Asynchronous Box Calculus

R. Devillers\textsuperscript{1}, H. Klaudel\textsuperscript{2}, M. Koutny\textsuperscript{3}, and F. Pommereneau\textsuperscript{2}

\textsuperscript{1} Département d’Informatique, Université Libre de Bruxelles, B-1050 Bruxelles, Belgium. rdevill@ulb.ac.be
\textsuperscript{2} Université Paris 12, LACL, 61 avenue du général de Gaulle 94010 Créteil, France. \{klaudel, pommereneau\}@univ-paris12.fr
\textsuperscript{3} Department of Computing Science, Univ. of Newcastle upon Tyne, NE1 7RU, United Kingdom. Maciej.Koutny@newcastle.ac.uk

\textbf{Abstract.} The starting point of this paper is an algebraic Petri net framework allowing one to express net compositions, such as iteration and parallel composition, as well as transition synchronisation and restriction. We enrich the original model by introducing new constructs supporting asynchronous interprocess communication. Such a communication is made possible thanks to special ‘buffer’ places where different transitions (processes) may deposit and remove tokens. We also provide an abstraction mechanism, which hides buffer places, effectively making them private to the processes communicating through them. We then provide a characterisation of the operational step sequence semantics of composite nets. These developments lead to an algebra of process expressions, whose constants and operators directly correspond to those used in the Petri net framework. Such a correspondence is used to associate nets to process expressions in a fully compositional way. Moreover, a structural characterisation of the Petri net semantics of composite nets guides the definition of a structured operational semantics of process expressions. That the resulting algebra of expressions is consistent with the net algebra is demonstrated by showing that an expression and the corresponding net generate isomorphic transition systems. This results in the \textit{Asynchronous Box Calculus} (or ABC), which is a coherent dual model, based on Petri nets and process expressions, suitable for modelling and analysing distributed systems whose components can interact using both synchronous and asynchronous communication.

\textbf{Keywords:} Petri nets, process algebra, synchronous and asynchronous communication, structured operational semantics.

1 Introduction

This paper is concerned with the theme of relating process algebras, such as CCS \textsuperscript{21} and CSP \textsuperscript{13}, and Petri nets \textsuperscript{26}. In general, the approaches proposed in the literature aim at providing a Petri net semantics to process algebras whose definition has been given independently of any Petri nets semantics as, \textit{e.g.}, in \textsuperscript{7, 8, 10–12, 15, 22, 23, 27}. Another approach is to translate elements from Petri nets into process algebras such as ACP \textsuperscript{1} as done, \textit{e.g.}, in \textsuperscript{2}.
A specific framework within which the present paper is set, is the Petri Box Calculus (PBC [3]). The PBC has been designed with the aim of allowing a compositional Petri net semantics of nondeterministic and concurrent programming languages [6], and was later extended into a more generic Petri Net Algebra (PNA [4, 5]). The model is composed of an algebra of process expressions (called box expressions) with a fully compositional translation into labelled safe Petri nets (called boxes). We will now introduce those of its aspects which will be needed in the work presented here.

1.1 An algebra of nets and process expressions

The variant of the PNA model relevant to this paper considers the following operators: sequence $E_1; E_2$ (the execution of $E_1$ is followed by that of $E_2$); choice $E_1 \square E_2$ (either $E_1$ or $E_2$ can be executed); parallel composition $E_1 \parallel E_2$ ($E_1$ and $E_2$ can be executed concurrently); iteration $E_1 \oplus E_2$ ($E_1$ can be executed an arbitrary number of times, and is followed by $E_2$); and scoping $E \mathsf{sc} a$ (all handshake synchronisations involving pairs of $a$- and $\widetilde{a}$-labelled transitions are enforced, and after that the synchronising transitions may no longer be executed). We illustrate these constructs using an example based on three process expressions:

$$\begin{align*}
\mathsf{CritSect} &\triangleq ((a_1; r_1) \square (a_2; r_2)) \oplus f \\
\mathsf{User}_1 &\triangleq \widehat{a}_1; \widehat{r}_1 \\
\mathsf{User}_2 &\triangleq \widehat{a}_2; \widehat{r}_2 ,
\end{align*}$$

modelling a critical section and two user processes. The atomic actions $a_1$ and $a_2$ (together with the matching actions $\widehat{a}_1$ and $\widehat{a}_2$) model the granting of access to a shared resource, $r_1$ and $r_2$ (together with the matching $\widehat{r}_1$ and $\widehat{r}_2$) model its release, and $f$ models a final action. The system where these three processes operate in parallel\footnote{We assume that operators like parallel composition associate to the right.} is

$$\mathsf{PreMutex} \triangleq \mathsf{User}_1 \parallel \mathsf{CritSect} \parallel \mathsf{User}_2 ,$$

and the net (called a box) on the left of figure 1 is the translation of this process expression (itself called a box expression). Note that, in a box, places are labelled by their status (e for entry, x for exit, and i for internal) while transitions are labelled by CCS-like synchronous communication actions, such as $a_1$, $\widehat{a}_1$, and $\tau$ (similarly as in CCS, $\tau$ represents an internal action).

The box expression $\mathsf{PreMutex}$ and the corresponding box correctly specify the three constituent processes, but it does not allow for interprocess communication. This is, however, easily achieved by applying the scoping w.r.t. the synchronisation actions $a_1$ and $r_1$, which results in the box expression

$$\mathsf{Mutex} \triangleq \mathsf{PreMutex} \mathsf{sc} a_1 \mathsf{sc} a_2 \mathsf{sc} r_1 \mathsf{sc} r_2 ,$$

and the corresponding box is shown on the right of figure 1.
The operational semantics of box expressions is given through SOS rules in Plotkin's style [24]. However, instead of expressing the evolutions through rules modifying the structure of the expressions, like $a \cdot E \xrightarrow{a} E$ 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 in figure 2, where the net on the left represents \texttt{PreMutex} after the two user processes have terminated, and the critical section is still in its initial state, while the net on the right represents the initial state of the system specified by \texttt{Mutex}, i.e., that corresponding to the box expression \texttt{Mutex}.

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

$$
\texttt{USER}_1 \parallel \texttt{CritSect} \parallel \texttt{USER}_2 \equiv \texttt{USER}_1 \parallel \texttt{CritSect} \parallel \texttt{USER}_2 \equiv \texttt{USER}_1 \parallel \texttt{CritSect} \parallel \texttt{USER}_2 ,
$$
and evolution rules specifying when we may have a state change, e.g., one can derive that

\[
(((a_1; r_1) \square (a_2; r_2)) @ f) \Rightarrow (((a_1; r_1) \square (a_2; r_2)) @ f).
\]

As it was shown in [5, 19], the two algebras constituting PNA are fully compatible, in the sense that a box expression and the corresponding box generate isomorphic transition systems.

### 1.2 Asynchronous communication

Recently, [17] (and then [9]²) introduced a novel feature into the model of box expressions and nets, aimed at the modelling of asynchronous interprocess communication (such an extension was used, in particular, to model time-dependent concurrent systems). The basic devices facilitating this are new kinds of basic actions, for sending, receiving and testing for the presence of tokens in buffer places. We introduce all three using the following very simple process expressions, modelling a producer, consumer and tester processes (each process can perform exactly one action, after which it terminates):

\[
\text{PRODONE} \equiv pb^+ \\
\text{CONSONE} \equiv cb^- \\
\text{TESTONE} \equiv tb^\pm.
\]

In the above, \(pb^+\) is an atomic action whose role is to ‘produce’ a token (resource) and deposit it in a buffer place identified by \(b\); in doing so, it generates the visible label \(p\). Similarly, \(cb^-\) is an atomic action which ‘consumes’ a resource from buffer \(b\), generating the visible label \(c\). And \(tb^\pm\) combines the effect of the two previous atomic actions, effectively ‘testing’ for the presence of a token in the buffer while generating a visible label \(t\). The Petri nets rendering of the above three processes are shown in figure 3, where the buffer places are identified by the \(b\) labels.

An intuitive meaning of a \(b\)-labelled place is that some transitions can insert tokens into it (perhaps several in a single step), while others can (later) remove them, thus effecting asynchronous communication. We can, for example, compose in parallel the simple producer and consumer processes, forming a new box expression \(\text{SYST} \equiv \text{PRODONE} \parallel \text{CONSONE}\). The corresponding box is shown on the left of figure 4, where the original parallel composition of PNA has been modified so that the two \(b\)-labelled buffer places are merged into a single one, also \(b\)-labelled. As a consequence, this place can later be merged with other similar buffer places, e.g., we might form \(\text{TESTSyst} \equiv \text{Syst} \parallel \text{TESTONE}\) with the corresponding box shown in the middle of figure 4. The reader may easily verify that the asynchronous communication outlined above is now indeed feasible, after placing one token in each of the \(e\)-labelled places of the box of \(\text{Syst}\) (or \(\text{TESTSyst}\)).

---

² Note that, with respect to [17], the notations and technical details are slightly different in [9], but the principles remain the same. In this paper, we adopt the notations from [9].
The scheme described thus far still needs an abstraction mechanism for the asynchronous communication part, which comes in the form of the buffer restriction operator, denoted by \texttt{tie}. The \texttt{tie} operator \texttt{w.r.t.} a buffer \texttt{b}, changes the status of the \texttt{b}-labelled buffer place into a \texttt{b}-labelled one, indicating that the place can no longer be merged with other buffer places. In effect, the \texttt{b}-labelled places may be viewed as internal places. This is illustrated on the right of figure 4, for the box expression \texttt{HIDE\textsc{Syst} \texttt{def} \textsc{Syst} \texttt{tie} \texttt{b}}.

The resulting model, called the \textbf{Asynchronous Box Calculus} (or ABC), is no longer based on safe Petri nets since one cannot expect the buffer places to be safe; indeed, they may even be unbounded. However, the extension is such that no auto-concurrency is ever generated (\textit{i.e.}, no transition can be executed concurrently with itself), as further discussed below.

### 1.3 About this paper

In [17], the asynchronous communication was introduced for the first time and its application illustrated (see also [16,25] where it is used to give an algebraic semantics to a parallel programming language), but its true impact on the theory of boxes and box expressions was not considered. A first step in this direction was made in the conference paper [9], and the main aim of this paper is to introduce and comprehensively analyse the ABC model. In doing so, we introduce new features to the original framework. Apart from introducing devices supporting
asynchronous interprocess communication, we also remove the restriction of the standard PNA theory to consider only 1-safe boxes (such a property was maintained in order to support a straightforward concurrency semantics of boxes and expressions based on causal partial orders). The 1-safeness is now replaced by auto-concurrency freeness which is a property guaranteeing powerful concurrency semantics in terms of event structures as shown in [14], and thus ensuring that the resulting model is still highly relevant from a theoretical point of view. In particular, and unlike in [9], we will be in a position to remove some awkward restrictions of the standard PNA theory where, e.g., no parallel composition directly under an iteration is allowed.

The model will be based on two algebras, an algebra of box expressions and an algebra of boxes, extending their counterparts developed in the original PNA framework. On the technical level, we will introduce specific syntactic constructs manipulating the tokens in buffer places, and the buffer restriction operator. The original operational semantics rules will be augmented, yielding a system of SOS rules providing the operational semantics of ABC expressions. The two algebras forming ABC will be related through a mapping which, for any box expression, will return a corresponding box with an isomorphic transition system.

This paper will not address recursion which allows one to use expressions such as \( X \stackrel{at}{=} \cdots \) where \( X \) can appear recursively in its definition (directly or indirectly). Such an extension is left for future research. However, this is not an important limitation here since recursion is a very special feature which often leads to infinite nets and thus does not fall into the scope of the present framework.

The structure of the paper is as follows. The next section presents three examples of systems employing asynchronous communication. Section 3 describes the class of labelled nets on which ABC is based. In section 4, we introduce the net algebra part of ABC; in particular, we define a number of operators, either directly, or by using an auxiliary net substitution meta-operator. Section 5 investigates the relationship between the structure and behaviour of composite nets. In section 6, we introduce an algebra of process expressions which forms the second part of the ABC model. 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 their labels. We also present there our main result that a box expression and the corresponding box generate isomorphic transition systems.

To streamline the presentation, all the proofs are presented in appendix A. Appendix B contains auxiliary results as well as generalisations of the results listed in the main body of the paper.

2 Three examples

In order to provide more intuition about the way the ABC algebra is defined and used, this section presents further examples built upon three iteration-based
processes, respectively modelling a producer, consumer and tester of some resource:

\[
\begin{align*}
\text{Prod} & \equiv \text{ProdOne} \circ f = pb^+ \circ f \\
\text{Cons} & \equiv \text{ConsOne} \circ f = cb^- \circ f \\
\text{Test} & \equiv \text{TestOne} \circ f = tb^\perp \circ f ,
\end{align*}
\]

where ProdOne, ConsOne and TestOne are as in the introduction. For example, since ProdOne constitutes the iterative part of Prod, the latter can repeatedly send a token to a \( b \)-labelled buffer place. Tokens produced in such a way can then be removed, also repeatedly, by the ConsOne part of the Cons process, provided that Prod and Cons operate in parallel. The \( f \) is in each case a finishing action.

**Example I.** The example, given on the left of figure 5, models a concurrent system composed of two producers and one consumer operating in parallel:

\[
\text{Syst}_1 \equiv s ; ((\text{Prod} || \text{Prod}) || \text{Cons}) \circ \text{tie } b .
\]

The two-producers/one-consumer system is encapsulated by an application of the buffer restriction operator. This makes the buffer place \( b \)-labelled, and so no longer available for merging with other buffer places. The whole system is preceded by a ‘startup’ action \( s \).

**Example II.** The encapsulating feature of buffer restriction is further illustrated by the second example:

\[
\text{Syst}_{II} \equiv (\text{ProdOne} ; ((\text{Prod} || \text{Cons}) \circ \text{tie } b) ; \text{ConsOne}) \circ \text{tie } b ,
\]

shown in the middle of figure 5. This example will be used to illustrate the result of a nested application of buffer restriction.

**Example III.** The last example, on the right of figure 5, shows a system with one producer, one consumer and two tester processes whose role is to check for the presence of tokens in the buffer:

\[
\text{Syst}_{III} \equiv \text{Prod} || \text{Cons} || \text{Test} || \text{Test} .
\]

Such an example allows one to illustrate a concurrent access to the buffer place. Notice that this system is not closed \( w.r.t. \) asynchronous communications; it would be necessary to apply buffer restriction on \( b \) in order to make private \( (i.e., b \)-labelled) the \( b \)-labelled buffer place.

### 2.1 Three system evolutions

We will now describe three evolutions, or execution scenarios, for the systems given above. In each case, we will use step sequences as a formal device for capturing concurrent behaviours, and assume that the system starts from its implicit initial state. Thus, for example, we consider SystI rather than SystI.
Fig. 5. Boxes for the three execution scenarios.
Scenario I. Consider the net $\Sigma_I$ in figure 5, also shown in figure 6 with its transition names, and the following evolution:

- the system is started up by executing the $s$-labelled transition;
- the two producers send a token each to the $b$-labelled buffer place;
- the consumer takes one of the two tokens from the buffer place and, at the same time, the first producer sends another token there;
- the two producers and the consumer finish their operation by simultaneously executing the three $f$-labelled transitions.

Such a scenario corresponds to the following step sequence:

$$\Sigma_I \left[\{t_1\} \{t_2, t_3\} \{t_2, t_4\} \{t_5, t_6, t_7\}\right] \Sigma_I' ,$$

where $\Sigma_I'$ is $\Sigma_I$ with two tokens in the buffer place, one token in each of the $x$-labelled places, and no token elsewhere. In terms of labelled step sequences, the scenario is represented by

$$\Sigma_I \left[\{s\} \{p, p\} \{p, c\} \{f, f, f\}\right] \Sigma_I' .$$

Since each $x$-labelled place has exactly one token, we will say that $\Sigma_I'$ is in a final marking (or state).

Fig. 6. The box of $\text{SYST}_I$ with transition names for the first execution scenario.

In the next two examples, we only consider labelled steps as these are usually sufficient to capture the relevant behavioural information. However, in order to obtain the desired properties of boxes and box expressions, the reasoning will have to be carried out first using the more detailed view provided by transition steps.

Scenario II. Consider the net $\Sigma_{II}$ in figure 5, and the following evolution:

- the system begins by executing the topmost $p$-labelled transition, and puts a token in the external (i.e., right) buffer place;
the producer sends twice in a row one token to the internal buffer place;
the consumer takes one of the two tokens from this place;
the producer and the consumer finish their operation by simultaneously ex-
cuting the \( f \)-labelled transitions;
the system finishes by firing the bottom \( e \)-labelled transition, which removes
the token from the external buffer place.

Such a scenario corresponds to the following labelled step sequence:

\[
\Sigma_{II} \left[ \{p\} \{f\} \{e\}, \{\{f\}\{e\}\} \right] \Sigma_{II}',
\]

where \( \Sigma_{II}' \) is \( \Sigma_{II} \) with one token in the internal buffer place and the \( x \)-labelled
place, and no token elsewhere, so that \( \Sigma_{II}' \) is in a final state.

\textit{Scenario III.} Consider the net \( \Sigma_{III} \) in figure 5, and the following evolution:

- the producer sends in a row two tokens to the buffer place;
- in a single step, the consumer removes one token, the first tester checks for
  the presence of the other one, and the producer sends another token to the
  buffer place;
- both testers check for the presence of tokens in parallel.

Such a scenario corresponds to the following labelled step sequence:

\[
\Sigma_{III} \left[ \{p\} \{e, t, p\} \{t, t\} \right] \Sigma_{III}',
\]

where \( \Sigma_{III}' \) is \( \Sigma_{III} \) with two additional tokens in the buffer place. Notice that
\( \Sigma_{III}' \) is not in a final state.

3 Preliminaries

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

3.1 Multisets

A \textit{multiset} over a set \( X \) is a function \( \mu : X \rightarrow \mathbb{N} \). We denote by \( \text{mult}(X) \) the set
of all finite multisets \( \mu \) over \( X \), \textit{i.e.}, those satisfying \( \sum_{x \in X} \mu(x) < \infty \). We will
write \( \mu \leq \mu' \) if the domain \( X \) of \( \mu \) is included in that of the multiset \( \mu' \), and
\( \mu(x) \leq \mu'(x) \), for all \( x \in X \). An element \( x \in X \) belongs to \( \mu \), 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 \( \mu \) 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, \textit{e.g.}, \{\( a, a, \tau \)\} denotes a multiset \( \mu \) such that
\( \mu(a) = 2, \mu(\tau) = 1 \) and \( \mu(x) = 0 \) for all \( x \in X \setminus \{a, \tau\} \).
3.2 Actions, buffers and synchronisation

We assume that there is a set $A_{\tau}$ of (atomic) actions representing synchronous interface activities used, in particular, to model handshake communication. We will also assume that, as in CCS, $A_{\tau} = A \cup \{\tau\}$ and for every $a \in A_{\tau}$, $\hat{a}$ is an action in $\hat{A}$ such that $\hat{a} = a$. In addition to the set of atomic actions, there is a finite set $B$ of buffer symbols (or buffers) for asynchronous interprocess communications.\(^3\)

We need a device to express formally that, e.g., two concurrent actions should synchronise. To this end, we will use scoping interface functions $\varphi_{\leq A}: \text{mult}(A_{\tau}) \setminus \{\emptyset\} \rightarrow A_{\tau}$, for $a \in A$. The function $\varphi_{\leq A}$ is partial, its domain is $\{\{c\} \mid c \in A_{\tau}\setminus\{a, \hat{a}\}\} \cup \{\{a, \hat{a}\}\}$, and $\varphi_{\leq A}(\{c\}) \equiv c$ for all $c \in A_{\tau}\setminus\{a, \hat{a}\}$ and $\varphi_{\leq A}(\{a, \hat{a}\}) = \tau$. Such a function will be used to enforce CCS-like synchronisations. More generally, a (communication) interface function is any partial function $\varphi: \text{mult}(A_{\tau}) \setminus \{\emptyset\} \rightarrow A_{\tau}$, and another commonly used interface function is the identity $\varphi_{\text{id}}$ whose domain is $\{\{a\} \mid a \in A_{\tau}\}$ and $\varphi_{\text{id}}(\{a\}) \equiv a$ for all $a \in A_{\tau}$.

3.3 Labelled nets

A (marked) labelled net is, in the present framework, a tuple $\Sigma \equiv (S, T, W, \lambda, M)$ 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 $\mathbb{N}$.
- $\lambda$ is a labelling for places and transitions such that $\lambda(s)$ is a symbol in $\{e, i, x, b\} \cup B$, for every place $s \in S$; and $\lambda(t)$ is an interface function $\varphi$ or an action in $A_{\tau}$, for every transition $t \in T$.
- $M$ is a marking, i.e., a multisets over $S$ (in other words, a mapping from the set of places $S$ to $\mathbb{N}$).

Moreover, $\Sigma$ 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. If the labelling is $b$ then $s$ is a closed buffer place, and if it is a buffer symbol $b \in B$, then $s$ is an open buffer place. Collectively, the $e$, $i$ and $x$-labelled places are called control (flow) places. Moreover, the set of all entry (resp. exit) places will be denoted by $^e\Sigma$ (resp. $^x\Sigma$). We shall also use $M^{\text{cfr}}$ and $M^{\text{op}}$ to denote $M$ restricted to the control places and to the open buffer places, respectively; finally, $\Sigma^{\text{cfr}}$ is $\Sigma$ 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 $\Sigma$).

We adopt the standard rules about representing nets as directed graphs; moreover, double-headed arrows will represent self-loops. To avoid ambiguity,

\(^3\) The finiteness of $B$ is not essential, but it will allow us to consider only finite nets, as for technical reasons a box will contain a buffer place for each $b \in B$. We shall assume that $e, x, i, b \notin B$. 
A deep sequence of $x$ is a (possibly empty) sequence of steps, the first of the second.

We will denote this by $\langle x \rangle$, a notation derived from the concept of a single step.

The definition of a (deep) step sequence.

We will denote the set of all sequences of steps, the first of the second, as $\Sigma^\omega$. In some cases, these sequences will be stored in a file, called a deep sequence.

The concept of a deep sequence is derived from the concept of a single step.

A deep sequence is a (possibly empty) sequence of steps in the order of the steps.

In particular, we will denote $\langle x \rangle$, where there is a single step $x$.

\[ (i) \langle x \rangle = \langle x \rangle \]

In this case, the function $\langle x \rangle$ is denoted as $\langle x \rangle$.

The set of all sequences of steps, the first of the second, is $\Sigma^\omega$. In some cases, these sequences will be stored in a file, called a deep sequence.

The concept of a deep sequence is derived from the concept of a single step.

A deep sequence is a (possibly empty) sequence of steps in the order of the steps.

In particular, we will denote $\langle x \rangle$, where there is a single step $x$.

\[ (i) \langle x \rangle = \langle x \rangle \]

In this case, the function $\langle x \rangle$ is denoted as $\langle x \rangle$.

The set of all sequences of steps, the first of the second, is $\Sigma^\omega$. In some cases, these sequences will be stored in a file, called a deep sequence.

The concept of a deep sequence is derived from the concept of a single step.

A deep sequence is a (possibly empty) sequence of steps in the order of the steps.

In particular, we will denote $\langle x \rangle$, where there is a single step $x$.

\[ (i) \langle x \rangle = \langle x \rangle \]
3.5 Clean, ac-free and quasi-safe markings

A marking $M$ of $\Sigma$ is:

- **clean** if $M^{\text{ctr}} \geq \Sigma \Rightarrow M^{\text{ctr}} = \Sigma$ and $M^{\text{ctr}} \geq \Sigma^\circ \Rightarrow M^{\text{ctr}} = \Sigma^\circ$.
  
  If $M^{\text{ctr}} = \Sigma$, we will say that $\Sigma$ is in an **initial** state (or marking), and if $M^{\text{ctr}} = \Sigma^\circ$, we will say that $\Sigma$ is in a **final** state (or marking).

- **ac-free** if, for every transition $t$, there is a control place $s \in \#$ 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 $s \in \#$ such that $M(s) \leq 1$; note that this implies ac-freeness.

4 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 [4]), which allows one to substitute transitions in an operator box by possibly complex asynchronous boxes.

4.1 Asynchronous boxes

An **asynchronous box** (or, shortly, a box) is a labelled net $\Sigma$ such that each transition is labelled by an action in $A_\tau$, and $\Sigma$ itself is:

- **$A_\tau$-restricted**: there is at least one entry place and at least one exit place.
- **$B$-restricted**: for every $b \in B$, there is exactly one $b$-labelled place.
- **control-restricted**: for every transition $t$ there is at least one control place in $\#$, and at least one control place in $\#^\circ$.

A box $\Sigma$ is **static** (resp. **dynamic** if $M^{\text{ctr}}_\Sigma = \emptyset$ (resp. $M^{\text{ctr}}_\Sigma \not= \emptyset$) and all the markings reachable from $M^{\text{ctr}}_\Sigma = \Sigma$ or $\Sigma^\circ$ in the box $\Sigma^{\text{ctr}}$ are both clean and ac-free. The asynchronous boxes, static boxes and dynamic boxes will respectively be denoted by $\text{abox}$, $\text{abox}^{\text{con}}$ and $\text{abox}^{\text{dyn}}$. In what follows, we will only consider finite asynchronous boxes and operator boxes.

The top row in figure 7 shows four kinds of static boxes $\Sigma_\alpha$, used in ABC, where $\alpha \in A \equiv A_\tau \cup \{ab^+, ab^-, ab^k \mid a \in A_\tau \land b \in B\}$. These are the basic building blocks, from which other static and dynamic boxes of the Asynchronous Box Calculus will be constructed.

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

1. If $\Sigma$ is static, then $U = \emptyset$ and $\Sigma = \Sigma'$.
2. If $\Sigma$ is dynamic, then $U$ is a set of transitions and $\Sigma'$ is a dynamic box.

---

But if $\Sigma[U] \Sigma'$, then the labelled step $\Gamma$ may be a true multiset of actions; see, e.g., scenario I in section 2.
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 \( \Sigma \), where \( b \in \mathbb{B} \) and \( B \in \text{mult}(\mathbb{B}) \):

- \( \Sigma.B \) adds \( B(b) \) tokens to the \( b \)-labelled open buffer place of \( \Sigma \), for each \( b \in \mathbb{B} \); in particular, \( \Sigma.b \overset{\Delta}{=} \Sigma.\{b\} \) adds one token to the \( b \)-labelled place of \( \Sigma \). This operation will be called buffer stuffing.
- \( \Sigma \) (resp. \( \Sigma \)) is \( \Sigma \) with one additional token in each entry (resp. exit) place, i.e., \( M^\Sigma = M_\Sigma + \circ \Sigma \) (resp. \( M^\Sigma = M_\Sigma + \circ \Sigma \)).
- \( \{ \Sigma \} \) is \( \Sigma \) with all the tokens in its control places removed, and \( [\Sigma] \) is \( \Sigma \) with the empty marking. Both notations extend component-wise to tuples of boxes.

**Proposition 2.** Let \( \Sigma \) be a box and \( B, B' \in \text{mult}(\mathbb{B}) \).

1. \( \Sigma \) is static iff \( \Sigma.B \) is static, and \( \Sigma \) is dynamic iff \( \Sigma.B \) is dynamic.
2. \( \Sigma \) is dynamic iff \( \Sigma \) is static iff \( \Sigma \) is dynamic.
3. \( \Sigma.\emptyset = \Sigma, \quad \Sigma.B.B' = \Sigma.(B + B'), \quad \Sigma.B = \Sigma.B + \Sigma.B \) and \( \Sigma.B = \Sigma.B \).
4. If \( \Sigma \) is static or dynamic, then \( \{ \Sigma \} \) and \( [\Sigma] \) are static boxes.
5. If \( \Sigma \) is static, then \( \{ \Sigma \} = \Sigma \).
6. \( \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} \).
7. \( \{ \Sigma.B \} = \{ \Sigma \} \) and \( \{ \Sigma.B \} = \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} = \{ [\Sigma] \} \).

### 4.2 Transition systems

The complete behavior of a static or dynamic box can be represented by a transition system. And, since we have two kinds of possible steps, we introduce two kinds of transition systems.

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

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

As usual, if \( (V, L, A, \text{init}) \) and \( (V', L', A', \text{init}') \) are two transition systems, an **isomorphism** between them is a bijection \( \text{iso} : V \to V' \) such that \( \text{iso}(\text{init}) = \text{init}' \) and, for all states \( v, w \in V \) and labels \( \ell \in L \cup L' \), \( (v, \ell, w) \in A \) iff \( (\text{iso}(v), \ell, \text{iso}(w)) \in A' \). In the consistency results presented in section 6, we will construct binary relations which will be proved to be isomorphisms between transition systems of boxes and the corresponding box expressions.
4.3 Operator boxes

An operator box $\Omega$ 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 $\Omega$, 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) $\Sigma = (\Sigma_1, \ldots, \Sigma_n)$, will be referred to as an $\Omega$-tuple (or, simply, a tuple); we will also use $\Sigma_i$ to denote $\Sigma_i$, for $i \leq n$. The notation $\Sigma$ will be used in the net substitution operation, denoted by $\Omega(\Sigma)$, and defined in the next section.

In the main body of this paper, we will consider four groups of operator boxes for ABC, which are either binary ($n = 2$, typically used in infix notation) or unary ($n = 1$, typically used in postfix notation), as described below and shown in figure 7.
Choice $\Omega_{\Box}$, iteration $\Omega_{\oplus}$ and sequence $\Omega_{\cdot}$. The first three operator boxes specify different ways to sequentially compose behaviours. They are all binary, with the domain of application $\text{dom}_{\Omega_{\Box}} = \text{dom}_{\Omega_{\oplus}} = \text{dom}_{\Omega_{\cdot}} \triangleq (\text{box}^{\text{stc}})^2 \cup (\text{box}^{\text{dyn}} \times \text{box}^{\text{stc}}) \cup (\text{box}^{\text{stc}} \times \text{box}^{\text{dyn}})$, so that at most one of their two operands is dynamic. We will denote: $\Sigma_1 \Box \Sigma_2 \triangleq \Omega_{\Box}(\Sigma_1, \Sigma_2)$, $\Sigma_1 \oplus \Sigma_2 \triangleq \Omega_{\oplus}(\Sigma_1, \Sigma_2)$ and $\Sigma_1 ; \Sigma_2 \triangleq \Omega_{\cdot}(\Sigma_1, \Sigma_2)$.

Parallel composition $\Omega_{\parallel}$. This is also a binary operator box, but with the domain of application $\text{dom}_{\Omega_{\parallel}} \triangleq (\text{box}^{\text{stc}})^2 \cup (\text{box}^{\text{dyn}})^2$, so that its two operands may evolve concurrently. We will denote $\Sigma_1 \| \Sigma_2 \triangleq \Omega_{\parallel}(\Sigma_1, \Sigma_2)$.

Scoping $\Omega_{\text{sc}}$. Parameterised by a communication action $a \in \mathbb{A}$, this is a unary operator with the domain of application $\text{dom}_{\Omega_{\text{sc}}} \triangleq \text{box}^{\text{stc}} \cup \text{box}^{\text{dyn}}$. The change of the synchronous communication interface is captured by $\nu_{\text{sc}} a$, already defined; essentially, this forces the synchronisation of the pairs of actions $a$ and $\bar{a}$. We will denote $\Sigma \text{sc} a \triangleq \Omega_{\text{sc}}(\Sigma)$.

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

4.4 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 $\eta$ of basic transition identities and a corresponding set of basic labelled trees with a single node labelled with an element of $\eta$. All the transitions in figure 7 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 (see figure 8 for an example):

- $v \triangleleft T$, 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 $v$.
- $v \triangleleft \{t\}$ denotes the tree $v \triangleleft \{t\}$, and $v \triangleright T$ denotes the set of trees $\{v \triangleleft t \mid t \in T\}$.

A similar, though slightly more complex, naming discipline could be used for the places of the constructed nets following the scheme used in [5]. However, since place trees were essentially needed for the definition of recursion, which is not considered in this paper, we will not use them here. Instead, 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 the
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.

**Binary operators.** Let \( \Omega_{\text{bin}} \in \{\Omega_{\text{□}}, \Omega_{\text{■}}, \Omega_{\text{●}}, \Omega_{\text{□}}\} \) be a binary ABC operator box
and \( \Sigma = (\Sigma_1, \Sigma_2) = (\Sigma_{v_1}, \Sigma_{v_2}) \) be a pair of boxes. The result of a simultaneous
substitution of the transitions \( v_{i1}^{\text{bin}} \) and \( v_{i2}^{\text{bin}} \) in \( \Omega_{\text{bin}} \) by the boxes \( \Sigma_1 \) and \( \Sigma_2 \) is a
labelled net \( \Omega_{\text{bin}}(\Sigma) = \Phi \) whose components are defined as follows.

The set of transitions of \( \Phi \) is the set of all trees \( v_{i1}^{\text{bin}} \prec t \) (with \( t \in T_{\Sigma} \) and
\( i = 1, 2 \)). The label of each \( v_{i}^{\text{bin}} \prec t \) is that of \( t \). Each \( i \)-labelled or \( b \)-labelled
place \( p \in S_{\Sigma_i} \) belongs to \( S_{\Phi} \). Its label and marking are unchanged and for every
transition \( w \prec t \), the weight function is given by:

\[
W_{\Phi}(p, w \prec t) = \begin{cases} 
W_{\Sigma_i}(p, t) & \text{if } w = v_{i1}^{\text{bin}} \\
0 & \text{otherwise},
\end{cases}
\]

and similarly for \( W_{\Phi}(w \prec t, p) \).

For every place \( s \in S_{\Phi} \) with \( s = \{u_1, \ldots, u_k\} \) and \( s^* = \{w_1, \ldots, w_m\} \), for
\( k, m \in \{0, 1, 2\} \), we construct in \( S_{\Phi} \) all the places of the form

\[
p \equiv (x_1, \ldots, x_k, e_1, \ldots, e_m),
\]

where each \( x_i \) (if any) is an exit place of \( \Sigma_{u_i} \), and each \( e_j \) (if any) is an entry
place of \( \Sigma_{e_j} \) (Notice that \( p \) is constructed exactly once in \( S_{\Phi} \) because, in the
binary operator boxes of ABC, each transition has exactly one input place
and exactly one output place.) 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 \( w \prec t \), the weight function is given by:

\[
W_{\Phi}(p, w \prec t) \equiv \begin{cases} 
W_{\Sigma_u}(x_i, t) + W_{\Sigma_e}(e_j, t) & \text{if } w \in s \cap s^* \text{ and } w = u_i = w_j \\
W_{\Sigma_u}(x_i, t) & \text{if } w \in s \setminus s^* \text{ and } w = u_i \\
W_{\Sigma_e}(e_j, t) & \text{if } w \in s^* \setminus s \text{ and } w = w_j \\
0 & \text{otherwise},
\end{cases}
\]

and similarly for \( W_{\Omega}(\Sigma)(w \prec t, p) \).

For every \( b \in B \), there is a unique \( \delta \)-labelled place \( p^b \equiv (p_{v1}^b, p_{v2}^b) \in S_{\Omega(\Sigma)} \),
where \( p_{v1}^b \) and \( p_{v2}^b \) are the unique \( \delta \)-labelled places of respectively \( \Sigma_1 \) and \( \Sigma_2 \).
The marking of \( p^b \) is the sum of the markings of \( p_{v1}^b \) and \( p_{v2}^b \), and for each
transition \( w \prec t \), the weight function is given by:

\[
W_{\Omega(\Sigma)}(\Sigma)(w \prec t) \equiv W_{\Sigma_w}(p^b_w, t),
\]

and similarly for \( W_{\Omega(\Sigma)}(w \prec t, p^b) \).
Scoping. An application of the scoping operator $\Omega_{\text{sc}}$ to a box $\Sigma$ results in a labelled net which is like $\Sigma$ with the only difference that the set of transitions comprises all trees $t \triangleq v^{\text{sc}} a \triangleq \{t_1, t_2\}$ with $t_1, t_2 \in T_\Sigma$ such that $\{\lambda_\Sigma(t_1), \lambda_\Sigma(t_2)\} = \{a, \widehat{a}\}$, as well as all trees $t' \triangleq v^{\text{sc}} a \triangleq t_3$ with $t_3 \in T_\Sigma$ such that $\lambda_\Sigma(t_3) \notin \{a, \widehat{a}\}$. The label of $t$ is $\tau$, and that of $t'$ is $\lambda(t_3)$. The weight function, for every place $p$, is given by:

$$W_{\Omega_{\text{sc}}(\Sigma)}(p, t) \triangleq W_\Sigma(p, t_1) + W_\Sigma(p, t_2) \quad \text{and} \quad W_{\Omega_{\text{sc}}(\Sigma)}(p, t') \triangleq W_\Sigma(p, t_3),$$

and similarly for $W_{\Omega_{\text{sc}}(\Sigma)}(t, p)$ and $W_{\Omega_{\text{sc}}(\Sigma)}(t', p)$.

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

4.5 Consistency in the box domain

The net substitution meta-operator is illustrated in figure 8, where explicit transition identities are shown for various stages of the construction, from the basic net and transition identities shown in figure 7.

**Fig. 8.** The trees in the bottom row give the identities of the transitions (from left to right) of the boxes shown in the upper row. In the linear notation the fourth tree is $v_1^\square \triangleq v^{\text{sc}} a \triangleq \{v_1^1 \triangleq v^{ab}^+, v_2 \triangleq v^{ab}^-\}$.

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 \( \Sigma_1, \Sigma_2 \) and \( \Sigma \) be any boxes.

1. If \( \Omega \) is a binary ABC operator box, then \( M_{\Omega(\Sigma)}^{\text{str}} = \varnothing \) iff \( M_{\Sigma_1}^{\text{str}} = \varnothing = M_{\Sigma_2}^{\text{str}} \).
2. If \( \Sigma' \) is \( \Sigma \text{ sc}\ a \) or \( \Sigma \text{ tie}\ b \) or \( \Sigma.B \), then \( M_{\Sigma'}^{\text{str}} = \varnothing \) iff \( M_{\Sigma}^{\text{str}} = \varnothing \).

Moreover, the operation of net substitution always returns a syntactically valid object provided that it is applied to operands belonging to the correct domain.

Theorem 4. Let \( \Omega \) be any binary ABC operator box and \( \Sigma \in \text{dom}_\Omega \). Then, \( \Omega(\Sigma) \) is a box with a clean and ac-free marking. Moreover, if all the dynamic boxes (if any) in \( \Sigma \) have quasi-safe markings, then the marking of \( \Omega(\Sigma) \) is also quasi-safe.

Theorem 5. Let \( \Sigma \) be a static or dynamic box, \( a \in A, b \in B \) and \( B \in \text{mult}(B) \). Then, \( \Sigma \text{ sc}\ a, \Sigma \text{ tie}\ b \) and \( \Sigma.B \) are boxes with clean and ac-free markings. Moreover, if \( \Sigma \) has a quasi-safe marking, then the markings of these boxes are also quasi-safe.

We finally observe that, if one makes no use of buffer stuffing nor buffer restriction nor the basic nets \( \Sigma_{ab^+}, \Sigma_{ab^-} \) and \( \Sigma_{ab^\pm} \), then the net operations described above are similar to those defined in the standard box algebra (see [3–5]), except for the additional \( b \)-labelled places, which are all isolated and unmarked.

5 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.

5.1 Static properties of boxes

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

Proposition 6. Let \( \Omega \) be a binary ABC operator, \( \Sigma \) and \( \Theta \) be static boxes, \( \Phi \) be a static or dynamic box, \( a \in A, b \in B \) and \( B' \in \text{mult}(B) \).

1. \( \Sigma \text{ sc}\ \Theta = \Sigma \text{ sc}\ \Theta \) and \( \Sigma \text{ sc}\ \Theta = \Sigma \text{ sc}\ \Theta \).
2. \( \Sigma \text{ tie}\ \Theta = \Sigma \text{ tie}\ \Theta \) and \( \Sigma \text{ tie}\ \Theta = \Sigma \text{ tie}\ \Theta \).
3. \( \Sigma; \Theta = \Sigma; \Theta \) and \( \Sigma; \Theta = \Sigma; \Theta \).
4. \( \Sigma || \Theta = \Sigma || \Theta \) and \( \Sigma || \Theta = \Sigma || \Theta \).
5. \( \Sigma \text{ sc}\ a = \Sigma \text{ sc}\ a \) and \( \Sigma \text{ sc}\ a = \Sigma \text{ sc}\ a \), and \( \Sigma \text{ tie}\ b = \Sigma \text{ tie}\ b \) and \( \Sigma \text{ tie}\ b = \Sigma \text{ tie}\ b \).
6. \( (\Phi \text{ sc}\ a).B = (\Phi.B) \text{ sc}\ a \), and \( (\Phi \text{ tie}\ b).B = (\Phi.B) \text{ tie}\ b \) provided that \( b \notin B \).
7. If \( \Sigma = (\Sigma_1, \Sigma_2) \in \text{dom}_\Omega \), then \( \Omega(\Sigma_1.B, \Sigma_2.B') = \Omega(\Sigma).B + B' \).
5.2 Structural equivalence

We intend here to capture situations where different applications of a same operator box lead to the same labelled net. We start by defining four auxiliary relations. For \( \Sigma \equiv (\Sigma_1, \Sigma_2) \) and \( \Theta \equiv (\Theta_1, \Theta_2) \), two pairs of boxes, we define:

- \( \Sigma \equiv_0 \Theta \) if there is a box \( \Psi \) such that \( \{ \Sigma_1, \Theta_1 \} = \{ \overline{\Psi}, \Psi \} \) and \( \Sigma_2 = \Theta_2 \). This relation is intended for the iteration operator in an initial marking of its first component, which can also be obtained from the final marking of the same component.
- \( \Sigma \equiv_1 \Theta \) if there are boxes \( \Psi_1 \) and \( \Psi_2 \) such that \( \{ \Sigma, \Theta \} = \{ \{ \Psi_1, \Psi_2 \}, (\Psi_1, \overline{\Psi_2}) \} \).
  This relation is intended for the choice or iteration operator in an initial marking, which can be obtained from an initial marking of the first component, or the second one.
- \( \Sigma \equiv_2 \Theta \) if there are boxes \( \Psi_1 \) and \( \Psi_2 \) such that \( \{ \Sigma, \Theta \} = \{ (\Psi_1, \Psi_2), (\Psi_1, \overline{\Psi_2}) \} \).
  This relation, involving final markings, is similar to the previous one but intended for choice only.
- \( \Sigma \equiv_3 \Theta \) if there are boxes \( \Psi_1 \) and \( \Psi_2 \) and \( B_1, B_2, B_1', B_2' \in \text{mult}(B) \) such that \( B_1 + B_2 = B_1' + B_2' \), \( \Sigma = \{ \Psi_1, B_1, \Psi_2, B_2 \} \) and \( \Theta = \{ \Psi_1, B_1', \Psi_2, B_2' \} \). In other words, \( \Sigma \) and \( \Theta \) are the same except perhaps the distribution of tokens in open buffer places\(^5\) corresponding to the same \( b \) but coming from different components. This relation is intended for each binary operator as it always merges together the corresponding open buffer, including their markings.

We then define six binary relations on boxes, one for each of the operator boxes of ABC, as follows:\(^6\)

\[
\begin{align*}
\equiv_{\text{id}} & \equiv \equiv_4 \\
\equiv_{\text{id}_{\text{ab}}} & \equiv \equiv_4 \circ (\text{id}_{\text{ab}} \cup \equiv_0 \cup \equiv_2) \\
\equiv_{\text{id}_{\text{ax}}} & \equiv \equiv_4 \circ (\text{id}_{\text{ab}} \cup \equiv_0 \cup \equiv_1 \cup \equiv_3) \\
\equiv_{\text{id}_{\text{ex}}} & \equiv \equiv_4 \circ (\text{id}_{\text{ab}} \cup \equiv_3)
\end{align*}
\]

where \( \text{id}_{\text{ab}} \) is the identity relation on \text{abox}. The relations \( \equiv_{\text{id}} \) are reflexive and symmetric, but they are not in general transitive since, for example,

\[
(\Sigma_1, \Sigma_2) \equiv_{\text{id}_{\text{ax}}} (\overline{\Sigma_1}, \overline{\Sigma_2}) \equiv \equiv_{\text{id}_{\text{ex}}} (\Sigma_1, \Sigma_2) \not\equiv \equiv_{\text{id}_{\text{ex}}} (\overline{\Sigma_1}, \overline{\Sigma_2}).
\]

However, when restricted to the application domain of the corresponding operator box, they become transitive and identify the tuples of boxes which give rise to the same composite nets.

\(^5\) This distribution only concerns open buffer places since buffer stuffing never changes the marking of closed buffer places.

\(^6\) The composition of two binary relations \( R, R' \) on a set \( X \) is defined as \( R \circ R' = \{(x, y) \mid \exists z \in X : (x, z) \in R \land (z, y) \in R'\} \).

Proposition 7. Let $\Omega$ be an ABC operator box, $\Sigma \in \text{dom}_\Omega$, and $\Theta$ be an $\Omega$-tuple of boxes.

1. If $\Sigma \equiv_\Omega \Theta$, then $\Theta \in \text{dom}_\Omega$ and $[\Sigma] = [\Theta]$.
2. $\equiv_\Omega$ is an equivalence relation on $\text{dom}_\Omega$.
3. If $[\Sigma] = [\Theta]$, then $\Omega(\Sigma) = \Omega(\Theta)$ iff $\Sigma \equiv_\Omega \Theta$.

That the precondition $[\Sigma] = [\Theta]$ cannot be dropped in the last property is justified by the following counter-example: $\Sigma_{ab^+} \neq \Omega_{a^+} \Sigma_{ab^-}$ and $\Sigma_{ab^+} \Sigma a = \Sigma_{ab^-} \Sigma a$ (no transition is left in the nets by the scoping operation).

5.3 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 $\Omega(\Sigma)$, knowing the steps enabled by the boxes in $\Sigma$.

Proposition 8. Let $\Omega_{\text{bin}} \in \{\Omega_{\text{g}}, \Omega_{\text{d}}, \Omega_{\text{b}}, \Omega_{\text{l}}\}$ and $\Sigma \in \text{dom}_\Omega$. Then enabled($\Omega_{\text{bin}}(\Sigma)$) comprises exactly all sets of transitions

$$U = (v_1^{\text{bin}} \bullet U_1) \cup (v_2^{\text{bin}} \bullet U_2)$$

such that there is a pair of boxes $\Theta$ satisfying $\Theta \equiv_{\text{bin}} \Sigma$ and $U_i \in \text{enabled}(\Theta_i)$, for $i \in \{1, 2\}$. Moreover, $\Omega_{\text{bin}}(\Sigma[U]) \equiv_{\text{bin}} \Omega_{\text{bin}}(\Phi)$, where $\Theta_i[U_i) \Phi_i$, for $i \in \{1, 2\}$.

Proposition 9. Let $\Sigma$ be a box, $a \in \Lambda$, $b \in \mathbb{B}$, $B \in \text{mult}(\{b\})$ and $B' \in \text{mult}(B)$.

1. enabled($\Sigma \Theta a$) comprises exactly all sets of transitions

$$U = (v^{\Theta a} \bullet U_1) \cup \{v^{\Theta a} \bullet \{v_1, w_1\}, \ldots, v^{\Theta a} \bullet \{v_k, w_k\}\}$$

such that $Z \in V \cup W \in \text{enabled}(\Sigma)$, where the steps $Z$, $V \not= \{v_1, \ldots, v_k\}$ and $W \not= \{w_1, \ldots, w_k\}$ satisfy $a, a \not= \lambda_{\Sigma}(Z)$, and for all $i \in \{1, \ldots, k\}$, $\lambda_{\Sigma}(v_i) = \{a\}$ and $\lambda_{\Sigma}(w_i) = \{a\}$.

Moreover, $\Sigma \Theta a[U] \not= \Sigma \Theta a$ (because $\Sigma[V \cup W \cup Z] \Phi$).

2. enabled($\Sigma \text{tie} b$) comprises exactly all sets of transitions $U = v^{\text{tie} b} \bullet V$ such that $V \in \text{enabled}(\Sigma)$. Moreover, $\Sigma \text{tie} b[U] \not= \Sigma \text{tie} b$, where $\Sigma[V] \Phi$.

3. enabled((\Sigma \text{tie} b).B) comprises exactly all sets $U \in \text{enabled}(\Sigma \text{tie} b)$.

Moreover, $(\Sigma \text{tie} b).B[U] \not= \Sigma \text{tie} b[B][U] \Phi$. Moreover, if $\Sigma[U] \Phi$ then $\Sigma[B'][U] \Phi B'$.

4. enabled($\Sigma$) is a subset of enabled($\Sigma.B'$).

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

The behaviours of the basic asynchronous boxes of ABC are captured below.
Proposition 10. Let \( B \in \text{mult}(\mathbb{B}) \) and \( \Sigma = \sum_{\alpha} B \), where \( \Sigma_{\alpha} \) is one of the basic boxes in figure 7.

1. For \( \alpha \in \{a, ab^+\} \), the non-empty steps of \( \Sigma \) are respectively
   \[
   \Sigma[\{v^a\}] \sum_{\alpha} B \quad \text{and} \quad \Sigma[\{v^{ab^+}\}] \sum_{ab^+} B .
   \]

2. For \( \alpha \in \{ab^-, ab^\} \), if \( b \notin B \) then \( \Sigma \) has no non-empty step; otherwise the non-empty steps are respectively
   \[
   \Sigma[\{v^{ab^-}\}] \sum_{ab^-} (B - \{b\}) \quad \text{and} \quad \Sigma[\{v^{ab}\}] \sum_{ab^\} B .
   \]

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

Proposition 11. Let \( \Omega \) be an operator box of ABC and \( \Sigma \in \text{dom}_\Omega \). Then every net derivable from \( \Omega(\Sigma) \) is of the form \( \Omega(\Theta) \), where \( \Theta \in \text{dom}_\Omega \) and \( \Theta[\Theta] = \Sigma \). Moreover, if no box in \( \Sigma \) is dynamic, then every net derivable from \( \Omega(\Sigma) \) or \( \Omega(\Sigma) \) is of the form \( \Omega(\Theta) \), where \( \Theta \in \text{dom}_\Omega \) and \( \Theta[\Theta] = \Sigma \).

Theorem 12. Every composite net of ABC is a quasi-safe static or dynamic box. Moreover, it is static if the marking operators \( \cdot \) are not used, unless in the scope of the \( \cdot \) or \( \cdot \) operators.

6 An algebra of asynchronous box expressions

We consider an algebra of process expressions over the signature:

\[
\mathcal{A} \cup \{\cdot\} \cup \{\cdot\} \cup \{\cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot\} \cup \{\text{sc } a \mid a \in \mathbb{A}\} \cup \{\text{tie } b \mid b \in \mathbb{B}\} \cup \{\cdot \mid b \in \mathbb{B}\} ,
\]

where \( \mathcal{A} \) are the constants; the binary operators \( \cdot, \cdot, \cdot, \cdot, \cdot, \cdot \) and \( \cdot \) will be used in the infix mode; the unary operators \( \text{sc } a, \text{tie } b \) and \( \cdot \) will be used in the postfix mode; and \( \cdot \) and \( \cdot \) 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 \texttt{aexp\texttt{stc}} and \texttt{aexp\texttt{dyn}}. Collectively, we will refer to them as the (asynchronous) box expressions, \texttt{aexp}. Their syntax is given by:

\[
\begin{align*}
\text{aexp\texttt{stc}} \quad E & ::= \alpha \mid E \text{sc } a \mid E \text{tie } b \mid E . b \mid E \mid E \mid E \ow E \mid E \ot E \\
\text{aexp\texttt{dyn}} \quad D & ::= E \mid E \mid D \text{sc } a \mid D \text{tie } b \mid D . b \mid D \mid D \ow D \mid D \ow D \mid D \ow D \mid D \ow D \mid D \ow D
\end{align*}
\]

where \( \alpha \in \mathcal{A}, a \in \mathcal{A} \) and \( b \in \mathbb{B} \). Moreover, we will use \( F \) to denote any static or dynamic expression.\(^7\)

Note that, w.r.t. the original PBC and PNA algebras, the above syntax uses slightly different symbols to denote scoping (\( E \text{ sc } a \) instead of \( [a : E] \)) and iteration (\( E \ow E' \) instead of \( E * E' \)). However, their meaning remains unchanged.
We also use the notations \([F]\) and \(\overline{F}\) yielding static expressions, where \([F]\) is \(F\) with all occurrences of \(\langle\rangle\) and \((\cdot)\) removed, and \(\overline{F}\) is \([F]\) with all occurrences of \(\cdot\) 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, an asynchronous box expression encodes the structure of a box net, together with the current marking of the control places (using overbars and underbars) and of the buffer places (using the \(\cdot\)'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, \(\overline{E}\) represents \(E\) in its final state. Note that the \(\cdot\) notation is needed for static as well as for dynamic box expressions because the dormant part of a dynamic box expression may still have \(\cdot\)'s which are later needed in the active part. For instance, \(D \equiv ab^+; b; \overline{\overline{b}}\) has a static component with a \(\cdot\) in it, and may be transformed into an equivalent \(D' \equiv ab^+; \overline{\overline{b}}\).

### 6.1 Denotational semantics

The denotational semantics of box expressions is given in the form of a mapping box : aexpr \(\rightarrow\) abox, defined homomorphically by induction on their structure, following the syntax (2). Below, \(\alpha \in \mathcal{A}\), \(b \in \mathbb{B}\), una stands for any unary operator \((\text{sc} a, \text{tie} b \text{ or } \cdot)\), and bin for any binary operator \(([], \sqcap, \; \text{or } \oplus)\).

\[
\begin{align*}
\text{box}(\alpha) & \equiv \Sigma_{\alpha} \quad \text{box}(\overline{E}) \equiv \text{box}(E) \quad \text{box}(E) \equiv \text{box}(\overline{E}) \\
\text{box}(F\text{ una}) & \equiv \text{box}(F)\text{ una} \quad \text{box}(F_1\text{ bin } F_2) \equiv \text{box}(F_1)\text{ bin }\text{box}(F_2).
\end{align*}
\]

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 13.** Let \(F\) be a box expression.

1. \(\text{box}(F)\) is a static or dynamic box.
2. \(\text{box}(F)\) is a static box iff \(F\) is a static box expression.

### 6.2 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. Using these rules, one can derive \((a \sqcap b)\|\{d; e\} \equiv (\overline{a} \sqcap b)\|\{\overline{d}; e\})\), as in figure 9.

The rules in table 1 either directly follow those of the PNA model or capture the fact that an asynchronous message, produced by the \(ab^+\) expression and represented by \(\cdot\), can freely move within a box expression in order to be received by some action of the form \(cb^-\) (see rules \(B_1\), \(\overline{C}_1\) and \(\overline{Q}_{R}\)). However, the \(\cdot\) may never cross the boundary imposed by the tie \(b\) operator (notice that the rule \(B_1\) excludes una from being tie \(b\), but not from being tie \(b\)', provided that \(b \neq b'\); the
\[
\begin{array}{c}
(a \square b) || (d; e) \equiv (\pi \square b) || (\bar{d}; e) \\
(a \square b) || (d; e) \equiv (a \square b) || (d; e) \\
\end{array}
\]

**Fig. 9.** The derivation tree of \((a \square b) || (d; e) \equiv (\pi \square b) || (\bar{d}; e)\).

same rule, with \(\eta a = b'\), also implies that \(F, b, b' \equiv F, b', b\), i.e., the commutativity of the marking operator \(b\).

It may be observed that, due to the rules \texttt{CON-2}, \texttt{ENT} and \texttt{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 14.** Let \(F_1\) and \(F_2\) be box expressions.

1. If \(F_1 \equiv F_2\) then \([F_1] \equiv [F_2]\), \([F_1] = [F_2]\) and \(\text{box}(F_1) = \text{box}(F_2)\).
2. If \([F_1] = [F_2]\), then \(\text{box}(F_1) = \text{box}(F_2)\) 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 \(F_1 \equiv a \triangleleft c \triangleleft a\) and \(F_2 \equiv a \triangleleft c \triangleleft a\) for which \(F_1 \not\equiv F_2\) but \(\text{box}(F_1) = \text{box}(F_2)\) (no transition is left in the nets by the scoping operation).

**Theorem 15.** Let \(F\) be a box expression.

1. \(\text{box}(F) = |\text{box}(F)|\) iff \(F \equiv [F]\).
2. \(\text{box}(F) = |\text{box}(F)|\) iff \(F \equiv [F]\).

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.
### 6.3 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 $F \xrightarrow{U} F'$ such that $F$ and $F'$ are box expressions and $U \in \mathbb{U} \equiv \text{mult}(\mathbb{T})$. The idea here is that $U$ is a valid step for the boxes associated with $F$ and $F'$, *i.e.*, that $\text{box}(F) \cup \text{box}(F')$, as demonstrated by theorem 17.

Formally, we define a ternary relation $\to$ which is the least relation comprising all $(F, U, F') \in \text{axp} \times \mathbb{U} \times \text{axp}$ such that the relations in table 3 hold. Notice that we use $F \xrightarrow{U} F'$ to denote $(F, U, F') \in \to$. In the definition of $\to$ we make no restriction on $U_1$ and $U_2$ but the domain of application of $\text{bin}$ 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.

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

---

**Table 1.** Structural similarity relation for ABC. where $b \in \mathbb{B}$, una stands for any unary ABC operator and bin stands for any binary ABC operator.
\[
s; ((pb^+ \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b)
\]

\[
\equiv s; ((pb^+ \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b)
\]

\[
\begin{align*}
\{t_1\} & \quad \Rightarrow s; ((pb^+ \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b) \\
& \quad \equiv s; ((pb^+ \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b) \\
& \quad \equiv s; ((pb^+ \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b)
\end{align*}
\]

\[
\begin{align*}
\{t_2, t_3\} & \quad \Rightarrow s; ((pb^+ \oplus b \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b) \\
& \quad \equiv s; ((pb^+ \oplus b \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b) \\
& \quad \equiv s; ((pb^+ \oplus b \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b)
\end{align*}
\]

\[
\begin{align*}
\{t_4, t_5, t_6, t_7\} & \quad \Rightarrow s; ((pb^+ \oplus b \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b) \\
& \quad \equiv s; ((pb^+ \oplus b \oplus f) ((pb^+ \oplus f) ((cb^- \oplus f))) \text{tie } b)
\end{align*}
\]

Table 2. Execution scenario I and part of the derivation tree for the move labelled by \(\{t_2, t_4\}\). The identities of various transitions are as follows: \(t_1 = \nu_1 \circ \nu^*, t_2 = \nu_2 \circ \nu_{1a} \circ \nu_{1b} \circ \nu_{1c} \circ \nu_{1d} \circ \nu_{1e} \circ \nu_{1f} \circ \nu^* \), \(t_3 = \nu_2 \circ \nu_{2a} \circ \nu_{2b} \circ \nu_{2c} \circ \nu_{2d} \circ \nu_{2e} \circ \nu_{2f} \circ \nu^* \), \(t_4 = \nu_2 \circ \nu_{4a} \circ \nu_{4b} \circ \nu_{4c} \circ \nu_{4d} \circ \nu_{4e} \circ \nu_{4f} \circ \nu^* \), \(t_5 = \nu_2 \circ \nu_{5a} \circ \nu_{5b} \circ \nu_{5c} \circ \nu_{5d} \circ \nu_{5e} \circ \nu_{5f} \circ \nu^* \), \(t_6 = \nu_2 \circ \nu_{6a} \circ \nu_{6b} \circ \nu_{6c} \circ \nu_{6d} \circ \nu_{6e} \circ \nu_{6f} \circ \nu^* \) and \(t_7 = \nu_2 \circ \nu_{7a} \circ \nu_{7b} \circ \nu_{7c} \circ \nu_{7d} \circ \nu_{7e} \circ \nu_{7f} \circ \nu^* \).
<table>
<thead>
<tr>
<th>Table 3. Transition based operational semantics for ABC, where $a \in \mathcal{A}$, except in \textsc{esc}, where $a \in \mathcal{A}$, $b \in \mathcal{B}$ and \text{bin} stands for any binary ABC operator.</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textsc{ea1}</td>
</tr>
<tr>
<td>\textsc{ea2}</td>
</tr>
<tr>
<td>\textsc{ea3}</td>
</tr>
<tr>
<td>\textsc{ea4}</td>
</tr>
<tr>
<td>\textsc{eq1}</td>
</tr>
<tr>
<td>\textsc{eq2}</td>
</tr>
<tr>
<td>\textsc{ebuf}</td>
</tr>
<tr>
<td>\textsc{etie}</td>
</tr>
<tr>
<td>\textsc{esc}</td>
</tr>
<tr>
<td>\textsc{op}</td>
</tr>
<tr>
<td>\text{tie}</td>
</tr>
</tbody>
</table>

\textbf{Proposition 16.} Let $F$ and $F'$ be two box expressions. Then, $F \xrightarrow{U} F'$ 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.

\textbf{Theorem 17.} Let $F$ be a box expression.

1. If $F \xrightarrow{U} F'$, then $F'$ is a box expression such that $\text{box}(F)[U] \text{box}(F')$ and $[F] = [F']$.

2. If $\text{box}(F)[U] \Sigma$, then there is a box expression $F'$ such that $\text{box}(F') = \Sigma$ and $F \xrightarrow{U} F'$.

Figure 2 shows the derivation of the step sequence representing scenario I in section 2, together with a part of the derivation tree for the move labelled by $(t_1, t_4)$ (see also figure 6).
6.4 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 17, 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 \( \{D\} \) to denote all the box expressions derivable from \( D \), i.e., the least set of expressions containing \( D \) such that if \( D' \in \{D\} \) and \( D' \xrightarrow{U} D'' \), for some \( U \in \U \), then \( D'' \in \{D\} \). Moreover, \( \{D\}_\equiv \) will denote the equivalence class of \( \equiv \) containing \( D \). The full transition system of \( D \) is \( \text{fts}_D \equiv (V, L, A, \text{init}) \), where \( V \equiv \{\{D'\}_\equiv \mid D' \in \{D\}\} \) is the set of states; \( L \equiv \U \) is the set of arc labels; \( A \equiv \{([D']_\equiv, U, [D'']_\equiv) \in V \times \U \times V \mid D' \xrightarrow{U} D'' \} \) is the set of arcs; and \( \text{init} \equiv [D]_\equiv \) is the initial state. For a static box expression \( E \), \( \text{fts}_E \equiv \text{fts}_{\text{box}(E)} \).

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 \( D \xrightarrow{\Sigma} D' \) for two different expressions \( D \) and \( D' \), whereas in the domain of boxes, \( \Sigma \emptyset \Theta \) 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 18.** For every box expression \( F \),

\[
\text{iso}_F \equiv \{([F']_\equiv, \text{box}(F')) \mid [F']_\equiv \text{ is a node of } \text{fts}_F\}
\]

is an isomorphism between the full transition systems \( \text{fts}_F \) and \( \text{fts}_{\text{box}(F)} \). Moreover, \( \text{iso}_F \) preserves the property of being in an initial or final state\(^8\).

6.5 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 \( \Gamma \) is a finite multiset in \( \mathbb{L} \equiv \text{mult}(\mathcal{A}_r) \), 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 19.** Let \( F \) be a box expression and \( \Gamma \in \mathbb{L} \). Then \( F \xrightarrow{\Gamma} F' \) iff there is \( U \in \U \) such that \( F \xrightarrow{U} F' \) and \( \lambda(U) = \Gamma \).

\(^8\) In terms of box expression (cf. theorem 15) or box net (cf. section 3.5), not w.r.t. the transition system.
The results concerning transition based operational semantics directly extend to the label based one. Let $F$ be a box expression. In view of proposition 19, the label based operational semantics of $F$ is faithfully captured by the *labelled transition system* of $F$, denoted by $lts_F$, and defined as $lts_F$ with each arc label $U$ changed to $\lambda(U)$. The consistency result for the label based operational semantics can then be formulated thus.

**Theorem 20.** For every box expression $F$,

$$iso_F \triangleq \{(F')_{[F]} \mid \text{box}(F') \} \text{ is a node of } lts_F\}$$

is an isomorphism between the labelled transition systems $lts_F$ and $lts_{box(F)}$. Moreover, $iso_F$ preserves the property of being in an initial or final state.

The rules of the label based operational semantics are put into work in tables 5 and 6, where we use the second and third scenarios introduced in section 2.

To further illustrate how the buffer restriction operator controls the way in which tokens in the buffer places may be used, let us consider the sixth line of the scenario II shown in table 5:

$$(pb^+;:((pb^T \oplus f)(cb^- \oplus f)) \text{ tie } b; cb^- \cdot b)) \text{ tie } b .$$

Notice that if we were able to move the $b$ from the end of the expression to under the bar of $cb^-$, then we would have been able to execute $cb^-$. However, the $B_1$ rule has been designed so that it is impossible to move $b$ inside the scope.

---

Table 4. Label based operational semantics for ABC, where $a \in A_r$ (except in $LSC$, where $a \in A$), $b \in B$ and bin stands for any binary ABC operator.

---

$\begin{array}{llll}
LA_1 \quad & \pi \overset{(a)}{\rightarrow} a & \quad & LA_2 \quad & ab^+ \overset{(a)}{\rightarrow} ab^+ b \\
LA_3 \quad & ab^- b \overset{(a)}{\rightarrow} ab^- & \quad & LA_4 \quad & ab^\pm b \overset{(a)}{\rightarrow} ab^\pm b \\
LQ_1 \quad & F \overset{a}{\Rightarrow} F & \quad & LQ_2 \quad & F \equiv F', F' \overset{a'}{\Rightarrow} F''', F''' \equiv F'''' \\
LB_{UF} \quad & F \overset{a}{\Rightarrow} F' \\
& F.b \overset{a}{\Rightarrow} F'.b & \quad & LTHE \quad & F \overset{a}{\Rightarrow} F' \\
& & & & F \text{ tie } b \overset{a}{\Rightarrow} F' \text{ tie } b \\
LQ_{OP} \quad & F_1 \overset{a_1}{\Rightarrow} F'_1, F_2 \overset{a_2}{\Rightarrow} F'_2, F_3 \overset{a_3}{\Rightarrow} F'_3 \\
& F_1 \text{ bin } F_2 \overset{a_1+a_2}{\Rightarrow} F'_1 \text{ bin } F'_2 & \quad & LSC \quad & D \overset{a}{\Rightarrow} D', a \notin \Gamma, k \in \mathbb{N} \\
& & & & D \text{ sc } a \overset{a+k}{\Rightarrow} D' \text{ sc } a \\
\end{array}$
\[
(\text{tie}_b;((\text{tie}_b \cdot f);((\text{tie}_b \cdot f);((\text{tie}_b \cdot f);((\text{tie}_b \cdot f)) \text{tie}_b;\text{tie}_b)) \text{tie}_b)
\]

Table 5. Execution scenario II

of the internal tie\_b operator. This is fully consistent with the net semantics of the corresponding box shown in the middle of figure 5 which, after executing the topmost p-labelled transition cannot execute the c-labelled transition whose input place is the internal b-labelled place.

7 Conclusions

In this paper, we proposed a framework which supports two consistent (in a very strong sense, since the corresponding transition systems are isomorphic and not only bisimilar) concurrent semantics for a class of process expressions with both synchronous and asynchronous communication. The artificial constraints introduced in previous models to stay in a safe net framework are no longer necessary. Such a model can be used, in particular, to give the semantics of a programming language with timing constraints and exceptions.

In our future work, we intend to treat recursion, i.e., process definitions of the form \(X \equiv E\), where \(X\) is allowed to occur (directly or indirectly) in \(E\), as well as other extensions. For the latter, it is worth noting that the results obtained in appendix B for more general kinds of operators than those considered in the main body of this paper, already make it possible to introduce other iteration constructs (e.g., one with a loop in the middle of a ternary operator), and interface transformations (e.g., multiway synchronisations).
Asynchronous Box Calculus

\[
(pb^+ f) ((cb^- f)) ((tb^- f)) \quad \text{PAR1, HIT1}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+ b \circ f) ((cb^- f)) ((tb^- f)) \quad \text{LA2, LOP}}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+_b \circ f) ((cb^- f)) ((tb^- f)) \quad \text{IT2, E1}}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+_b b \circ f) ((cb^- f)) ((tb^- f)) \quad \text{LA2, LOP}}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+_b \circ f) ((cb^- b \circ f)) ((tb^- f)) \quad \text{OPL-R, HIT2, E1}}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+ f) ((cb^- f)) ((tb^- b \circ f)) \quad \text{LA2-4, LOP}}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+_b \circ f) ((cb^- f)) ((tb^- b \circ f)) \quad \text{IT2, HIT2, OPL-R, E1}}
\]

\[
\overset{<p>}{\overset{<p>}{(pb^+_b \circ f) ((cb^- f)) ((tb^- b \circ f)) \quad \text{LA4, LOP}}
\]

Table 6. Execution scenario III.

8 Acknowledgements

This research was partially supported by the ARC JIP and EPSRC BEACON projects.

References


A Proofs of the results from the main body of the paper

To start with, propositions 1, 2, 9, 10 and 19 follow directly from the definitions. (In particular, proposition 1 follows from the control-restrictedness of any box and the ac-freeness of any dynamic box, while proposition 2(2), from the cleanliness of dynamic boxes.) For the remaining results, the proofs are as follows (many results follow from more general ones given in appendix B).

Proof of proposition 3: Follows from proposition 21.
Proofs of theorems 4 and 5: Follow from theorem 24 and the definitions.

Proof of proposition 6: Follows from proposition 25, proposition 26(2) and the definitions.

Proof of proposition 7: Follows from proposition 26.

Proof of proposition 8: For \( \Omega \) the result follows from the definitions, and \( \equiv \) is needed in order to rearrange tokens on the open buffer places in \( \Sigma_1 \) and \( \Sigma_2 \), to ensure that the steps \( U_1 \) and \( U_2 \) are enabled separately. For the other \( \Omega \)'s, the result follows from proposition 27.

Proof of proposition 11: The first part follows from propositions 8 and 9(1,2). The second part follows from the first one and proposition 6.

Proof of theorem 12: Follows from theorem 4, propositions 2 and 11, and the fact that the asynchronous boxes in figure 7 are all static. The quasi-safety results also from theorem 4, since all the clean markings of the asynchronous boxes in figure 7 are quasi-safe (they are exactly the initial and final markings).

Proof of theorem 13: Follows from theorem 12, by induction on the structure of \( F \).

Proofs of theorems 14 and 15: Follow from the definitions (in particular, table 1 and the translation rules (3)) and propositions 2, 6 and 7, by induction on the structure of \( F \).

Proof of proposition 16: \((\Longrightarrow)\) Consider the derivation tree \( T \) for \( F \xrightarrow{\sigma} F' \). Since no rule can ever produce an empty step from a non-empty one, the leaves of \( T \) must refer to instances of rule \( \equiv_1 \). We then proceed by induction on the structure of \( T \), taking into account that the structural similarity of expressions is a congruence and that every other inference rule preserves the equivalence of expressions.

\((\Longleftarrow)\) Follows from the rules \( \equiv_1 \) and \( \equiv_2 \), with \( F' = F'' = F''' \).

Proof of theorem 17: Follows by structural induction on the form of \( F \) from the rules in tables 1 and 3, the translation rules (3), propositions 6, 8, 9, 10, 16 and theorem 14(2).

Proof of theorem 18: Follows from theorem 17, proposition 16 and the translation rules (3).

Proof of theorem 20: Follows from the definitions, theorem 18 and proposition 19.
B Generalisations and further proofs

B.1 Sequential and interface operator boxes

In order to avoid proving properties separately for each of the ABC operators, and also to prepare the ground for future extensions of ABC, we will first slightly extend the framework.

First, $\Omega_{\mathcal{D}}, \Omega_{\mathcal{S}}$ and $\Omega_{\mathcal{I}}$ will be considered as instances of a more general class of operator boxes. A sequential operator box $\Omega_{\text{seq}}$ is an operator box such that: no place is isolated; there is exactly one e-labelled place, and exactly one x-labelled place; and, for every transition $v \in T_{\Omega_{\text{seq}}}$, $|v| = |v^*| = 1$ and $\lambda_{\Omega_{\text{seq}}}(v) = \varphi_{id}$. That is, $\Omega_{\text{seq}}$ 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 $\Omega_{\text{seq}}$ is the set $\text{dom}_{\Omega_{\text{seq}}}$ comprising all $\Omega_{\text{seq}}$-tuples of static and dynamic boxes such that at most one box is dynamic. An example of a a sequential operator box (other than those mentioned above) is $\Omega_{\mathcal{I}}$ shown on the left of figure 10, which was used in [5] to model a ternary iteration without self loop.

Similarly, scoping can be considered as an instance of a more general class of operator boxes. A unary communication interface operator box $\Omega_{\varphi}$, shown on the right of figure 10, is parameterised by an interface function $\varphi : \text{mult}(\mathcal{A}) \setminus \{\varnothing\} \rightarrow \mathcal{A}_r$, and has the domain of application $\text{dom}_{\Omega_{\varphi}} \equiv \text{abox}_{\text{stc}} \cup \text{abox}_{\text{dyn}}$. To ensure the finiteness of the nets created, we assume that there is no finite set of labels $A \subseteq \mathcal{A}_s$ such that $\text{mult}(A) \cap \text{dom}_{\varphi}$ is infinite; it may be checked that $\varphi_{\text{stc}}$ fulfills this constraint. The role of $\Omega_{\varphi}$ will be to effect the change of synchronous communication interface specified by $\varphi$.

![Figure 10](image)

**Fig. 10.** Sequential and interface operator boxes.

The current framework as well as results could be extended to other operator boxes, but extra conditions (like the factorisability and the e- x- or ex-directedness [5]) would then be necessary in order not to lose the properties established for ABC. However, the theory is already very expressive as it stands at the moment.
In what follows the sequential operator boxes, communication operator boxes, buffer restriction operator boxes, and the parallel composition operator box will be referred to as general operator boxes.

B.2 Some results on net substitution

The net substitution operation defined in section 4 for the binary ABC operator boxes may be readily extended to operators of any arity.

Let \( \Omega \) be a sequential operator with transitions \( v_1, \ldots, v_n \), and

\[
\Sigma = (\Sigma_1, \ldots, \Sigma_n) = (\Sigma_{v_1}, \ldots, \Sigma_{v_n})
\]

be a tuple of boxes in \( \text{dom}\Omega \). Then \( \Omega(\Sigma) = \Phi \) whose components are defined as follows.

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

\[
W_\Phi(p, w \prec t) = \begin{cases} W_{\Sigma_i}(p, t) & \text{if } w = v_i \\ 0 & \text{otherwise}, \end{cases}
\]

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

For every place \( s \in S_{\Phi} \) with \( s^* = \{u_1, \ldots, u_k\} \) and \( s^* = \{w_1, \ldots, w_m\} \), we construct in \( S_{\Phi} \) all the places of the form \( p \equiv (x_1, \ldots, x_k, e_1, \ldots, e_m) \), where each \( x_i \) is an exit place of \( \Sigma_{u_i} \), and each \( e_j \) is an entry place of \( \Sigma_{w_j} \). 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 \( w \prec t \), the weight function is given by:

\[
W_\Phi(p, w \prec t) = \begin{cases} W_{\Sigma_{u_i}}(x_i, t) + W_{\Sigma_{w_j}}(e_j, t) & \text{if } w \in s^* \text{ and } w = u_i = w_j \\ W_{\Sigma_{u_i}}(x_i, t) & \text{if } w \in s^* \text{ and } w = u_i \\ W_{\Sigma_{w_j}}(e_j, t) & \text{if } w \in s^* \text{ and } w = w_j \\ 0 & \text{otherwise}, \end{cases}
\]

and similarly for \( W_{\Omega}(w \prec t, p) \). In the proofs, we will denote by \( \mathbb{P}(s) \) the set of all places \( p \) as defined above for a given \( s \), and by \( \mathbb{P}(s, q) \) (resp. \( \mathbb{P}(s, q, q') \)) the sets of all those \( p \in \mathbb{P}(s) \) in which \( q \) (resp. \( q \) and \( q' \)) is present.

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

\[
W_{\Omega}(p^b, w \prec t) = W_{\Sigma_w}(p_{w}^b, t),
\]

and similarly for \( W_{\Omega}(w \prec t, p^b) \).

For a communication interface operator \( \Omega_\varphi \), the intuition behind a multiset \( \Gamma \) in the domain of \( \varphi \) is that some interface change can be applied to any finite set of transitions whose labels match the argument, i.e., the non-empty multiset
of actions \( \Gamma \). 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 \( \varphi_{id} \), no transition label is changed for them.) Hence, the application of a communication interface operator \( \Omega_{\varphi} \) to a box \( \Sigma \) results in a labelled net which is like \( \Sigma \) with the only difference that the set of transitions comprises all trees \( t \equiv \nu^\varphi_0 \preceq \{t_1, \ldots, t_l\} \) such that \( \{t_1, \ldots, t_l\} \in \text{mult}(T\Sigma) \) and the multiset \( \Lambda \equiv \{\lambda_{\Sigma}(t_1), \ldots, \lambda_{\Sigma}(t_l)\} \) belongs to the domain of \( \varphi \). The label of \( t \) is \( \varphi(\Lambda) \), and for a place \( p \) of \( \Omega_{\varphi}(\Sigma) \), the weight function is given by:

\[
W_{\Omega_{\varphi}(\Sigma)}(p, t) \equiv \sum_{i=1}^{l} W_{\Sigma}(p, t_i),
\]

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

We then obtain the following proposition, lemmata and general result on net substitution.

**Proposition 21.** Let \( \Omega \) be any sequential operator box or the parallel composition operator box \( \Omega || \), \( \Sigma \) be any \( \Omega \)-tuple of boxes, and \( \Sigma \) be any box.

1. \( M_{\Omega(\Sigma)}^{\text{ctr}} = \emptyset \) if \( M_{\Sigma_{vi}}^{\text{ctr}} = \emptyset \), for each \( v_i \in To \).
2. If and \( \Sigma' \) is \( \Omega_{\varphi}(\Sigma) \) or \( \Sigma \) is a box or \( \Sigma_B \), then \( M_{\Sigma'}^{\text{ctr}} = \emptyset \) if \( M_{\Sigma}^{\text{ctr}} = \emptyset \).

**Proof.** (1) Follows from the definitions and, in particular, from the control-restrictedness of \( \Omega \) and the ex-restrictedness of each box \( \Sigma_{vi} \). Indeed, due to such constraints, for each \( e \in \Sigma_{vi} \), if \( s \in \bullet v_i \), then \( p \in \mathcal{P}(s, e) \neq \emptyset \) is a control place in \( H(\Sigma) \), \( M_{\Sigma_{vi}}(e) \leq M_{\Omega(\Sigma)}(p) \). The situation is similar for \( x \in \Sigma_{vi} \), and the internal places of \( \Sigma_{vi} \) remain unchanged.

Notice that the property does not rely on the fact that \( \Sigma \) belongs to \( \text{dom}(\Omega) \), and that it would hold for any operator based on net substitution.

(2) Follows immediately from the fact that the control places are unchanged.

**Lemma 22.** If \( M \) is a clean marking of a box \( \Sigma \) such that \( M(e) + M(x) \geq 1 \), for all \( e \in \Sigma \) and \( x \in \Sigma^o \), then \( M^{\text{ctr}} \in \{\Sigma, \Sigma^o\} \).

**Proof.** Suppose that \( M^{\text{ctr}} \neq \Sigma \) and \( M^{\text{ctr}} \neq \Sigma^o \). Then, since \( M \) is clean, there are \( e \in \Sigma \) and \( x \in \Sigma^o \) such that \( M(e) = M(x) = 0 \), a contradiction.

**Lemma 23.** Let \( \Omega_{\varphi}(\Sigma) \) be a legal application of a sequential operator box \( \Omega_{\varphi} \), and \( z \in T_{\Omega_{\varphi}} \) be such that \( \Sigma = \Sigma_z \) is a dynamic box. Moreover, let \( s'' \) be a place in \( \Omega_{\varphi} \), \( M \equiv M_{\Omega_{\varphi}(\Sigma)} \), \( \bullet s = \{s\} \) and \( z^* = \{s'\} \).

1. If there is a place in \( \mathcal{P}(s'') \) which is marked at \( M \) then \( s'' \in \{s, s'\} \).
2. If \( M_{\Sigma}^{\text{ctr}} = \emptyset \) then \( M^{\text{ctr}} = \mathcal{P}(s) \), and if \( M_{\Sigma}^{\text{ctr}} = \Sigma^o \) then \( M^{\text{ctr}} = \mathcal{P}(s') \).
3. If \( M_{\Sigma}^{\text{ctr}} \geq \mathcal{P}(s'') \) then \( M^{\text{ctr}} = \mathcal{P}(s'') \), and one of the following holds:
   (a) \( s'' = s \neq s' \) and \( M_{\Sigma}^{\text{ctr}} = \emptyset \).
   (b) \( s'' = s' \neq s \) and \( M_{\Sigma}^{\text{ctr}} = \Sigma^o \).
(c) \( s'' = s = s' \) and \( M_{\Sigma}^{\text{ctr}} \in \{^0 \Sigma, \Sigma^0 \} \).

**Proof.** (1) Follows from the definition of \( \Omega_{\Sigma}(\Sigma) \) and the fact that all control places in the nets \( \Sigma_v \) for \( v \neq z \), are empty.

(2) Suppose that \( M_{\Sigma}^{\text{ctr}} = ^0 \Sigma \) (the case \( M_{\Sigma}^{\text{ctr}} = \Sigma^0 \) is symmetric). Then the control places marked in \( \Omega_{\Sigma}(\Sigma) \) are exactly those in whose construction the places from \( ^0 \Sigma \) were used. Moreover, \( M(q) = M_{\Sigma}(e) \), for all \( e \in ^0 \Sigma \) and \( q \in \mathbb{P}(s,e) \). Hence \( M_{\Sigma}^{\text{ctr}} = \mathbb{P}(s) \).

(3) From (1) it follows that \( s'' \in \{ s, s' \} \). Suppose that \( s'' = s \neq s' \) (the case \( s'' = s' \neq s \) is symmetric). Then \( M(q) = M_{\Sigma}(e) \), for all \( e \in ^0 \Sigma \) and \( q \in \mathbb{P}(s'', e) \). Hence, by lemma 22, \( M_{\Sigma}^{\text{ctr}} \in \{^0 \Sigma, \Sigma^0 \} \). Moreover, by part (2), \( M_{\Sigma}^{\text{ctr}} = \mathbb{P}(s'') \).

**Theorem 24.** Let \( \Omega \) be any general operator box and \( \Sigma \in \text{dom}_\Omega \). Then \( \Omega(\Sigma) \) is a box with a clean and ac-free marking. Moreover, if all the dynamic boxes (if any) in \( \Sigma \) have quasi-safe markings, then the marking of \( \Omega(\Sigma) \) is also quasi-safe.

**Proof.** The result is straightforward for operators other than the sequential ones, so suppose that \( \Omega \) is such an operator. Then \( \Omega(\Sigma) \) is ex-restricted since \( \mathbb{P}(s) \neq \emptyset \) for every \( s \in \Sigma_{\Omega} \) (due to the ex-restrictedness of the boxes in \( \Sigma \)) and, in particular, for \( s \in ^0 \Omega \cup \mathbb{P}^i \). Moreover, the \( \beta \)-restrictedness follows directly from definitions, and the control-restrictedness from the same property of the boxes in \( \Sigma \), and the fact that \( \mathbb{P}(s,p) \neq \emptyset \) whenever \( s \in \Sigma_{\Omega} \) and \( p \in ^0 \Sigma_v \) for \( v \in \bullet s \). That \( M(\Sigma_{\Omega}) \) is clean follows from lemma 23(3), with \( \{ s \} = ^0 \Omega \) or \( \{ s \} = \Omega^0 \), and to prove that it is ac-free (and possibly quasi-safe), we proceed in the following way.

If \( \Sigma \) are static boxes then, by definition, all the control places of \( \Omega(\Sigma) \) are marked and the property follows immediately from the control-restrictedness of \( \Omega(\Sigma) \). So, let us assume that \( \Sigma_{\Omega} \) is the unique dynamic box in \( \Sigma \) and take a transition \( v \prec t \) of \( \Omega(\Sigma) \). We will consider two cases:

Case 1: \( v \neq w \). If \( \bullet \) contains an internal (\( \pi \)-labelled) place \( p \), then \( p \) is also an internal place of \( \Omega(\Sigma) \), \( p \in \bullet (v \prec t) \) and \( M(\Omega_{\Sigma})(p) = M_{\Sigma}(p) = 0 \). Otherwise, \( \bullet \) contains a place \( p \) in \( ^0 \Sigma_v \cup \Sigma_v^0 \), and we assume that \( p \in ^0 \Sigma_v \) (the case \( p \in \Sigma_v^0 \) is symmetric). We also assume that \( \bullet v = \{ s \} \), and so \( \mathbb{P}(s,p) \subseteq \bullet (v \prec t) \). We then observe that the following hold:

- If \( q \in \bullet \bullet s \cup \bullet w \) then, by the definition of \( \Omega(\Sigma) \), for every \( q \in \mathbb{P}(v,p) \) we have \( M(\Omega_{\Sigma})(q) = 0 \). Hence the property holds.
- If \( w \in \bullet s \setminus s \) then, due to the cleaness of \( \Sigma_{\Omega} \), either \( M_{\Sigma}^{\text{ctr}} = \Sigma_{\Omega}^0 \), or there is \( e \in \Sigma_{\Omega}^0 \) such that \( M_{\Sigma}(e) = 0 \). In the former case, for every \( q \in \mathbb{P}(s,p) \), we have \( M(\Omega_{\Sigma})(q) = 1 \). In the latter case, for every \( q \in \mathbb{P}(s,p,e) \), we have \( M(\Omega_{\Sigma})(q) = 0 \). Hence the property holds.
- If \( w \in \bullet s \setminus \bullet \bullet s \) then, due to the cleaness of \( \Sigma_{\Omega} \), either \( M_{\Sigma}^{\text{ctr}} = \Sigma_{\Omega}^0 \), or there is \( e \in \Sigma_{\Omega}^0 \) such that \( M_{\Sigma}(e) = 0 \). In the former case, for every \( q \in \mathbb{P}(s,p) \),
we have $M_{O}(\Sigma)(q) = 1$. In the latter case, for every $q \in \mathbb{P}(s, p, e)$, we have $M_{\Omega}(\Sigma)(q) = 0$. Hence the property holds.

- If $w \in * \cap *$ then, due to the cleanness of $\Sigma_w$, either $M^{cfr}_{\Sigma_w} \in \{ ^{o}\Sigma_w, \Sigma_w \}$, or there are $e \in ^{o}\Sigma_w$ and $x \in \Sigma_w$ such that $M_{\Sigma_w}(e) = M_{\Sigma_w}(x) = 0$. In the former case, for every $q \in \mathbb{P}(s, p, e)$, we have $M_{O}(\Sigma)(q) = 1$. In the latter case, for every $q \in \mathbb{P}(s, p, e, x)$, we have $M_{\Omega}(\Sigma)(q) = 0$. Hence the property holds.

Case 2: $v = w$. Since $\Sigma_w$ is an ac-free labelled net, there is a control place $p$ of $\Sigma_w$ such that $M_{\Sigma_w}(p) < 2 \cdot W_{\Sigma_w}(p, t)$; in particular, this means that $1 \leq W_{\Sigma_w}(p, t)$ and so $p \in *$. If, moreover, $\Sigma_w$ is quasi-safe, we can choose $p$ such that $M_{\Sigma_w}(p) \leq 1$.

If $p$ is internal, then $p$ is also an internal place of $\Omega(\Sigma), p \in \bullet(w \triangleleft t)$ and

$$M_{O}(\Sigma)(p) = M_{\Sigma_w}(p) < 2 \cdot W_{\Sigma_w}(p, t) = 2 \cdot W_{\Omega}(\Sigma)(p, w \triangleleft t).$$

Moreover, $M_{O}(\Sigma)(p) = M_{\Sigma_w}(p) \leq 1$ in the quasi-safe case.

If $p \in ^{o}\Sigma_w$ (the case $p \in \Sigma_w$ is symmetric), let $s$ be the place in $\Omega$ such that $\{s\} = \bullet w$. We then observe that the following hold:

- If $w^* \neq \{s\}$ then, for every $q \in \mathbb{P}(s, p)$ we have

$$M_{O}(\Sigma)(q) = M_{\Sigma_w}(p) < 2 \cdot W_{\Sigma_w}(p, t) = 2 \cdot W_{\Omega}(\Sigma)(q, w \triangleleft t).$$

Moreover, $M_{O}(\Sigma)(q) = M_{\Sigma_w}(p) \leq 1$ in the quasi-safe case. Hence the property holds.

- If $w = \{s\} = w^*$ then, due to the cleanness of $\Sigma_w$, either $M^{cfr}_{\Sigma_w} = \Sigma_w$, or there is $x \in \Sigma_w$ such that $M_{\Sigma_w}(x) = 0$. In the former case, for every $q \in \mathbb{P}(s, p, e)$, we have $M_{O}(\Sigma)(q) = 1$. In the latter case, for every $q \in \mathbb{P}(s, p, e, x)$, we have

$$M_{O}(\Sigma)(q) = M_{\Sigma_w}(p) < 2 \cdot W_{\Sigma_w}(p, t) \leq 2 \cdot W_{\Omega}(\Sigma)(q, w \triangleleft t).$$

Moreover, $M_{O}(\Sigma)(q) = M_{\Sigma_w}(p) \leq 1$ in the quasi-safe case. Hence the property holds.

**Proposition 25.** Let $\Omega$ be a sequential operator box, and $\Sigma$ be an $\Omega$-tuple of static boxes.

1. If $v \in T_\Omega$ is such that $^v \Omega = ^* v$ or $^v \Omega = v^*$, then $\Omega(\Sigma) = \Omega(\Sigma')$, where $\Sigma'$ is $\Sigma$ with $\Sigma_v$ replaced respectively by $\Sigma_v$ or $\Sigma_w$.

2. If $v \in T_\Omega$ is such that $^{v^*} \Omega = ^* v$ or $^{v^*} \Omega = v^*$, then $\Omega(\Sigma) = \Omega(\Sigma')$, where $\Sigma'$ is $\Sigma$ with $\Sigma_v$ replaced respectively by $\Sigma_v$ or $\Sigma_w$.

**Proof.** Follows from the definitions.
B.3 General structural equivalence and related results

Let \( \Omega \) be a generalised operator box and \( \Sigma, \Theta \) be \( \Omega \)-tuples of boxes. We start by defining five auxiliary relations \( \equiv_1^\Omega \), in the following way.

- \( \Sigma \equiv_0^\Omega \Theta \) if there is a transition \( v \in T_\Omega \) and a box \( \Psi \) such that \( v = v^* \), \( \{v, \Theta_v\} = \{\overline{\Psi}, \Psi\} \) and \( \Sigma_u = \Theta_u \), for all \( u \in T_\Omega \setminus \{v\} \).
- \( \Sigma \equiv_1^\Omega \Theta \) if there are two transitions \( v \neq w \in T_\Omega \) and two boxes \( \Psi_v \) and \( \Psi_w \) such that \( v^* = w^* \), \( \{(\Sigma_v, \Sigma_w), (\Theta_v, \Theta_w)\} = \{\overline{\Psi_v}, \Psi_v\} \) and \( \Sigma_u = \Theta_u \), for all \( u \in T_\Omega \setminus \{v, w\} \).
- \( \Sigma \equiv_2^\Omega \Theta \) if there are two transitions \( v \neq w \in T_\Omega \) and two boxes \( \Psi_v \) and \( \Psi_w \) such that \( v^* = w^* \), \( \{(\Sigma_v, \Sigma_w), (\Theta_v, \Theta_w)\} = \{\overline{\Psi_v}, \Psi_v\} \) and \( \Sigma_u = \Theta_u \), for all \( u \in T_\Omega \setminus \{v, w\} \).
- \( \Sigma \equiv_3^\Omega \Theta \) if, for each \( v \in T_\Omega \), there is a box \( \Psi_v \) 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 \( \equiv_4^\Omega \equiv_1^\Omega \bigcup_{i \in I} \equiv_i^\Omega \), where \( I = \{0, 1, 2, 3, 5\} \) and \( \equiv_5^\Omega \equiv \text{id}_{\text{abox}} \).

Note that \( \equiv_0^\Omega \equiv_4^\Omega \) and \( \equiv_5^\Omega = \text{id}_{\text{abox}} \) for every unary general operator box \( \Omega \).

**Proposition 26.** Let \( \Omega \) be a generalised operator box, and \( \Sigma \in \text{dom}_\Omega \).

1. If \( \Sigma \equiv_0^\Omega \Theta \), then \( \Theta \in \text{dom}_\Omega \) and \( [\Sigma] = [\Theta] \).
2. \( \equiv_0^\Omega \) is an equivalence relation on \( \text{dom}_\Omega \).
3. If \( \Theta \in \text{dom}_\Omega \) and \( [\Sigma] = [\Theta] \), then \( \Omega(\Sigma) = \Omega(\Theta) \) iff \( \Sigma \equiv_0^\Omega \Theta \).

**Proof.** (1) The result is obvious for all unary operator boxes. For the parallel operator box, it follows from the observation that \( \equiv_1^\Omega = \equiv_4^\Omega \) only modifies the marking of the open buffer places, and consequently preserves the property of being a static or dynamic component (see also proposition 2(1)). For a sequential operator box \( \Omega \), it suffices to observe that the relations \( \equiv_3^\Omega \) only act on the markings of the components and preserve the number of static and dynamic boxes in \( \Sigma \). Indeed, in the case of \( \equiv_4^\Omega \) this follows from proposition 2(1), and in the remaining cases can be shown by case analysis. Take, for instance, the relation \( \equiv_2^\Omega \) with \( (\Sigma_v, \Sigma_w) = (\overline{\Psi_v}, \Psi_v) \) and \( (\Theta_v, \Theta_w) = (\Psi_v, \overline{\Psi_w}) \). Then, by the definition of \( \text{dom}_\Omega \), \( (\Sigma_v, \Sigma_w) \in \text{abox}^{\text{dyn}} \times \text{abox}^{\text{stc}} \). Thus, by proposition 2(2), we have \( \Psi_v \in \text{abox}^{\text{stc}} \), and so \( (\Theta_v, \Theta_w) \in \text{abox}^{\text{stc}} \times \text{abox}^{\text{dyn}} \). Hence \( \Theta \in \text{dom}_\Omega \).

(2) The result is again obvious for all unary operator boxes. For the parallel operator box, it is an easy observation that \( \equiv_1^\Omega = \equiv_4^\Omega \) is reflexive, symmetric and transitive.
Consider now a sequential operator \( \Omega \). That \( \equiv^{\Omega} \) is reflexive follows from
\[ id_{\Delta_{\Omega}} \subseteq (\equiv^{4}_{\Omega} \circ \equiv^{5}_{\Omega}). \] Moreover, it is easy to see that each \( \equiv^{i}_{\Omega} \) is symmetric and
\[ (\equiv^{i}_{\Omega} \circ \equiv^{j}_{\Omega}) = (\equiv^{i}_{\Omega} \circ \equiv^{j}_{\Omega}), \] for every \( i \in I \). Hence \( \equiv^{\Omega} \) is also symmetric. That it is also transitive on \( \text{dom}_{\Omega} \) results from the following two facts holding for all \( i, j \in I \), where in the second case, the domains of the relations are restricted to \( \text{dom}_{\Omega} \):
\[
(\equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega}) = (\equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega}) = (\equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega}) \
(\equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega}) \subseteq \bigcup_{i \in I} \equiv^{i}_{\Omega}.
\]
The former fact is straightforward, while the latter can be shown by case analysis (using, in particular, the property that in any \( \Sigma \in \text{dom}_{\Omega} \), each transition in \( \Omega \) has only one pre-place and one post-place, and at most one \( \Sigma_{v} \) is a dynamic box). For example, one can easily see that:
\[
(\equiv^{1}_{\Omega} \circ \equiv^{0}_{\Omega}) \subseteq \equiv^{3}_{\Omega}, \quad (\equiv^{1}_{\Omega} \circ \equiv^{1}_{\Omega}) \subseteq (\equiv^{1}_{\Omega} \cup \equiv^{5}_{\Omega}), \quad (\equiv^{1}_{\Omega} \circ \equiv^{2}_{\Omega}) = \emptyset.
\]

(3) Since \( [\Sigma] = [\Theta] \), \( \Sigma \) and \( \Theta \) may only differ in their markings. For a communication interface operator box, the property is obvious since it only modifies the transitions (and the corresponding arcs), in a way that only depends on the transition labels. For a buffer restriction operator box, the property is also obvious since it only modifies the status of the same place in \( \Sigma \) and \( \Theta \), and adds the same empty place to both of them. For the parallel operator box, the property results from the fact that the marking of the non-open buffer places is unchanged, while the marking of the open buffer places is modified in a way that is exactly preserved by \( \equiv_{\Omega} = \equiv^{\Omega} \).

Finally, we consider a sequential operator box \( \Omega \). Then \( \Omega(\Sigma) = \Omega(\Theta) \) since \( [\Sigma] = [\Theta] \). Moreover, by the definition of net substitution,
\[
M_{\text{op}}^{\Omega(\Sigma)} = M_{\text{op}}^{\Omega(\Theta)} \quad \iff \quad \sum_{v \in T_{\Omega}} M_{\Sigma_{v}}^{\text{op}} = \sum_{v \in T_{\Theta}} M_{\Theta_{v}}^{\text{op}},
\]
where we identify the marking of the open buffer places with the multiset of buffer symbols it represents, and this exactly corresponds to the property preserved by \( \equiv^{\Omega} \). Now, the closed buffer places (with their marking) are the same, by definition, in \( \Sigma \) and \( \Omega(\Sigma) \), as well as in \( \Theta \) and \( \Omega(\Theta) \), and they are not influenced by \( \equiv_{\Omega} \), so that we only need to consider the control places.

We first consider the (\( \iff \)) direction of the property. If all the boxes in \( \Sigma \) are static, then \( \Sigma \equiv^{4}_{\Omega} \Theta \) and we clearly have \( \Omega(\Sigma) = \Omega(\Theta) \). What we still need to show is that for every \( i \in I, \Sigma \equiv^{i}_{\Omega} \Theta \) implies \( M_{\Omega(\Sigma)}^{\text{ctr}} = M_{\Omega(\Theta)}^{\text{ctr}} \). We consider the following three cases (the proof for \( i \in \{2, 3\} \) is similar to that for \( i = 1 \)):

- \( \Sigma \equiv^{0}_{\Omega} \Theta \). Suppose that \( v \) and \( \Psi \) are as in the definition of \( \equiv^{0}_{\Omega} \). Without loss of generality, we may assume that \( \Sigma_{v} = \overline{\Psi} \) and \( \Theta_{v} = \overline{\Psi} \). Thus, by the clearness of \( \Sigma_{v} \) and \( \Theta_{v} \), we have \( M_{\Sigma_{v}}^{\text{ctr}} = M_{\Theta_{v}}^{\text{ctr}} \). Therefore, hence, by lemma 22(2), \( M_{\Omega(\Sigma)}^{\text{ctr}} = P(s) = M_{\Omega(\Theta)}^{\text{ctr}} \), where \( \{s\} = v = v^{*} \).
Asynchronous Box Calculus

1. \( \Sigma \equiv_{\theta}^{1} \Theta \). Suppose that \( v, w \) and \( \Psi, \Psi \) are as in the definition of \( \equiv_{\theta}^{1} \).

Without loss of generality, we may assume that \( \Sigma_v = \overline{\Psi} \) and \( \Theta_w = \overline{\Psi} \). Thus, by the cleanness of \( \Sigma_v \) and \( \Theta_w \), we have \( M_{\Sigma_v} = \circ \Sigma_v \) and \( M_{\Theta_w} = \circ \Theta_w \).

Hence, by lemma 22(2), \( M_{\Theta_w}^{\text{tr}} = \mathbb{P}(s) = M_{\Theta_w}^{\overline{\Theta}(\Theta)} \), where \( \{ s \} = \Psi = \Psi_w \).

2. \( \Sigma \equiv_{\theta}^{1} \Theta \). Then \( \Sigma = \Theta \) and we are done.

We now consider the \( (\implies) \) direction of the property. If \( \Sigma \) is a tuple of static boxes, then \( M_{\Theta_w}^{\text{tr}} = \circ \Theta_w \), and \( \Theta \) is also a tuple of static boxes. Indeed, if there is a marked \( \theta \)-labelled place \( p \) in some \( \Theta_w \), then \( p \) is also a marked \( \theta \)-labelled place in \( \Theta \) (in the sense of \( \Theta_w \)), producing a contradiction. Similarly, if there is a marked entry place \( e \) in some \( \Theta_w \), with \( \{ e \} = \{ s \} \), then each place \( q \in \mathbb{P}(s, e) \) is also marked in \( \Theta_w \) (in the sense of \( \Theta_w \)), again producing a contradiction (the argument is similar if there is a marked exit place in some \( \Theta_w \)). Hence, in this case, \( \Sigma \equiv_{\theta}^{1} \Theta \), and so \( \Sigma \equiv_{\theta}^{1} \Theta \).

Let us now assume there is one dynamic box in \( \Sigma \), for the transition \( v \in T_r \), with \( \Psi = \{ s \} \) and \( \Psi = \{ s' \} \).

For every \( \theta \)-labelled place \( p \) in every \( \Sigma_v \), by definition, we have \( p \in \Omega(\Sigma) = \Omega(\Theta) \) and \( M_{\Sigma_v}(p) = M_{\Theta}(p) = M_{\Theta}(p) = M_{\Theta}(p) \).

For the places in any \( \mathbb{P}(s') \), where \( s'' \in S_r \), i.e., those constructed from the entry/exit places of \( \Sigma \) and \( \Theta \), we consider two cases.

Case 1: \( M_{\Theta}^{\text{tr}} \not\in \circ \Sigma_v \). By the cleanness of \( \Sigma_v \), there is \( e \in \circ \Sigma_v \) and \( x \in \Sigma_e \) with \( M_{\Sigma_v}(e) = 0 = M_{\Sigma_v}(x) \). Hence, for every place \( s'' \in \Omega \) and every place \( q \) belonging to

\[
\mathbb{P}(s'') \quad \text{if } s'' \not\in \{ s, s' \}, \quad \mathbb{P}(s'', e) \quad \text{if } s'' = s \neq s', \quad \mathbb{P}(s'', x) \quad \text{if } s'' = s' = s',
\]

we have that \( M_{\Theta}(x) = 0 = M_{\Theta}(q) \), and all the places in \( \Theta \) used to construct \( q \) have the empty marking. In particular, this applies to \( e, x \) and all the entry/exit places of \( \Theta_w \) for \( w \neq v \). Moreover, as a consequence, for every entry place \( e' \neq e \) in \( \Theta_w \), if we replace \( e \) by \( e' \) in place \( q \in \mathbb{P}(s, e) \) or \( q \in \mathbb{P}(s, e, x) \) above, we will have \( M_{\Theta}(x) = M_{\Theta}(e') = M_{\Theta}(q) = M_{\Theta}(e') \), and similarly for any exit place \( x' \neq x \) in \( \Theta_w \). Hence, \( \Sigma \equiv_{\theta}^{1} \Theta \).

Case 2: \( M_{\Theta}^{\text{tr}} = \circ \Sigma_v \) (the case \( M_{\Theta}^{\text{tr}} = \Sigma_v \) is similar). Now, by lemma 22(2), for every \( q \in \mathbb{P}(s), M_{\Theta}(q) = 1 = M_{\Theta}(q) \). Let us choose one of those \( q \)'s: exactly one of the entry/exit places used to construct \( q \) must be marked with one token in \( \Theta \), while the other ones have the empty marking. Let us assume that this is due to an exit place \( x \in \Theta_w \), so that \( M_{\Theta_w}(x) = 1 \) and \( \Psi = \{ s \} \) (the case where the token comes from an entry place \( e \) in some \( \circ \Theta_w \) with \( \Psi = \{ s \} \) is similar). If we replace the \( x \) in \( q \) by any other exit place \( x' \) of \( \Theta_w \), we will obtain another place in \( \mathbb{P}(s) \), with a marking which may only come from \( x' \).

Thus, by the cleanness of \( \Theta_w \), \( \Theta_w^{\text{tr}} = \Theta_w \). Moreover, \( \Theta_z \) must be a static box, for every \( x \in T_r \setminus \{ w \} \). We then observe that if \( v = w \), then \( \Sigma (\equiv_{\theta}^{1} \circ \equiv_{\theta}^{1} \Theta \), and if \( v \neq w \), then \( \Sigma (\equiv_{\theta}^{1} \circ \equiv_{\theta}^{1} \Theta) \).

**Proposition 27.** Let \( \Omega \) be an \( n \)-ary sequential operator box and \( \Sigma \in \text{dom}_\Omega \).

1. If \( \Sigma_i \mid \Theta_i \) (for \( i \leq n \)), then \( \Theta \in \text{dom}_\Omega \), and

\[
\Omega(\Sigma) = \{(v_1 \bullet U_1) \cup \ldots \cup (v_n \bullet U_n) ) \} \Omega(\Theta).
\]
2. If $\Omega(\Sigma) \models [U] \Theta$, then there are $\Psi, \Phi \in \text{dom}_\Omega$ and steps $U_1, \ldots, U_n$ such that 
$\Sigma \equiv_{\Omega} \Psi, \Psi_i[U_i] \Phi_i$ (for $i \leq n$), $\Theta = \Omega(\Phi)$ and 
$U = (v_1 \downarrow U_1) \cup \ldots \cup (v_n \downarrow U_n)$.

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

Proof: Below, for any box $\Phi$ and $b \in \mathbb{B}$, $b$ will represent the only $b$-labelled open buffer place of $\Phi$.

(1) Since $\Sigma_i[U_i] \Theta_i$ (for $i \leq n$), each $\Theta_i$ is a static (resp. dynamic) box iff so is $\Sigma_i$. Hence $\Theta \in \text{dom}_\Omega$ as $\Sigma \in \text{dom}_\Omega$. If $\Sigma$ are static boxes, then each $U_j$ is empty, $\Sigma = \Theta$ and $\Omega(\Sigma) \models \Theta(\Theta)$. Otherwise, $\Sigma$ has exactly one dynamic component $\Sigma_i$. We then observe that, for all $j \neq i$, $U_j = \emptyset$ and $\Theta_j = \Sigma_j$. Moreover, the following hold:

- For every internal or closed buffer place $p$ in $\Sigma_i$, we have 
  
$$M_{\Omega(\Sigma)}(p) = M_{\Sigma_i}(p) \geq \sum_{t \in U_i} W_{\Sigma_i}(p, t) = \sum_{t \in U_i} W_{\Omega(\Sigma)}(p, v_i < t) \quad \text{and} \quad \sum_{t \in U_i} W_{\Sigma_i}(t, p) = \sum_{t \in U_i} W_{\Omega(\Sigma)}(v_i < t, p).$$

- For every $b \in \mathbb{B}$, we have:
  
$$M_{\Omega(\Sigma)}(b) \geq M_{\Sigma_i}(b) \geq \sum_{t \in U_i} W_{\Sigma_i}(b, t) = \sum_{t \in U_i} W_{\Omega(\Sigma)}(b, v_i < t) \quad \text{and} \quad \sum_{t \in U_i} W_{\Sigma_i}(t, b) = \sum_{t \in U_i} W_{\Omega(\Sigma)}(v_i < t, b).$$

- For every place $s \in S_{\Omega}$ and every place $q = (e_1, \ldots, e_k, x_1, \ldots, x_m) \in \mathbb{P}(s)$, we have (below $P \equiv \{e_1, \ldots, e_k, x_1, \ldots, x_m\} \cap S_{\Sigma_i}$):
  
$$M_{\Omega(\Sigma)}(q) = \sum_{p \in P} M_{\Sigma_i}(p) \geq \sum_{p \in P} \sum_{t \in U_i} W_{\Sigma_i}(p, t) = \sum_{p \in P} W_{\Omega(\Sigma)}(q, v_i < t) \quad \text{and} \quad \sum_{p \in P} \sum_{t \in U_i} W_{\Sigma_i}(t, p) = \sum_{p \in P} W_{\Omega(\Sigma)}(v_i < t, q).$$

Hence $\Omega(\Sigma) \models [(v_1 \downarrow U_1) \cup \ldots \cup (v_n \downarrow U_n)] \Omega(\Theta)$.

(2) If $\Omega(\Sigma) \models [U] \Theta$, then there is a tuple of steps $U_1, \ldots, U_n$ such that $U = (v_1 \downarrow U_1) \cup \ldots \cup (v_n \downarrow U_n)$. If $U = \emptyset$, then each $U_i$ is empty too, and the property obviously holds, with $\Psi = \Phi = \Sigma$. This will be the case, in particular, if $\Sigma$ are static boxes. Otherwise, we will assume that $\Sigma_i$ is the only dynamic box in $\Sigma$ and consider two cases:

Case 1: $M_{\Sigma_i}^{\text{var}} \not\subseteq \{\circ \Sigma_i, \Sigma_i^v\}$. By the cleanness of $\Sigma_i$, there is $e \in \circ \Sigma_i$ and $x \in \Sigma_i^v$ with $M_{\Sigma_i}(e) = 0 = M_{\Sigma_i}(x)$. For all $j \neq i$ and $t \in T_{\Sigma_j}$, we observe that the following hold:
- If \( q \) is an internal place in \( \psi_i \), we have that \( q \in \psi(v_j \prec t) \) and \( M_{D(\Sigma)}(q) = M_{\Sigma_i}(q) = 0 \).
- If \( p \) is an entry place in \( \psi_i \) and \( v_j = \{ s \} \) (or \( p \) is an exit place in \( \psi_i \) and \( v_j^e = \{ s \} \)), for every

\[
q \in \mathcal{P}(s, p) \quad \text{if} \quad v_i \neq \{ s \} \neq v_j^e, \\
q \in \mathcal{P}(s, p, x) \quad \text{if} \quad v_i \neq \{ s \} = v_j^e, \\
q \in \mathcal{P}(s, p, e) \quad \text{if} \quad v_i = \{ s \} = v_j^e,
\]

we have that \( q \in \psi(v_j \prec t) \) and \( M_{D(\Sigma)}(q) = 0 \).

Hence \( U_j = \emptyset \), for all \( j \neq i \).

Let \( \psi_i \) be \( \Sigma_i \) with one change, namely for each \( b \in \mathbb{B} \), \( M_{\psi_i}(b) = M_{D(\Sigma)}(b) \); moreover, for all \( j \neq i \), let \( \psi_j \) be \( \Sigma_j \) with the marking of all the open buffer places set to 0. We then have \( \Sigma = \bigcup \psi_i \) and the following hold:

- For every internal or closed buffer place \( p \) in \( S_{\psi_i} \), we have:

\[
M_{\psi_i}(p) = M_{\Sigma_i}(p) = M_{D(\Sigma)}(p) \geq \sum_{t \in U_i} W_{D(\Sigma)}(p, v_i \prec t) = \sum_{t \in U_i} W_{\psi_i}(p, t).
\]

- For every \( b \in \mathbb{B} \), we have:

\[
M_{\psi_i}(b) = M_{D(\Sigma)}(b) \geq \sum_{t \in U_i} W_{D(\Sigma)}(b, v_i \prec t) = \sum_{t \in U_i} W_{\psi_i}(b, t).
\]

- For every entry place \( p \) in \( S_{\psi_i} \) (the situation is symmetric if \( p \) is an exit place), for every \( q \in \mathcal{P}(s, p) \) if \( v_i = \{ s \} \neq v_j^e \), we have:

\[
M_{\psi_i}(p) = M_{\Sigma_i}(p) = M_{D(\Sigma)}(p) \geq \sum_{t \in U_i} W_{D(\Sigma)}(q, v_i \prec t) = \sum_{t \in U_i} W_{\psi_i}(q, t);
\]

and for every \( q \in \mathcal{P}(s, p, x) \) if \( v_i = \{ s \} = v_j^e \), we have:

\[
M_{\psi_i}(p) = M_{\Sigma_i}(p) = M_{D(\Sigma)}(q) \geq \sum_{t \in U_i} W_{D(\Sigma)}(q, v_i \prec t) = \sum_{t \in U_i} (W_{\psi_i}(p, t) + W_{\psi_i}(x, t)) \geq \sum_{t \in U_i} W_{\psi_i}(p, t).
\]

Hence \( \psi_i(U_i) \psi_i \) and \( \psi_j(U_j) \psi_j \) for all \( j \neq i \), so that from the first part of the proposition, we obtain that \( \Theta = \Omega(\Phi) \).

Case 2: \( M_{\Sigma}^{\text{tr}} = \Sigma_i \) (the case \( M_{\Sigma}^{\text{tr}} = \Sigma_o \) is similar). Suppose that \( v_i = \{ s \} \).

Then, by lemma 23(2), \( M_{D(\Sigma)} = \mathcal{P}(s) \). Hence, for every \( v \prec t \in U \), we have the following: \( \psi \) does not contain any internal places; if \( \psi \cap o \Sigma \neq \emptyset \) then \( v = \{ s \} \); and if \( \psi \cap o \Sigma \neq \emptyset \) then \( v^* = \{ s \} \).

Suppose now that \( v \prec t, v' \prec t' \in U \), where \( v \neq v' \). Then assume that there is \( e \in \psi \cap o \Sigma \) and \( e' \in \psi' \cap o \Sigma \). We have, for every \( q \in \mathcal{P}(s, e, e') \), that

\[
1 = M_{D(\Sigma)}(q) \geq W_{D(\Sigma)}(q, v \prec t) + W_{D(\Sigma)}(q, v' \prec t') = W_{\Sigma_i}(e, t) + W_{\Sigma_i}(e', t') \geq 2.
\]
Hence we obtained a contradiction (the situation is similar if \( c \) is replaced by an 
\( x \in \Sigma_i \cap \Sigma_i^o \) and/or \( c' \) is replaced by an \( x' \in \Sigma_i^o \cap \Sigma_i^o \)). As a consequence, there
is a unique non-empty \( U_j \). We then consider two cases:

- \( j = i \). We may first observe that it is not possible to have \( t, t' \in U_i \) (we do
not exclude here the case where \( t = t' \)) with \( e \in \Sigma_i \setminus \Sigma_i^o \) and \( x \in \Sigma_i^o \setminus \Sigma_i^o \)
because either \( v_i^* = \{ s' \} \neq \{ s \} \) and for \( q \in \mathbb{P}(s', x) \) we have

\[
M_{\Theta_i}(q) = 0 < 1 \leq W_{\Sigma_i}(x, t') = W_{\Theta_i}(q, v_i < t')
\]
or \( v_i^* = \{ s \} \) and, for \( q \in \mathbb{P}(s, e, x) \), we have

\[
M_{\Theta_i}(q) = 1 < 2 \leq W_{\Sigma_i}(e, t)+W_{\Sigma_i}(x, t') = W_{\Theta_i}(q, v_i < t)+W_{\Theta_i}(q, v_i < t')
\]

Hence the control pre-places of the transitions in \( U_i \) are either all in \( \Sigma_i \)
or all in \( \Sigma_i^o \). In the first case, we may not have \( t = t' \in U_i \) with some \( e \in \Sigma_i \setminus \Sigma_i^o \) since then, for every \( q \in \mathbb{P}(s, e) \), we would have had:

\[
M_{\Theta_i}(q) = 1 < 2 \leq W_{\Sigma_i}(e, t)+W_{\Sigma_i}(e, t') = W_{\Theta_i}(q, v_i < t)+W_{\Theta_i}(q, v_i < t')
\]

Now, for all \( e \in \Sigma_i \) and \( q \in \mathbb{P}(s, e) \), we have:

\[
M_{\Sigma_i}(e) = M_{\Theta_i}(q) \geq \sum_{t \in U_i} W_{\Theta_i}(q, v_i < t) = \sum_{t \in U_i} W_{\Sigma_i}(e, t)
\]

Moreover, for every closed buffer place \( p \in S_{\Sigma_i} \), we have:

\[
M_{\Sigma_i}(p) = M_{\Theta_i}(p) \geq \sum_{t \in U_i} W_{\Theta_i}(p, v_i < t) = \sum_{t \in U_i} W_{\Sigma_i}(p, t)
\]

Define \( \Phi_i \) to be \( \Sigma_i \) with the only change that, for each \( b \in B \), \( M_{\Phi_i}(b) = M_{\Theta_i}(b) \); moreover, for all \( m \neq i \), let \( \Phi_m \) be \( \Sigma_m \) with the marking of all its
open buffer places set to 0. We then have, for every \( b \in B \),

\[
M_{\Phi_i}(b) = M_{\Theta_i}(b) \geq \sum_{t \in U_i} W_{\Theta_i}(b, v_i < t) = \sum_{t \in U_i} W_{\Phi_i}(b, t)
\]

Hence \( \Phi_i \mid U_i \Phi_i \) and \( \Phi_m \mid U_m \Phi_m \), for all \( m \neq i \), with \( \Psi \equiv_{\Omega} \Sigma \), and so from
the first part of the proposition, \( \Theta \equiv_{\Omega} \Phi \).

In the second case, i.e., \( U_i \subseteq \Sigma_i^o \), we must have \( v_i^* = \{ s \} = v_i^* \) and we may replace in \( \Sigma \) the tokens in the entry places of \( \Sigma_i \) by one token in each exit
place of \( \Sigma_i \), leading to an \( \Omega \)-equivalent tuple of arguments (see the definition of \( \equiv^{\omega}_{\Omega} \)), to which we may apply the previous analysis, when \( M_{\omega_i}^{\omega_i} = \Sigma_i^o \).

- \( j \neq i \). Then either \( v_i^* = \{ s \} \) and we replace \( \Sigma \) by \( \Psi \) where the only modifications consist in replacing \( M_{\Phi_i}^{\omega_i} \) by the empty marking and \( M_{\Phi_i}^{\omega_i} \) by \( \omega_i \Phi_i \) (see the definition of \( \equiv^{\omega}_{\Omega} \)), or \( v_i^* = \{ s \} \) and we replace \( \Sigma \) by \( \Phi_i \) where the only modifications consist in replacing \( M_{\Phi_i}^{\omega_i} \) by the empty marking and \( M_{\Phi_i}^{\omega_i} \) by \( \Phi_i \) (see the definition of \( \equiv^{\omega}_{\Omega} \)), and we have the same situation as before with
\( \Sigma \equiv_{\Omega} \Psi \).
Notice that for the sequential operators, at most one \( U \) is non-empty. Otherwise, the property is very similar to that which holds for the parallel operator box.

**Proposition 28.** Let \( \Sigma \) be a static or dynamic box, and \( \Omega_\varphi \) be a communication interface operator box.

1. If \( \Sigma [U_1 \uplus \ldots \uplus U_k] \Theta \) and each \( \lambda_\Sigma(U_i) \) belongs to the domain of \( \varphi \), then
   \[
   \Omega_{\varphi}(\Sigma) \{ \{ \nu \varphi < U_1, \ldots, \nu \varphi < U_k \} \} \Omega_\varphi(\Theta).
   \]

2. If \( \Omega_{\varphi}(\Sigma) [U] \Psi \) then there is a box \( \Theta \) and steps of transitions \( U_1, \ldots, U_k \) such that \( \Psi = \Omega_\varphi(\Theta) \), \( U = \{ \nu \varphi < U_1, \ldots, \nu \varphi < U_k \} \), and \( \Sigma [U_1 \uplus \ldots \uplus U_k] \Theta \).

Note: As a consequence, \( \text{enabled}(\Omega_{\varphi}(\Sigma)) \) comprises exactly all
\[
U = \{ \nu \varphi < U_1, \ldots, \nu \varphi < U_k \}
\]
such that \( U_1 \uplus \ldots \uplus U_k \in \text{enabled}(\Sigma) \) and each \( \lambda_\Sigma(U_i) \) belongs to the domain of \( \varphi \).

**Proof.** Follows from the definitions.