators $\|$, $;$, $\Box$ and $\circledast$ will be used in the infix mode; the unary operators $\mathsf{sc}\, a$, $\mathsf{tie}\, b$ and $.b$ will be used in the postfix mode; and $\overline{(\cdot)}$ and $\underline{(\cdot)}$ are two positional unary operators (the position of the argument being given by the dot).

There are two classes of process expressions corresponding to the static and dynamic boxes, viz. the *static* and *dynamic* expressions, denoted respectively by aexpr^{stc} and aexpr^{dyn}. Collectively, we will refer to them as the (asynchronous) *box expressions*, aexpr. Their syntax is given by:

$$\begin{array}{lrcl}
\mathsf{aexpr}^{stc} & E & ::= & \alpha\beta \mid E\,\mathsf{sc}\, a \mid E\,\mathsf{tie}\, b \mid \ E.b \mid E \| E \mid E \Box E \mid \ E; E \mid E \circledast E \\
\mathsf{aexpr}^{dyn} & D & ::= & \overline{E} \ \mid \ \underline{E} \ \mid D\,\mathsf{sc}\, a \mid D\,\mathsf{tie}\, b \mid \ D.b \ \mid \ D \| D \mid D \Box E \mid E \Box D \\
 & & & \mid D; E \mid \ E; D \mid D \circledast E \mid E \circledast D
\end{array} \tag{2}$$

where $\alpha\beta \in \mathcal{C}$, $a \in \mathbb{A}$ and $b \in \mathbb{B}$. Moreover, we will use F to denote any static or dynamic expression.

We also use the notations $\lfloor F \rfloor$ and $[\![F]\!]$ yielding static expressions, where $\lfloor F \rfloor$ is F with all occurrences of $\overline{(\cdot)}$ and $\underline{(\cdot)}$ removed, and $[\![F]\!]$ is $\lfloor F \rfloor$ with all occurrences of $.b$ removed. Note that we do not need terms of the form $F.B$ since $F.\{b, \ldots, b'\}$ would be equivalent to $F.b \cdots .b'$ (but such terms can be used as a convenient shorthand).

Essentially, a box expression encodes the structure of a box, together with the current marking of the control places (using overbars and underbars) and of the buffer places (using the $.b$'s). Thus, a box expression $\overline{E}$ represents E in its initial state (in terms of nets, this corresponds to the initially marked box of E). Similarly, $\underline{E}$ represents E in its final state. Note that the $.b$ notation is needed for static as well as for dynamic box expressions because the dormant part of a dynamic box expression may still have $.b$'s which are later needed in the active part. For instance, $D \stackrel{\text{df}}{=} \{a\}\{b^+\}.b\,;\overline{\{a_f\}\{b^-\}}$ has a static component with a $.b$ in it, and may be transformed into an equivalent $D' \stackrel{\text{df}}{=} \{a\}\{b^+\};\overline{\{a_f\}\{b^-\}.b}$ (see section 5.2).

5.1 Denotational semantics

The denotational semantics of box expressions is given in the form of a mapping $\mathsf{box} : \mathsf{aexpr} \to \mathsf{abox}$, defined homomorphically by induction on their structure, following the syntax (2). Below, $\alpha\beta \in \mathcal{C}$, $a \in \mathbb{A}$, $b \in \mathbb{B}$, una stands for any unary operator ($\mathsf{sc}\, a$, $\mathsf{tie}\, b$ or $.b$), and bin for any binary operator ($\|$, $\Box$, $;$ or $\circledast$).

$$\begin{array}{c}
\mathsf{box}(\alpha\beta) \stackrel{\text{df}}{=} \Sigma_{\alpha\beta} \qquad \mathsf{box}(\overline{E}) \stackrel{\text{df}}{=} \overline{\mathsf{box}(E)} \qquad \mathsf{box}(\underline{E}) \stackrel{\text{df}}{=} \underline{\mathsf{box}(E)} \\
\mathsf{box}(F\,\mathsf{una}) \stackrel{\text{df}}{=} \mathsf{box}(F)\,\mathsf{una} \qquad \mathsf{box}(F_1\,\mathsf{bin}\,F_2) \stackrel{\text{df}}{=} \mathsf{box}(F_1)\,\mathsf{bin}\,\mathsf{box}(F_2)\ .
\end{array} \tag{3}$$

The semantical mapping always returns a box, and the property of corresponding to a static or dynamic box has been captured by the syntax (2).

Theorem 16. *Let F be a box expression.*

1. $\mathsf{box}(F)$ *is a static or dynamic box.*
2. $\mathsf{box}(F)$ *is a static box* iff *F is a static box expression.* □

5.2 Structural similarity relation

We define the *structural similarity* relation on box expressions, denoted by $\equiv$, as the least equivalence relation on box expressions such that all the equations in table 1 are satisfied. These rules directly follow those of the ABC and PNA models.

CON1	$\dfrac{F \equiv F'}{F\,\mathsf{una} \equiv F'\,\mathsf{una}}$	CON2	$\dfrac{F_1 \equiv F_1', F_2 \equiv F_2'}{F_1\,\mathsf{bin}\,F_2 \equiv F_1'\,\mathsf{bin}\,F_2'}$
ENT	$\dfrac{E \equiv E'}{\overline{E} \equiv \overline{E'}}$	EX	$\dfrac{E \equiv E'}{\underline{E} \equiv \underline{E'}}$
OPL	$(F.b)\,\mathsf{bin}\,F' \equiv (F\,\mathsf{bin}\,F').b$	OPR	$F\,\mathsf{bin}(F'.b) \equiv (F\,\mathsf{bin}\,F').b$
E1	$\overline{E}\,\mathsf{una} \equiv \overline{E\,\mathsf{una}}$	X1	$\underline{E}\,\mathsf{una} \equiv \underline{E\,\mathsf{una}}$
B1	$(F.b)\,\mathsf{una} \equiv (F\,\mathsf{una}).b$ if $\mathsf{una} \neq \mathsf{tie}\,b$	IS1	$\overline{E_1; E_2} \equiv \overline{E_1}; E_2$
IS2	$\underline{E_1}; E_2 \equiv E_1; \overline{E_2}$	IS3	$E_1; \underline{E_2} \equiv \underline{E_1; E_2}$
IPAR1	$\overline{E_1 \| E_2} \equiv \overline{E_1} \| \overline{E_2}$	IPAR2	$\underline{E_1} \| \underline{E_2} \equiv \underline{E_1 \| E_2}$
IC1L	$\overline{E_1 \,\square\, E_2} \equiv \overline{E_1} \,\square\, E_2$	IC1R	$\overline{E_1 \,\square\, E_2} \equiv E_1 \,\square\, \overline{E_2}$
IC2L	$\underline{E_1} \,\square\, E_2 \equiv \underline{E_1 \,\square\, E_2}$	IC2R	$E_1 \,\square\, \underline{E_2} \equiv \underline{E_1 \,\square\, E_2}$
IIT1	$\overline{E_1 \circledast E_2} \equiv \overline{E_1} \circledast E_2$	IIT2	$\underline{E_1} \circledast E_2 \equiv \overline{E_1} \circledast E_2$
IIT3	$\underline{E_1} \circledast E_2 \equiv E_1 \circledast \overline{E_2}$	IIT4	$\overline{E_1} \circledast E_2 \equiv E_1 \circledast \overline{E_2}$
IIT5	$E_1 \circledast \underline{E_2} \equiv \underline{E_1 \circledast E_2}$		

Table 1. Structural similarity relation for MBC, where $b \in \mathbb{B}$, una stands for any unary MBC operator and bin stands for any binary MBC operator.

It may be observed that, due to the rules CON1-2, ENT and EX, the equivalence relation so defined is in fact a congruence for all the operators of the algebra. It is easy to see that the structural similarity relation is closed in the domain of expressions, in the sense that, if a box expression matches one of the sides of any rule then, the other side defines a legal box expression. Moreover, it preserves the types of box expressions (static or dynamic), and captures the fact that box expressions have the same net translation, as shown below.

Theorem 17. *Let F_1 and F_2 be box expressions.*

1. *If* $F_1 \equiv F_2$ *then* $\lfloor F_1 \rfloor \equiv \lfloor F_2 \rfloor$, $[\![F_1]\!] = [\![F_2]\!]$ *and* $\mathsf{box}(F_1) = \mathsf{box}(F_2)$.
2. *If* $[\![F_1]\!] = [\![F_2]\!]$, *then* $\mathsf{box}(F_1) = \mathsf{box}(F_2)$ iff $F_1 \equiv F_2$. □

That the precondition $[\![F_1]\!] = [\![F_2]\!]$ is needed in the second part of the last result may be justified by the counter-example $F_1 \stackrel{\text{df}}{=} \{a\}\{\}\,\mathsf{sc}\,a$ and $F_2 \stackrel{\text{df}}{=} \{\widehat{a}\}\{\}\,\mathsf{sc}\,a$ for which $F_1 \not\equiv F_2$ but $\mathsf{box}(F_1) = \mathsf{box}(F_2)$ (no transition is left in the nets by the scoping operation).

Theorem 18. *Let F be a box expression.*

1. $\mathsf{box}(F) = \overline{\lfloor \mathsf{box}(F) \rfloor}$ iff $F \equiv \overline{\lfloor F \rfloor}$.
2. $\mathsf{box}(F) = \underline{\lfloor \mathsf{box}(F) \rfloor}$ iff $F \equiv \underline{\lfloor F \rfloor}$. □

In the first case above we say that F is an *initial* expression, and in the second a *final* one.

In developing the operational semantics of the box algebra, we first introduce operational rules. They are based on transitions of the nets providing the denotational semantics of box expressions. Based on these, we will formulate our key consistency result. Then we will introduce the label based rules, together with the derived consistency results.

5.3 Transition based operational semantics

Consider the set $\mathbb{T}$ of all transition trees in the boxes derived through the box mapping. It is easy to check that each $t \in \mathbb{T}$ has always the same label in all the boxes derived through the box mapping where it occurs; it will be denoted by $\lambda(t)$.

The first operational semantics we will consider has moves of the form $F \stackrel{U}{\longrightarrow} F'$ such that F and F' are box expressions and $U \in \mathbb{U} \stackrel{\text{df}}{=} \mathsf{mult}(\mathbb{T})$. The idea here is that U is a valid step for the boxes associated with F and F', *i.e.*, that $\mathsf{box}(F)\,[U\rangle\,\mathsf{box}(F')$, as stated by theorem 20.

Formally, we define a ternary relation $\longrightarrow$ which is the least relation comprising all $(F, U, F') \in \mathsf{aexpr} \times \mathbb{U} \times \mathsf{aexpr}$ such that the relations in table 2 hold. Notice that we use $F \stackrel{U}{\longrightarrow} F'$ to denote $(F, U, F') \in \longrightarrow$. In the definition of EOP we make no restriction on U_1 and U_2 but the domain of application of bin will ensure that this rule will always be used with the correct static/dynamic mixture of boxes. For instance, in the case of the choice operator, one of U_1 and U_2 is necessarily empty, and in the rule of the $\mathsf{sc}\,a$ operator, each U_i contains only transitions whose labels can be scoped together.

EA $\quad \overline{\alpha\beta.\left(\sum_{b\in\mathbb{B}} \beta(b^-)\cdot b\right)} \stackrel{\{v^{\alpha\beta}\}}{\longrightarrow} \underline{\alpha\beta.\left(\sum_{b\in\mathbb{B}} \beta(b^+)\cdot b\right)}$

EQ1 $\quad F \stackrel{\{\}}{\longrightarrow} F$

EQ2 $\quad \dfrac{F \equiv F',\ F' \stackrel{U}{\longrightarrow} F'',\ F'' \equiv F'''}{F \stackrel{U}{\longrightarrow} F'''}$

EBUF $\quad \dfrac{F \stackrel{U}{\longrightarrow} F'}{F.b \stackrel{U}{\longrightarrow} F'.b}$

ETIE $\quad \dfrac{F \stackrel{U}{\longrightarrow} F'}{F\,\mathsf{tie}\,b \stackrel{v^{\mathsf{tie}\,b}\blacktriangleleft U}{\longrightarrow} F'\,\mathsf{tie}\,b}$

ESC $\quad \dfrac{F \stackrel{U_1+\ldots+U_k}{\longrightarrow} F'}{F\,\mathsf{sc}\,a \stackrel{\{v^{\mathsf{sc}\,a}\triangleleft U_1,\ldots,v^{\mathsf{sc}\,a}\triangleleft U_k\}}{\longrightarrow} F'\,\mathsf{sc}\,a}$ if $\forall i : \lambda(U_i) \in dom(\varphi_{\mathsf{sc}\,a})$

EOP $\quad \dfrac{F_1 \stackrel{U_1}{\longrightarrow} F_1',\ F_2 \stackrel{U_2}{\longrightarrow} F_2'}{F_1\,\mathsf{bin}\,F_2 \stackrel{(v_1^{\mathsf{bin}}\blacktriangleleft U_1)\cup(v_2^{\mathsf{bin}}\blacktriangleleft U_2)}{\longrightarrow} F_1'\,\mathsf{bin}\,F_2'}$

Table 2. Transition based operational semantics for MBC, where $a \in \mathbb{A}$, $b \in \mathbb{B}$ and bin stands for any binary MBC operator.

$\overline{\textsc{PayRise}}$

$$\equiv \ (\ \overline{(\{a_r\}\{\} \circledast \{a_c\}\{\})} \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\})} \,\mathsf{tie}\, b_d)$$
$$\| \ \overline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{E1, IPAR1}$$

$$\equiv \ (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \overline{\{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{IIT1, IC1R}$$

$$\xrightarrow{\{t_1\}} \ (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\underline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) \qquad \text{EA, EOP}$$
$$\| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \underline{\{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{ESC, ETIE}$$

$$\equiv \ (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\underline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ (\underline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{IC2R}$$

$$\equiv \ (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ (\overline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\})} \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{IIT2, IC1L}$$

$$\equiv \ (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ (\overline{((\{a_r, \widehat{a_y}\}\{\}} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{IS1}$$

$$\xrightarrow{\{t_2\}} \ (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\underline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) \qquad \text{EA, EOP}$$
$$\| \ (((\underline{\{a_r, \widehat{a_y}\}\{\}} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{ESY, ETIE}$$

$$\equiv \ (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\underline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ (((\{a_r, \widehat{a_y}\}\{\} \ ; \ \overline{\{\}\{b_a^+\}}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{IS2}$$

$$\xrightarrow{\{t_3\}} \ (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\underline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ (((\{a_r, \widehat{a_y}\}\{\} \ ; \ \underline{\{\}\{b_a^+\}.\ b_a)} \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{EA}$$

$$\equiv \ (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\underline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d)$$
$$\| \ (\underline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}.\ b_a) \ \square \ \{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{IC2L}$$

$$\equiv \ (\ (\{a_r\}\{\} \circledast \overline{\{a_c\}\{\}}) \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \overline{\{\widehat{a_c}\}\{b_a^-, b_d^+\}.\ b_a}) \,\mathsf{tie}\, b_d) \qquad \text{IIT3, OPR}$$
$$\| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \overline{\{a_f\}\{\}}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{OPL, B1}$$

$$\xrightarrow{\{t_4, t_5\}} \ (\ (\{a_r\}\{\} \circledast \underline{\{a_c\}\{\})} \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \underline{\{\widehat{a_c}\}\{b_a^-, b_d^+\}.\ b_d)} \,\mathsf{tie}\, b_d) \qquad \text{EA, EOP, ESY}$$
$$\| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \underline{\{a_f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c \qquad \text{ERS, ETIE}$$

$$\equiv \ \underline{\textsc{PayRise}} \qquad \text{IIT5, IC2R, IPAR2, X1}$$

Table 3. Execution scenario I

We will now derive some properties of the derivation rules. First, an empty move always relates two structurally equivalent box expressions.

Proposition 19 *Let F and F' be two box expressions. Then, $F \xrightarrow{\{\}} F'$ iff $F \equiv F'$.* □

Next, a move of the operational semantics transforms a box expression into another expression with a structurally equivalent underlying static expression, and the move generated is a valid step for the corresponding boxes. We interpret this as establishing the soundness of the operational semantics of box expressions. We then reverse the implication obtaining the completeness of the operational semantics.

Theorem 20. *Let F be a box expression.*

1. *If $F \xrightarrow{U} F'$, then F' is a box expression such that $\mathsf{box}(F)\,[U\rangle\,\mathsf{box}(F')$ and $\llbracket F \rrbracket = \llbracket F' \rrbracket$.*
2. *If $\mathsf{box}(F)\,[U\rangle\,\Sigma$, then there is a box expression F' such that $\mathsf{box}(F') = \Sigma$ and $F \xrightarrow{U} F'$.* □

5.4 Consistency of the denotational and operational semantics

The consistency between the denotational and the operational semantics of box expressions will be formulated in terms of the transition systems they generate. This will be possible since, thanks to theorem 20, we are now in a position to relate transition systems generated by a box expression and the corresponding box.

Let D be a dynamic box expression. We will use $[D\rangle$ to denote all the box expressions derivable from D, *i.e.*, the least set of expressions containing D such that if $D' \in [D\rangle$ and $D' \xrightarrow{U} D''$, for some $U \in \mathbb{U}$, then $D'' \in [D\rangle$. Moreover, $[D]_{\equiv}$ will denote the equivalence class of $\equiv$ containing D. The *full transition system* of D is $\mathsf{fts}_D \stackrel{\text{df}}{=} (V, L, A, \mathit{init})$, where $V \stackrel{\text{df}}{=} \{[D']_{\equiv} \mid D' \in [D\rangle\}$ is the set of states; $L \stackrel{\text{df}}{=} \mathbb{U}$ is the set of arc labels; $A \stackrel{\text{df}}{=} \left\{([D']_{\equiv}, U, [D'']_{\equiv}) \in V \times \mathbb{U} \times V \mid D' \xrightarrow{U} D''\right\}$ is the set of arcs; and $\mathit{init} \stackrel{\text{df}}{=} [D]_{\equiv}$ is the initial state. For a static box expression E, $\mathsf{fts}_E \stackrel{\text{df}}{=} \mathsf{fts}_{\overline{E}}$.

Note that we base transition systems of box expressions on the equivalence classes of $\equiv$, rather than on box expressions themselves, since we may have $D \xrightarrow{\{\}} D'$ for two different expressions D and D', whereas in the domain of boxes, $\Sigma\,[\{\}\rangle\,\Theta$ always implies $\Sigma = \Theta$.

We now state a fundamental result which demonstrates that the operational and denotational semantics of a box expression capture the same behaviour, in arguably the strongest sense.

Theorem 21. *For every box expression F, $\mathsf{iso}_F \stackrel{\text{df}}{=} \left\{([F']_{\equiv}, \mathsf{box}(F')) \;\middle|\; [F']_{\equiv} \text{ is a node of } \mathsf{fts}_F\right\}$ is an isomorphism between the full transition systems fts_F and $\mathsf{fts}_{\mathsf{box}(F)}$. Moreover, iso_F preserves the property of being in an initial or final state.*[4] □

5.5 Label based operational semantics

First, we retain the structural similarity relation $\equiv$ on box expressions without any change. Next, we define moves of the form $F \xrightarrow{\Gamma} F'$, where F and F' are box expressions as before, and $\Gamma \in \mathsf{mult}(\mathsf{m}\mathbb{A})$, as shown in table 4.

The two types of operational semantics are clearly related; essentially, each label based move is a transition based move with only transitions labels being recorded.

Proposition 22 *Let F be a box expression and $\Gamma \in \mathbb{L}$. Then $F \xrightarrow{\Gamma} F'$* iff *there is $U \in \mathbb{U}$ such that $F \xrightarrow{U} F'$ and $\lambda(U) = \Gamma$.* □

[4] In terms of box expression (*cf.* theorem 18) or box net (*cf.* section 2.5), not *w.r.t.* the transition system.

LA	$\overline{\alpha\beta.\left(\sum_{b\in\mathbb{B}}\beta(b^-)\cdot b\right)} \xrightarrow{\{\alpha\}} \underline{\alpha\beta.\left(\sum_{b\in\mathbb{B}}\beta(b^+)\cdot b\right)}$	LOP	$\dfrac{F_1 \xrightarrow{\Gamma_1} F_1',\ F_2 \xrightarrow{\Gamma_2} F_2'}{F_1 \,\mathsf{bin}\, F_2 \xrightarrow{\Gamma_1+\Gamma_2} F_1' \,\mathsf{bin}\, F_2'}$
LQ1	$F \xrightarrow{\{\}} F$	LQ2	$\dfrac{F \equiv F', F' \xrightarrow{\Gamma} F'', F'' \equiv F'''}{F \xrightarrow{\Gamma} F'''}$
LBUF	$\dfrac{F \xrightarrow{\Gamma} F'}{F.b \xrightarrow{\Gamma} F'.b}$	LTIE	$\dfrac{F \xrightarrow{\Gamma} F'}{F \,\mathsf{tie}\, b \xrightarrow{\Gamma} F' \,\mathsf{tie}\, b}$
LSC	$\dfrac{F \xrightarrow{\Gamma_1+\ldots+\Gamma_k} F'}{F \,\mathsf{sc}\, a \xrightarrow{\varphi_{\mathsf{sc}\,a}(\Gamma_1)+\ldots+\varphi_{\mathsf{sc}\,a}(\Gamma_k)} F' \,\mathsf{sc}\, a}$ if $\forall i : \Gamma_i \in dom(\varphi_{\mathsf{sc}\,a})$		

Table 4. Label based operational semantics for MBC, where $a \in \mathbb{A}$, $b \in \mathbb{B}$ and bin stands for any binary MBC operator.

The results concerning transition based operational semantics directly extend to the label based one. Let F be a box expression. In view of proposition 22, the label based operational semantics of F is faithfully captured by the *labelled transition system* of F, denoted by lts_F, and defined as fts_F with each arc label U changed to $\lambda(U)$. The consistency result for the label based operational semantics can then be formulated thus.

Theorem 23. *For every box expression F, $\mathsf{iso}_F \stackrel{\text{df}}{=} \left\{([F']_\equiv, \mathsf{box}(F')) \;\middle|\; [F']_\equiv \text{ is a node of } \mathsf{lts}_F\right\}$ is an isomorphism between the labelled transition systems lts_F and $\mathsf{lts}_{\mathsf{box}(F)}$. Moreover, iso_F preserves the property of being in an initial or final state.* □

6 Conclusion

We extended the synchronous and asynchronous capabilities of ABC by allowing the use of multisets of actions as well as that of multisets of links. These modifications has been introduced at the level of ABC process expressions and also at the level of their associated structured operational semantics. The resulting framework, called MBC, comprises an algebra of process expressions and an algebra of nets which are consistent in the sense that an expression and the corresponding net generate isomorphic transition systems. As it was the case with PNA, MBC allows also for expressing compositionally processes with multi-way synchronisation, what was not possible with ABC.

In the process of extending PNA with asynchronous communication, the extension we still need is the recursion. Moreover, the enhancement presented in this paper is crucial when one investigates high-level extensions of such formalisms. Actually, a high-level synchronous action may encode a set of low-level actions, and similarly, a high-level communication link may correspond to a set of low-level links. The work presented here is based on multisets which are a natural generalisation of sets. Our future work will go in both directions.

References

1. E. Best, R. Devillers and J. Hall: The Petri Box Calculus: a New Causal Algebra with Multilabel Communication. In: *Advances in Petri Nets 1992*, G. Rozenberg (Ed.). Springer, Lecture Notes in Computer Science 609 (1992) 21–69.

$\overline{\textsc{PayRise}}$

$$
\begin{array}{lll}
\equiv & (\ \overline{(\{a_r\}\{\} \circledast \{a_c\}\{\})} \ \| \ ((\overline{\{\widehat{a_r}, \widehat{a_r}\}\{\} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ \overline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{E1, IPAR1} \\
\equiv & (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\overline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \overline{\{a_r, \widehat{a_n}\}\{\}}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IIT1, IC1R} \\
\xrightarrow{\{\{a_n\}\}} & (\ \underline{(\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \text{LA, LOP} \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \underline{\{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{LSC, LTIE} \\
\equiv & (\ \underline{(\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ \underline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IC2R} \\
\equiv & (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\overline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ \overline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IIT2, IC1L} \\
\equiv & (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\overline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ (\overline{(\{a_r, \widehat{a_y}\}\{\}} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IS1} \\
\xrightarrow{\{\{a_y\}\}} & (\ \underline{(\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \text{LA, LOP} \\
& \quad \| \ (\underline{(\{a_r, \widehat{a_y}\}\{\}} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{LSC, LTIE} \\
\equiv & (\ \underline{(\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \overline{\{\}\{b_a^+\}}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IS2} \\
\xrightarrow{\{\{\}\}} & (\ \underline{(\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \underline{\{\}\{b_a^+\}.\ b_a)} \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{LA} \\
\equiv & (\ \underline{(\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ (\underline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}.\ b_a) \ \square \ \{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IC2L} \\
\equiv & (\ (\{a_r\}\{\} \circledast \overline{\{a_c\}\{\}}) \ \| \ ((\{\widehat{a_r}, \widehat{a_r}\}\{\} \circledast \overline{\{\widehat{a_c}\}\{b_a^-, b_d^+\}.\ b_a}) \,\mathsf{tie}\, b_d) & \text{IIT3, B1} \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \overline{\{a_f\}\{\}}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{OPR, OPL} \\
\xrightarrow{\{\{a_f\},\{\}\}} & (\ (\{a_r\}\{\} \circledast \underline{\{a_c\}\{\})} \ \| \ ((\{\widehat{a_r}, \widehat{a_r}\}\{\} \circledast \underline{\{\widehat{a_c}\}\{b_a^-, b_d^+\}.\ b_a)} \,\mathsf{tie}\, b_d) & \text{LA, LOP} \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \underline{\{a_f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{LSC, LTIE} \\
\equiv & \underline{\textsc{PayRise}} & \text{IIT5, IC2R} \\
& & \text{IPAR2, X1}
\end{array}
$$

Table 5. Execution scenario I.

$\overline{\text{PayRise}}$

$$
\begin{array}{lll}
\equiv & (\ \overline{(\{a_r\}\{\} \circledast \{a_c\}\{\})} \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\})} \,\mathsf{tie}\, b_d) & \\
& \quad \| \ \overline{((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \{f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{E1, IPAR1} \\
\equiv & (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ (\overline{(\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \overline{\{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IIT1, IC1R} \\
\xrightarrow{\{\{a_n\}\}} & (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \text{LA, LOP} \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \underline{\{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{LSC, LTIE} \\
\equiv & (\ (\underline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\underline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ (\underline{(\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\})} \circledast \{a_f\}\{\}) \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IC2R} \\
\equiv & (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\overline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \overline{\{a_f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{IIT2, IIT3} \\
\xrightarrow{\{\{a_f\}\}} & (\ (\overline{\{a_r\}\{\}} \circledast \{a_c\}\{\}) \ \| \ ((\overline{\{\widehat{a_r}, \widehat{a_r}\}\{\}} \circledast \{\widehat{a_c}\}\{b_a^-, b_d^+\}) \,\mathsf{tie}\, b_d) & \\
& \quad \| \ ((\{a_r, \widehat{a_y}\}\{\} \ ; \ \{\}\{b_a^+\}) \ \square \ \{a_r, \widehat{a_n}\}\{\}) \circledast \underline{\{a_f\}\{\})} \ ) \ \mathsf{sc} \ a_r \ \mathsf{sc} \ a_c & \text{LA, LOP, LSC}
\end{array}
$$

Table 6. Execution scenario II.

2. E. Best, R. Devillers and M. Koutny: A Unified Model for Nets and Process Algebras. In: *Handbook of Process Algebra*, J. A. Bergstra, A. Ponse, S. A. Smolka, (Eds.). Elsevier (2001) 873–944.
3. E. Best, R. Devillers and M. Koutny: *Petri Net Algebra*. EATCS Monographs on TCS, Springer (2001).
4. E. Best and R. P. Hopkins: B(PN)^2 – a Basic Petri Net Programming Notation. Proc. of *PARLE '93*, A. Bode, M. Reeve and G. Wolf (Eds.). Springer, Lecture Notes in Computer Science 694 (1993) 379–390.
5. R. Devillers, H. Klaudel, M. Koutny and F. Pommereau. An Algebra of Non-safe Petri Boxes. *AMAST'2002*, Volume 2422 of *Lecture Notes in Computer Science*, pages 192 – 207, Springer-Verlag, 2002.
6. H. Klaudel and F. Pommereau: Asynchronous links in the PBC and M-nets. Proc. of *ASIAN'99*, P. S. Thiagarajan and R. Yap (Eds.). Springer, Lecture Notes in Computer Science 1742 (1999) 190–200.
7. G. D. Plotkin: A Structural Approach to Operational Semantics. Technical Report FN-19, Computer Science Department, University of Aarhus (1981).