
# COMPUTING SCIENCE

## Interval Temporal Logic Semantics of Box Algebra

Hanna Klaudel, Maciej Koutny and Zhenhua Duan

**TECHNICAL REPORT SERIES**

**No. CS-TR-1373 February 2013**

**TECHNICAL REPORT SERIES**

**No. CS-TR-1373 February, 2013**

# Interval Temporal Logic Semantics of Box Algebra

**H. Klaudel, M. Koutny and Z. Duan**

## Abstract

In this paper, we further develop a recently introduced semantical link between temporal logics and Petri nets. We focus on two specific formalisms, Interval Temporal Logic (ITL) and Box Algebra (BA), which are closely related by their compositional approach to constructing systems' descriptions. The overall goal of our investigation is to translate Petri nets into behaviourally equivalent logic formulas. As a result, the analysis of system properties can be carried out using either of the two formalisms, taking advantage of their respective strengths and powerful tool support. The contribution of this paper is twofold. First, we extend the existing translation from BA to ITL, by removing restrictions concerning the way in which control flow of concurrent system is modelled, and by allowing fully general synchronisation operator. Second, we strengthen the notion of equivalence between a Petri net and the corresponding logic formula, by proving such an equivalence at the level of transition based executions of Petri nets, rather than just by looking at their labels.


Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

# Bibliographical details

KLAUDEL, H., KOUTNY, M., DUAN, Z.

Interval Temporal Logic Semantics of Box Algebra
[By] H. Klaudel, M. Koutny, Z. Duan

Newcastle upon Tyne: Newcastle University: Computing Science, 2013.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1373)

### Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1373

### Abstract

In this paper, we further develop a recently introduced semantical link between temporal logics and Petri nets. We focus on two specific formalisms, Interval Temporal Logic (ITL) and Box Algebra (BA), which are closely related by their compositional approach to constructing systems' descriptions. The overall goal of our investigation is to translate Petri nets into behaviourally equivalent logic formulas. As a result, the analysis of system properties can be carried out using either of the two formalisms, taking advantage of their respective strengths and powerful tool support.

The contribution of this paper is twofold. First, we extend the existing translation from BA to ITL, by removing restrictions concerning the way in which control flow of concurrent system is modelled, and by allowing fully general synchronisation operator. Second, we strengthen the notion of equivalence between a Petri net and the corresponding logic formula, by proving such an equivalence at the level of transition based executions of Petri nets, rather than just by looking at their labels.

### About the authors

Hanna Klaudel is a Professor at the University of Evry, France.

Maciej Koutny is a Professor in the School of Computing Science, Newcastle University. His research interests centre on the theory of distributed and concurrent systems, including both theoretical aspects of their semantics and application of formal techniques to the modelling and verification of such systems; in particular, model checking based on net unfoldings. He has also investigated non-interleaving semantics of priority systems, and the relationship between temporal logic and process algebras. Recently, he has been working on the development of a formal model combining Petri nets and process algebras as well as on Petri net based behavioural models of membrane systems.

Zhen-hua Duan obtained his B.Sc. (1982) and M.Sc. (1987) degrees from Northwest University (China), and Ph.D. (1996) degree from University of Newcastle upon Tyne (UK). He became a lecturer in 1984 and was promoted to a professor in 1995 at Northwest University. In 1989 he joined Computer Science Department at University of Sheffield (UK) to work as a Visiting Scholar. From 1990 to 1997, he worked as a research associate in three universities including University of Ulster (UK), University of Newcastle upon Tyne and University of Sheffield. From1997 to 2002, he worked as a (experienced/ senior) Software Engineer in three companies including Chorleys, Airsys ATM, and Altera in UK. In2003 he joined Xidian University (China) as a professor of Computer Science. His research interests concentrate on concurrent, real-time and hybrid systems, including modeling, simulation, and verification of such systems. In addition, he is interested in temporal logic programming, formal languages and automata, and formal semantics. He is also interested in design and development of high-tech software.

### Suggested keywords

ITL
PETRI NET
BOX ALGEBRA
COMPOSITION
SEMANTICS
GENERAL SYNCHRONISATION
STEP SEQUENCE
EQUIVALENCE

# Interval Temporal Logic Semantics of Box Algebra

Hanna Klaudel[1], Maciej Koutny[2], and Zhenhua Duan[3]

[1] IBISC, Université d'Évry, 91000 Évry, France
klaudel@ibisc.univ-evry.fr
[2] School of Computing Science, Newcastle University
Newcastle upon Tyne, NE1 7RU, United Kingdom
maciej.koutny@newcastle.ac.uk
[3] School of Computer Science and Engineering
Xidian University, Xi'an, P.R.China
zhenhua_d@yahoo.com

**Abstract.** In this paper, we further develop a recently introduced semantical link between temporal logics and Petri nets. We focus on two specific formalisms, Interval Temporal Logic (ITL) and Box Algebra (BA), which are closely related by their compositional approach to constructing systems' descriptions. The overall goal of our investigation is to translate Petri nets into behaviourally equivalent logic formulas. As a result, the analysis of system properties can be carried out using either of the two formalisms, taking advantage of their respective strengths and powerful tool support.
The contribution of this paper is twofold. First, we extend the existing translation from BA to ITL, by removing restrictions concerning the way in which control flow of concurrent system is modelled, and by allowing fully general synchronisation operator. Second, we strengthen the notion of equivalence between a Petri net and the corresponding logic formula, by proving such an equivalence at the level of transition based executions of Petri nets, rather than just by looking at their labels.

*Keywords:* ITL, Petri net, box algebra, composition, semantics, general synchronisation, step sequence, equivalence

## 1 Introduction

In general, temporal logics [6, 11] and Petri nets [5, 16] are regarded as fundamentally different approaches to the specification and analysis of concurrent systems. Temporal logics allow one to specify both the system designs and correctness requirements within the same framework, and the verification of correctness can be done by checking the satisfaction of logic formulas. Petri nets, on the other hand, are an automata inspired model with semantics based on actions and local states which allows one to capture causal relationships in systems' behaviour. As a result, verification of behavioural properties can be carried out using invariant techniques [19] based on the graph structure of nets, or model checking techniques based on partial order reductions [20].

To establish a semantical link between logics and Petri nets, we focused in [3] on two specific formalisms, Interval Temporal Logic (ITL) [13, 15] and Box Algebra (BA) [1], which are closely related by the their compositional approaches to constructing systems' descriptions. In particular, in both ITL and BA the control flow of a system is specified using commonly used programming operators, such as sequence, choice, parallelism and iteration. The synchronisation between concurrently executed subsystems is, however, achieved in different ways and therefore needs to be suitably handled.

In [3] we proved correctness of the translation from a submodel of BA to semantically equivalent ITL formulas. The submodel we considered disallowed the nesting of the parallel composition operator. Moreover, synchronisation was binary. In this paper, we provide a syntax-driven translation for the core BA [1] syntax comprising parallel composition, sequence, choice, synchronisation and iteration, but without considering data variables. Crucially, we relax the syntactical constraints forbidding the use of the parallel composition outside the topmost level. Moreover, we consider a fully general synchronisation operator. Finally, we strengthen the notion of equivalence between a BA net and the corresponding ITL formula, by proving such an equivalence at the level of transition based executions of Petri nets, rather than just by taking their labels.

The paper is organised as follows. In the next section we recall the basic notions of Box Algebra. We start with the definition of box expressions which are then used to compositionally construct box nets. We also recall a number of results demonstrating how the execution semantics of a composite box can be derived from the executions semantics of its components. In Section 3 we similarly recall the relevant fragment of Internal Temporal Logic, and present some basic semantical properties of ITL formulas. Section 4 is central to the whole paper as it contains a formal translation from box expressions to ITL formulas as well as the proof of a semantical equivalence between box expressions and the corresponding formulas. The next section presents examples of this translation, and Section 6 briefly discusses future work.

Throughout the paper we use the standard mathematical notation. In particular, $\mathbb{N}$ denotes the set of all positive integers, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ and $\mathbb{N}_\omega = \mathbb{N}_0 \cup \{\omega\}$, where $\omega$ denotes the first transfinite ordinal. We extend to $\mathbb{N}_\omega$ the standard arithmetic comparison operators, assuming that $\omega = \omega$ and $n < \omega$, for all $n \in \mathbb{N}_0$. Moreover, we define $\preceq$ as $\leq$ without the pair $(\omega, \omega)$. The concatenation operator for sequences of sets will be denoted by "$\circ$", and for sequences of symbols by "$\cdot$". For a symbol $s$ and a set of sequences $S$, we will write $s \cdot S$ to denote the set $\{s \cdot s' \mid s' \in S\}$. We will also denote $\varnothing^\infty = \{\varnothing\varnothing \ldots\}$ and $\varnothing^* = \{\epsilon, \varnothing, \varnothing\varnothing, \ldots\}$, i.e., $\varnothing^\infty$ comprises a single infinite sequence, and $\varnothing^*$ an infinite number of finite sequences.

## 2 Box algebra

Let $\mathcal{A}$ be a finite set of actions (or action labels). A synchronisation relation $\rho$ is any set of tuples of actions $(a_1, \ldots, a_n, a)$ such that $n \geq 1$. Intuitively, $a_1, \ldots, a_n$

represent $n$ concurrent actions of which can be synchronised to yield a single composite action with the label $a$. To reflect this intuition we will often denote $(a_1, \ldots, a_n, a)$ by $a_1 \ldots a_n \mapsto a$.

The syntax given below defines two kinds of box expressions, namely non-synchronised expressions $E$ capturing the control flow in a concurrent system, and synchronised expressions $F$ (below $a$ is an action and $\rho$ a synchronisation relation):

$$\begin{array}{lcl} E & ::= & \mathsf{stop} \mid a \mid E\,;E \mid E \,\Box\, E \mid E \parallel E \mid [\![E \circledast E \circledast E]\!] \\ F & ::= & E \,\mathsf{sco}\, \rho \end{array}$$

The intuition behind the above syntax is that: $(i)$ $\mathsf{stop}$ denotes a blocked process; $(ii)$ $a$ denotes a process which can execute an action $a \in \mathcal{A}$ and terminate; $(iii)$ $E\,;E'$ denotes sequential composition; $(iv)$ $E \,\Box\, E'$ denotes choice composition; $(v)$ $E \parallel E'$ denotes parallel composition; $(vi)$ $[\![E \circledast E' \circledast E'']\!]$ denotes a loop with an initial part $E$, iterated part $E'$, and terminal part $E''$; and $(vii)$ $E \,\mathsf{sco}\, \rho$ denotes scoping which combines action synchronisation and restriction in a single construct, i.e., $\mathsf{sco}$ enforces all the synchronisations specified by $\rho$ and blocks all non-synchronised actions.

### 2.1 Box nets

The semantics of box expressions is given through a mapping into Petri nets called boxes. (Note that one can also define the semantics of box expressions through suitable SOS rules of operational semantics; the two semantics can be shown to coincide [1].)

A box is a tuple $N = (P, T, F, \ell)$ such that: $(i)$ $P$ and $T$ are disjoint finite sets of respectively places (representing local states) and transitions (representing actions); $(ii)$ $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation; and $(iii)$ $\ell$ is a labelling function for places and transitions such that $\ell(p) \in \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$, associating an entry, internal or exit status with every place $p$, and $\ell(t) \in \mathcal{A}$, for every transition $t$.

The sets of entry, internal and exit places of $N$ are given respectively by $N^{\mathsf{e}} = \ell^{-1}(\mathsf{e})$, $N^{\mathsf{i}} = \ell^{-1}(\mathsf{i})$ and $N^{\mathsf{x}} = \ell^{-1}(\mathsf{x})$. Moreover, we set $N^{\mathsf{ei}} = N^{\mathsf{e}} \cup N^{\mathsf{i}}$ and $N^{\mathsf{ix}} = N^{\mathsf{i}} \cup N^{\mathsf{x}}$, and will use $N^{\mathsf{p}}$ and $N^{\mathsf{t}}$ to respectively denote the places and transitions of $N$. We also adopt the standard rules about representing nets as directed graphs.

### 2.2 Semantics of box nets

The global states of a box $N$ are called markings, each marking being a mapping $M : N^{\mathsf{p}} \to \mathbb{N}_0$ assigning a natural number to every place. Markings are indicated by tokens drawn inside circles representing places. The default initial (or entry) and final (or exit) markings of $N$, denoted respectively by $M_N^{\mathsf{e}}$ and $M_N^{\mathsf{x}}$, are defined, for every $p \in P$, as follows:

$$M_N^{\mathsf{e}}(p) = \begin{cases} 1 \text{ if } p \in N^{\mathsf{e}} \\ 0 \text{ otherwise} \end{cases} \quad \text{and} \quad M_N^{\mathsf{x}}(p) = \begin{cases} 1 \text{ if } p \in N^{\mathsf{x}} \\ 0 \text{ otherwise} \end{cases}$$

The change of a marking of $N$ results from a simultaneous execution of a set of transitions, called a step. Formally, a step of $N$ is any set of transitions $U \subseteq N^{\mathsf{t}}$. It is enabled at a marking $M$ if, for every place $p \in N^{\mathsf{p}}$:

$$M(p) \geq |\{t \in U \mid (p,t) \in F\}| \ .$$

We denote this by $M[U\rangle$. An enabled step $U$ can be executed leading to a marking $M'$ given, for every place $p \in N^{\mathsf{p}}$, by:

$$M'(p) = M(p) - |\{t \in U \mid (p,t) \in F\}| + |\{t \in U \mid (t,p) \in F\}| \ .$$

We denote this by $M[U\rangle M'$.

The semantics of $N$ is given through its step sequences starting from the default initial marking $M_N^{\mathsf{e}}$. We will assume that each such step sequences is infinite which is a harmless requirement as any finite step sequence can be extended by an infinite sequence of empty steps (note that $M[\varnothing\rangle M$ for every marking $M$). In addition, we single out a set of finite step sequences which lead from the default initial marking $M_N^{\mathsf{e}}$ to the default final marking $M_N^{\mathsf{x}}$. Intuitively, each such step sequence will be interpreted as a successfully terminated execution of $N$.

A step sequence of $N$ is an infinite sequence $\theta = U_1 U_2 \ldots$ of steps such that there exist markings $M_1, M_2, \ldots$ of $N$ satisfying

$$M_N^{\mathsf{e}}[U_1\rangle M_1 \quad M_1[U_2\rangle M_2 \quad M_2[U_3\rangle M_3 \quad \ldots$$

Moreover, we define a terminated step sequence of $N$ as a finite sequence $\theta = U_1 \ldots U_m$ $(m \geq 0)$ of steps such that there exist markings $M_1, \ldots, M_{m-1}$ of $N$ satisfying

$$M_N^{\mathsf{e}}[U_1\rangle M_1 \quad M_1[U_2\rangle M_2 \quad M_2[U_3\rangle M_3 \quad \ldots \quad M_{m-1}[U_m\rangle M_N^{\mathsf{x}} \ .$$

The sets of step sequences and terminated sequences of $N$ are respectively denoted by $\mathsf{steps}(N)$ and $\mathsf{termsteps}(N)$. The $k$-th element of a step sequence $\theta$ will be denoted by $\theta_{(k)}$, and its length by $|\theta|$.

### 2.3 Composite boxes

The above definition of a box net is too general. In the BA approach one is only interested in nets derived compositionally, following the syntax of box expressions. The labelling of places provides the necessary device for composing boxes along the entry and exit interfaces, i.e., the sets of entry places $N^{\mathsf{e}}$ and exit places $N^{\mathsf{x}}$.

As a consequence of such a compositional approach to constructing boxes, the nets we are going to define will have very specific form of their places and transitions, making it easier to establish connections between boxes and ITL formulas. Intuitively, we will use the syntax of a non-synchronised box expression $E$ to construct concrete places and transitions of the corresponding box $N$, by

embedding paths from the root of the parse tree of $E$ in the definitions of places and transitions. In what follows, finite sequences in the set

$$\Pi = \{ ;_L, ;_R, \Box_L, \Box_R, \shuffle_L, \shuffle_R, \circledast_L, \circledast_M, \circledast_R \}^*$$

will be called syntax paths. Note that symbols appearing in syntax paths correspond to the arguments of operators used in box expressions (with '$L$' indicating the left argument, etc). For two syntax paths, $\pi_1$ and $\pi_2$, we denote $\pi_1 | \pi_2$ if $\{\pi_1, \pi_2\} = \{\pi \cdot \shuffle_L \cdot \pi', \pi \cdot \shuffle_R \cdot \pi''\}$, for some $\pi$, $\pi'$ and $\pi''$. Intuitively, two actions of a non-synchronised box expression are concurrent iff their positions, $\pi_1$ and $\pi_2$, in the parse tree are such that $\pi_1 | \pi_2$.

The form of each place in compositionally defined boxes will be $p_Z$, where $p \in \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$ and $Z \subset \Pi \cdot \{\mathsf{e}, \mathsf{x}\}$, while each transition will be of the form $a_W$, where $a \in \mathcal{A}$ and $W \subset \Pi$. Moreover, for brevity, the sets $Z$ and $W$ will be written as coma separated lists without brackets.

For a syntax path $\pi \in \Pi$ and transition $a_W$, we denote $\pi \cdot a_W = a_{\pi \cdot W}$. This prefix notation extends in the usual way to sets of transitions and sequences of sets of transitions as well as (sets of) places.

The specific form of places and transitions, together with the systematic way in which boxes are manipulated below, will mean that for a compositional box $N = (P, T, F, \ell)$ it will be the case that, for all $p_Z \in P$ and $a_W \in T$, $\ell(p_Z) = p$ and $\ell(a_W) = a$, as well as:

$$\begin{aligned} (p_Z, a_W) \in F &\iff \exists \pi \in W \colon \pi \cdot \mathsf{e} \in Z \\ (a_W, p_Z) \in F &\iff \exists \pi \in W \colon \pi \cdot \mathsf{x} \in Z \ . \end{aligned}$$

As a result, we will represent such a box simply by $N = (P, T)$.

We will now present a systematic way of constructing composite boxes. For each constant expression we define a specific box and, for each operator used in box expressions, we define a corresponding operator on boxes, in the following way.

*Constants.* With the blocking expression $\mathsf{stop}$ and a single-action expression $a \in \mathcal{A}$, we respectively associate the following two simple boxes:

$$\begin{aligned} \mathsf{N}_{\mathsf{stop}} &= (\{\mathsf{e}_\mathsf{e}, \mathsf{x}_\mathsf{x}\}, \varnothing) \\ \mathsf{N}_a \quad &= (\{\mathsf{e}_\mathsf{e}, \mathsf{x}_\mathsf{x}\}, \{a_\epsilon\}) \ . \end{aligned} \tag{1}$$

Their diagrams are depicted in Figure 1 with labels shown inside the nodes.

*Sequence.* $N ; K$ combines the exit interface of $N$ with the entry interface of $K$. The entry interface of the resulting box is that of $N$, and the exit interface is that of $K$; for an example see the diagram of $\mathsf{N}_a ; \mathsf{N}_b$ in Figure 1.

$$N ; K \ = \ \big( \ P_L \ \cup \ P_R \ \cup \ \mathcal{X} \ , \ T_L \ \cup \ T_R \ \big) \tag{2}$$

where

$$\begin{aligned} T_L &= \,;_L{\cdot}N^{\mathsf{t}} \qquad T_R = \,;_R{\cdot}K^{\mathsf{t}} \\ P_L &= \,;_L{\cdot}N^{\mathsf{ei}} \qquad P_R = \,;_R{\cdot}K^{\mathsf{ix}} \\ \mathcal{X} &= \{\mathsf{i}_{;_L{\cdot}Z \,\cup\, ;_R{\cdot}W} \mid \mathsf{x}_Z \in N^{\mathsf{x}} \wedge \mathsf{e}_W \in K^{\mathsf{e}}\}\,. \end{aligned} \tag{3}$$

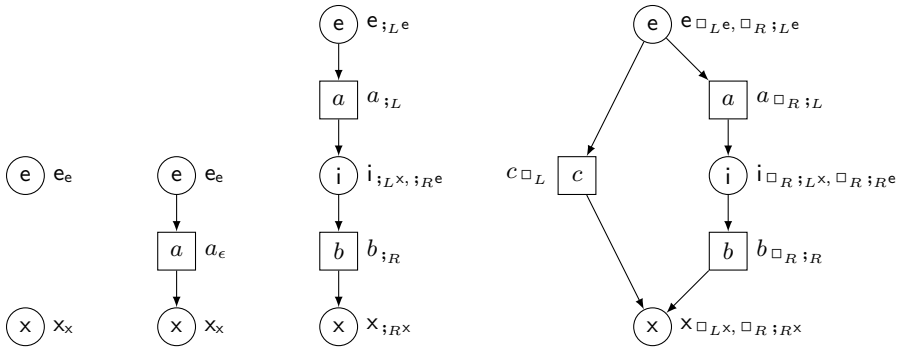


**Fig. 1.** From the left to right: diagrams of $\mathsf{N}_{\mathsf{stop}}$, $\mathsf{N}_a$, $\mathsf{N}_a\,;\mathsf{N}_b$ and $\mathsf{N}_c\,\Box\,(\mathsf{N}_a\,;\mathsf{N}_b)$. Note that the last two boxes correspond to the box expressions $a\,;b$ and $c\,\Box\,(a\,;b)$.

*Choice.* $N\,\Box\,K$ combines together the entry interfaces of the two boxes creating a new entry interface, as well as their exit interfaces creating a new exit interface; for an example see the diagram of $\mathsf{N}_c\,\Box\,(\mathsf{N}_a\,;\mathsf{N}_b)$ in Figure 1.

$$N\,\Box\,K \;=\; \big(\; P_L \;\cup\; P_R \;\cup\; \mathcal{X} \;\cup\; \mathcal{Y}\;,\; T_L \;\cup\; T_R\;\big) \tag{4}$$

where

$$\begin{aligned} T_L &= \,\Box_L{\cdot}N^{\mathsf{t}} \qquad T_R = \,\Box_R{\cdot}K^{\mathsf{t}} \\ P_L &= \,\Box_L{\cdot}N^{\mathsf{i}} \qquad P_R = \,\Box_R{\cdot}K^{\mathsf{i}} \\ \mathcal{X} &= \{\mathsf{e}_{\Box_L{\cdot}Z \,\cup\, \Box_R{\cdot}W} \mid \mathsf{e}_Z \in N^{\mathsf{e}} \wedge \mathsf{e}_W \in K^{\mathsf{e}}\} \\ \mathcal{Y} &= \{\mathsf{x}_{\Box_L{\cdot}Z \,\cup\, \Box_R{\cdot}W} \mid \mathsf{x}_Z \in N^{\mathsf{x}} \wedge \mathsf{x}_W \in K^{\mathsf{x}}\}\,. \end{aligned} \tag{5}$$

*Parallelism.* $N \,\|\, K$ puts next to each other the boxes $N$ and $K$; for an example see the diagram of $((a \,\|\, a)\,;b) \,\|\, (b\,;c)$ on the left of Figure 3. The new entry (resp. exit) interface is the union of the entry (resp. exit) interfaces of the composed boxes.

$$N \,\|\, K \;=\; \big(\; P_L \;\cup\; P_R\;,\; T_L \;\cup\; T_R\;\big) \tag{6}$$

where

$$\begin{aligned} T_L &= \,\|_L{\cdot}N^{\mathsf{t}} \qquad T_R = \,\|_R{\cdot}K^{\mathsf{t}} \\ P_L &= \,\|_L{\cdot}N^{\mathsf{p}} \qquad P_R = \,\|_R{\cdot}K^{\mathsf{p}} \end{aligned} \tag{7}$$

*Iteration.* $[\![N \circledast K \circledast J]\!]$ combines the exit interfaces of $N$ and $K$ with the entry interfaces of $K$ and $J$; for an example see the diagram of $[\![\mathsf{N}_a \circledast (\mathsf{N}_b \parallel \mathsf{N}_c) \circledast \mathsf{N}_d]\!]$ on the left of Figure 2. The new entry interface is that of $N$, and the exit interface is that of $J$.

$$[\![N \circledast K \circledast J]\!] = \big( P_L \cup P_M \cup P_R \cup \mathcal{X} \,,\, T_L \cup T_M \cup T_R \big) \tag{8}$$

where

$$\begin{aligned} T_L &= \circledast_L{\cdot}N^{\mathsf{t}} \qquad T_M = \circledast_M{\cdot}K^{\mathsf{t}} \qquad T_R = \circledast_R{\cdot}J^{\mathsf{t}} \\ P_L &= \circledast_L{\cdot}N^{\mathsf{ei}} \qquad P_M = \circledast_M{\cdot}K^{\mathsf{i}} \qquad P_R = \circledast_R{\cdot}J^{\mathsf{ix}} \\ \mathcal{X} &= \{\mathsf{i}_{\circledast_L{\cdot}Z \cup \circledast_M{\cdot}W \cup \circledast_M{\cdot}V \cup \circledast_R{\cdot}Y} \mid \\ &\qquad \mathsf{x}_Z \in N^{\mathsf{x}} \wedge \mathsf{e}_W \in K^{\mathsf{e}} \wedge \mathsf{x}_V \in K^{\mathsf{x}} \wedge \mathsf{e}_Y \in J^{\mathsf{e}}\} \end{aligned} \tag{9}$$

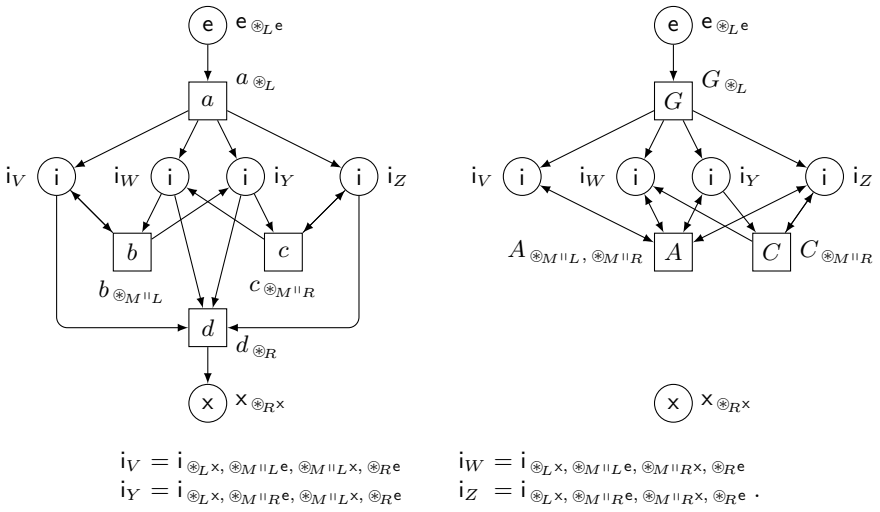


**Fig. 2.** Diagrams of two boxes involving iteration: $[\![\mathsf{N}_a \circledast (\mathsf{N}_b \parallel \mathsf{N}_c) \circledast \mathsf{N}_d]\!]$ (left) and $([\![\mathsf{N}_a \circledast (\mathsf{N}_b \parallel \mathsf{N}_c) \circledast \mathsf{N}_d]\!])\,\mathsf{sco}\,\{a \mapsto G, bc \mapsto A, c \mapsto C\}$ (right). Note that the four internal places are defined below the diagrams.

At this point it is possible to formulate a useful result which holds for all boxes which can be constructed using the rules defined so far (note that it is obvious that any net constructed in this way is a box).

**Fact 1** *Let $N$ be any net constructed from boxes corresponding to the constant box expressions as well as the operators of sequence, choice, parallelism and iteration. Then the set of steps appearing in the step sequences of $N$ is included in the following set:*

$$N^{\mathsf{psteps}} = \big\{\{a^1_{\pi_1}, \ldots, a^n_{\pi_n}\} \subseteq N^{\mathsf{t}} \mid \forall i < j : \pi_i | \pi_j\big\} \ .$$

*Moreover, if the net* $\mathsf{N}_{\mathsf{stop}}$ *defined in* (1) *is not used in the construction, then* $N^{\mathsf{psteps}}$ *is exactly the set of steps appearing in the step sequences of* $N$.

We will call $N^{\mathsf{psteps}}$ the set of potential steps of $N$. Moreover, for every synchronisation relation $\rho$, we define:

$$\rho_N = \{(U, a_{\{\pi_1,\ldots,\pi_n\}}) \mid U = \{a^1_{\pi_1}, \ldots, a^n_{\pi_n}\} \in N^{\mathsf{psteps}} \wedge (a^1, \ldots, a^n, a) \in \rho\} \ .$$

For example, if we take the box $N$ on the left of Figure 2, then we have:

$$\begin{aligned} N^{\mathsf{psteps}} &= \{\varnothing, \{a_{\circledast_L}\}, \{b_{\circledast_{M\shuffle L}}\}, \{c_{\circledast_{M\shuffle R}}\}, \{d_{\circledast_R}\}, \{b_{\circledast_{M\shuffle L}}, c_{\circledast_{M\shuffle R}}\}\} \\ \rho_N &= \{(\{a_{\circledast_L}\}, \{G_{\circledast_L}\}), (\{c_{\circledast_{M\shuffle R}}\}, \{C_{\circledast_{M\shuffle R}}\}), \\ &\quad (\{b_{\circledast_{M\shuffle L}}, c_{\circledast_{M\shuffle R}}\}, \{A_{\circledast_{M\shuffle L}, \circledast_{M\shuffle R}}\})\} \ , \end{aligned}$$

where $\rho = \{a \mapsto G, bc \mapsto A, c \mapsto C\}$.

*Scoping.* $N \,\mathsf{sco}\, \rho$ has the same places as $N$ and, for each potential step of $N$, one creates a new transition (this transition represents a synchronisation of two or more actions of $N$ if the potential step is not a singleton). After that all the transitions of $N$ are removed. Formally,

$$N \,\mathsf{sco}\, \rho \ = \ (\ N^{\mathsf{p}}\ ,\ \mathcal{Z}\ ) \tag{10}$$

where:

$$\mathcal{Z} = \{t \mid (U, t) \in \rho_N\} \ . \tag{11}$$

For example, the diagram of $[\![\mathsf{N}_a \circledast (\mathsf{N}_b \,\|\, \mathsf{N}_c) \circledast \mathsf{N}_d]\!] \,\mathsf{sco}\, \{a \mapsto G, bc \mapsto A, c \mapsto C\}$ is depicted on the right of Figure 2.

Note that by defining a suitable synchronisation relation $\rho$, one can capture all practically useful forms of synchronisation of two or more actions (e.g., $bc \mapsto A$ above), as well as action relabelling (e.g., $a \mapsto G$ and $c \mapsto C$) and action restriction (as for $d$).

### 2.4 From expressions to boxes

We can now define the semantics of box expressions by transforming them compositionally into the corresponding box nets, and then adopting the execution semantics of the latter. Formally, we define a mapping $\mathsf{box}(.)$ from expressions to boxes, in the following way:

$$\begin{aligned} \mathsf{box}(\mathsf{stop}) &= \mathsf{N}_{\mathsf{stop}} \\ \mathsf{box}(a) &= \mathsf{N}_a \\ \mathsf{box}(E \,;\, E') &= \mathsf{box}(E) \,;\, \mathsf{box}(E') \\ \mathsf{box}(E \,\Box\, E') &= \mathsf{box}(E) \,\Box\, \mathsf{box}(E') \\ \mathsf{box}(E \,\|\, E') &= \mathsf{box}(E) \,\|\, \mathsf{box}(E') \\ \mathsf{box}([\![E \circledast E' \circledast E'']\!]) &= [\![\mathsf{box}(E) \circledast \mathsf{box}(E') \circledast \mathsf{box}(E'')]\!] \\ \mathsf{box}(E \,\mathsf{sco}\, \rho) &= \mathsf{box}(E) \,\mathsf{sco}\, \rho \end{aligned} \tag{12}$$

From now on, by a box we will mean a composite box constructed using (12). According to the BA theory, such boxes enjoy a number of interesting behavioural properties when we consider executions starting from their initial markings, as follows:

- The number of tokens on any place for any reachable marking is bounded; more precisely, the number of tokens on any internal place is never greater than two (i.e., the internal places are 2-bounded), and an entry or exit place never holds more than one token (i.e., the entry and exit places are 1-bounded).
- Each box is clean which means that when all the exit places contain tokens, there is no token left elsewhere in the net.
- Boxes do not allow auto-concurrency which means that there is no transition enabled 'twice' for any reachable marking. As a consequence, steps can be represented by sets rather than by multiset of transitions.

### 2.5 Behavioural properties of composite box

Both finite and infinite step sequences of composite boxes exhibit clear compositional properties [1], i.e., one can (easily) derive the semantics of a composite box from the semantics of the composed boxes. This is demonstrated by a series of results which follow from the general properties of boxes [1].

For boxes modelling the blocking expression and a single-action expression, the semantics capture is straightforward.

**Fact 2 (basic boxes)** *Let $a \in \mathcal{A}$. Then:*

$$\mathsf{steps}(\mathsf{N}_{\mathsf{stop}}) = \varnothing^{\infty} \qquad \mathsf{steps}(\mathsf{N}_a) = \varnothing^{\infty} \ \cup \ \varnothing^{*} \circ \{\{a_\epsilon\}\} \circ \varnothing^{\infty}$$
$$\mathsf{termsteps}(\mathsf{N}_{\mathsf{stop}}) = \varnothing \qquad \mathsf{termsteps}(\mathsf{N}_a) = \varnothing^{*} \circ \{\{a_\epsilon\}\} \circ \varnothing^{*} \ .$$

For choice and parallelism, the semantics of a composite box can easily be expressed in terms of the semantics of the composed boxes. We need, however, a notion of parallel composition of step sequences.

For two step sequences, $\theta$ and $\tau$, of equal length,[1] $\theta \parallel \delta$ is a step sequence of the same length as $\theta$ and $\tau$, and $(\theta \parallel \delta)_{(k)} = \theta_{(k)} \cup \delta_{(k)}$, for all $k$. Moreover, for two sets of step sequences, $\Theta$ and $\Delta$, the set $\Theta \parallel \Delta$ comprises all step sequences $\theta \parallel \delta$, where $\theta \in \Theta$ and $\delta \in \Delta$ are of equal length.

**Fact 3 (choice and parallelism)** *Let $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$. Then:*

$$\mathsf{f}(N \,\square\, K) = \ \square_L{\cdot}\mathsf{f}(N) \ \cup \ \square_R{\cdot}\mathsf{f}(N)$$
$$\mathsf{f}(N \parallel K) = \ \ \shuffle_L{\cdot}\mathsf{f}(N) \ \parallel \ \shuffle_R{\cdot}\mathsf{f}(N) \ .$$

[1] That is, either $\theta$ and $\tau$ are infinite, or both are finite and have the same length.

**Fact 4 (sequence)**

$$\begin{aligned}\mathsf{steps}(N\,;K) &= {;_L}{\cdot}\mathsf{steps}(N) \cup \\ &\quad {;_L}{\cdot}\mathsf{termsteps}(N) \circ {;_R}{\cdot}\mathsf{steps}(K) \\ \mathsf{termsteps}(N\,;K) &= {;_L}{\cdot}\mathsf{termsteps}(N) \circ {;_R}{\cdot}\mathsf{termsteps}(K)\ .\end{aligned}$$

**Fact 5 (iteration)**

$$\begin{aligned}\mathsf{steps}([\![N \circledast K \circledast J]\!]) &= {\circledast_L}{\cdot}\mathsf{steps}(N) \cup \\ &\quad {\circledast_L}{\cdot}\mathsf{termsteps}(N) \circ ({\circledast_M}{\cdot}\mathsf{termsteps}(K))^* \\ &\qquad \circ {\circledast_M}{\cdot}\mathsf{steps}(K) \cup \\ &\quad {\circledast_L}{\cdot}\mathsf{termsteps}(N) \circ ({\circledast_M}{\cdot}\mathsf{termsteps}(K))^* \\ &\qquad \circ {\circledast_R}{\cdot}\mathsf{steps}(J) \\ \mathsf{termsteps}([\![N \circledast K \circledast J]\!]) &= {\circledast_L}{\cdot}\mathsf{termsteps}(N) \circ ({\circledast_M}{\cdot}\mathsf{termsteps}(K))^* \\ &\qquad \circ {\circledast_R}{\cdot}\mathsf{termsteps}(J)\ .\end{aligned}$$

Finally, we consider a box $N\,\mathsf{sco}\,\rho$, where $N$ is constructed from some non-synchronised box expression. In this case, relating step sequences of $N\,\mathsf{sco}\,\rho$ and $N$ is more involved.

First, we define a relation $\widetilde{\rho}_N$ comprising all pairs $(U, \{t_1, \ldots, t_k\})$ $(k \geq 0)$, where $U \in N^{\mathsf{psteps}}$ and $\{t_1, \ldots, t_k\} \subseteq \mathsf{box}(N\,\mathsf{sco}\,\rho)^{\mathsf{t}}$ are such that there is a partition $U_1, \ldots, U_k$ of $U$ satisfying $(U_j, t_j) \in \rho_N$, for each $j \leq k$. Moreover, for two equal length sequences of sets of transitions, $\tau$ and $\theta$, we denote $(\tau, \theta) \in \widetilde{\rho}_N$ if $(\tau_{(j)}, \theta_{(j)}) \in \widetilde{\rho}_N$, for all $j$.

**Fact 6 (scoping)** *Let* $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$. *Then:*

$$\mathsf{f}(N\,\mathsf{sco}\,\rho) = \{\theta \mid \exists\, \tau \in \mathsf{f}(N)\ :\ (\tau, \theta) \in \widetilde{\rho}_N\}\ .$$

### 2.6 Streamlined box expression

The last stage in the subsequent translation from BA to ITL is particularly simple for the class of streamlined synchronised expressions. We call a box expression $E\,\mathsf{sco}\,\rho$ streamlined if, for each transition $a_\pi \in \mathsf{box}(E)^{\mathsf{t}}$ there is exactly one transition $b_W \in \mathsf{box}(E\,\mathsf{sco}\,\rho)^{\mathsf{t}}$ such that $\pi \in W$.

It turns out that each synchronised box expression $F = E\,\mathsf{sco}\,\rho$ can be transformed into semantically equivalent streamlined expression $\mathsf{stl}(F) = F' = E'\,\mathsf{sco}\,\rho'$, in the following way. First, for every $a_\pi \in \mathsf{box}(E)^{\mathsf{t}}$, let

$$\mathsf{trans}(a_\pi) = \{b_W \in \mathsf{box}(F)^{\mathsf{t}} \mid \pi \in W\} = \{t \mid \exists (U, t) \in \rho_{\mathsf{box}(F)}\ :\ a_\pi \in U\}\ .$$

For example, if we take $[\![a \circledast (b \parallel c) \circledast d]\!]\,\mathsf{sco}\,\{a \mapsto G, bc \mapsto A, c \mapsto C\}$ with the corresponding box depicted in Figure 2, then we have:

$$\begin{aligned}&\mathsf{trans}(a_{\circledast_L}) = \{G_{\circledast_L}\} \qquad \mathsf{trans}(b_{\circledast_{M\parallel_L}}) = \{A_{\circledast_{M\parallel_L}}, {\circledast_{M\parallel_R}}\} \\ &\mathsf{trans}(d_{\circledast_R}) = \varnothing \qquad\qquad \mathsf{trans}(c_{\circledast_{M\parallel_R}}) = \{A_{\circledast_{M\parallel_L}}, {\circledast_{M\parallel_R}}, C_{\circledast_{M\parallel_R}}\}\end{aligned}$$

Then a suitable $E'$ is obtained by replacing each occurrence of an action $a \in \mathcal{A}$ in $E$ corresponding to transition $a_\pi$ in $\mathsf{box}(E)$,[2] by:

- $\mathsf{stop}$ if $\mathsf{trans}(a_\pi) = \varnothing$;
- $b_W$ if $\mathsf{trans}(a_\pi) = \{b_W\}$; and
- $b^1_{W_1} \,\Box\, (\ldots \Box\, (b^{m-1}_{W_{m-1}} \,\Box\, b^m_{W_m}) \ldots)$ if $\mathsf{trans}(a_\pi) = \{b^1_{W_1}, \ldots, b^m_{W_m}\}$ and $m \geq 2$.[3]

Furthermore,

$$\rho' = \{\underbrace{b_W \ldots b_W}_{|W| \text{ times}} \mapsto b_W \mid b_W \in \mathsf{box}(E \,\mathsf{sco}\, \rho)^{\mathsf{t}}\}$$

defines a suitable synchronisation relation. For the previous example, we obtain $F' = E' \,\mathsf{sco}\, \rho'$, with

$$E' = (\llbracket \gamma \circledast (\alpha \parallel (\alpha \,\Box\, \zeta)) \circledast \mathsf{stop} \rrbracket) \quad \text{and} \quad \rho' = \{\gamma \mapsto \gamma, \alpha\alpha \mapsto \alpha, \zeta \mapsto \zeta\} \,,$$

where $\gamma = G_{\circledast_L}$, $\alpha = A_{\circledast_{M \uplus L}, \circledast_{M \uplus R}}$ and $\zeta = C_{\circledast_{M \uplus R}}$. The box corresponding to $F'$ is shown on the right of Figure 2.

We now observe that each transition of $\mathsf{box}(F')$ is of the form $(b_W)_Y$, where $b_W \in \mathsf{box}(E \,\mathsf{sco}\, \rho)^{\mathsf{t}}$. As a result, we can define a bijection $\lambda : \mathsf{box}(F')^{\mathsf{t}} \to \mathsf{box}(F)^{\mathsf{t}}$ by setting $\lambda((b_W)_Y) = b_W$, for each transition $(b_W)_Y$ in $\mathsf{box}(F')$. Crucially, we then obtain

**Fact 7 (streamlined expression)** *The nets* $\mathsf{box}(F')$ *and* $\mathsf{box}(F)$ *are isomorphic after replacing each transition label* $b_W$ *in* $\mathsf{box}(F')$ *by* $b$. *Moreover,*

$$\lambda(\mathsf{steps}(\mathsf{box}(F'))) = \mathsf{steps}(\mathsf{box}(F))$$
$$\lambda(\mathsf{termsteps}(\mathsf{box}(F'))) = \mathsf{termsteps}(\mathsf{box}(F)) \,.$$

## 3 Interval Temporal Logic

We now provide the syntax and semantics of a small fragment of ITL, including only those constructs (basic and derived) which are used in the subsequent translation of box expressions. In particular, we assume that $V$ is a countable set of boolean variables, all such variables being (the identities of) transitions of boxes created using the $\mathsf{box}$ mapping.[4]

The formulas of the fragment of the ITL logic we need are defined by:

$$\phi ::= \mathsf{true} \mid \mathsf{flip}(v) \mid \mathsf{skipstable}(v) \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid \phi \,;\, \phi' \mid \phi^* \mid \mathsf{inf}$$

[2] Such an occurrence of $a$ is identified by the path in the syntax tree of the non-synchronised expression $E$ which corresponds to $\pi$.

[3] We assume a fixed ordering on the transitions of $\mathsf{box}(F)$ so that the enumeration of $\mathsf{trans}(a_\pi)$ is unique.

[4] The ITL syntax is as in [3] except that we now use a different set of names for the logic variables in $V$.

where $v \in V$. The intuition behind the above constructs is as follows: $\mathsf{flip}(v)$ inverts the value of a boolean variable $v$ over a unit interval while $\mathsf{skipstable}(v)$ keeps the value of $v$ over a unit interval; "$;$" is a sequential composition operator (called chop); "$*$" is an iterative version of chop; and $\mathsf{inf}$ indicates an infinite interval.

A state is a mapping which assigns values to the (boolean) variables $V$, and an interval $\sigma$ is a possibly infinite non-empty sequence of states. Its length, $|\sigma|$, is $\omega$ if $\sigma$ is infinite, and otherwise its number of states minus 1. To simplify definitions, we will denote $\sigma$ as $\langle \sigma_0, \sigma_1, \ldots, \sigma_{|\sigma|} \rangle$, where $\sigma_{|\sigma|}$ is undefined if $\sigma$ is infinite. With such a notation, $\mathsf{hd}_\sigma = \sigma_0$ and $\mathsf{tl}_\sigma = \sigma_{|\sigma|}$ (whenever $\sigma$ is finite). Moreover, for $0 \leq j \preceq k \leq |\sigma|$:

$$\sigma_{j..k} = \langle \sigma_j, \ldots, \sigma_k \rangle \quad \text{and} \quad \sigma^j = \langle \sigma_0, \ldots, \sigma_j \rangle \quad \text{and} \quad \sigma^{(j)} = \langle \sigma_j, \ldots, \sigma_{|\sigma|} \rangle$$

The meaning of formulas is given by the satisfaction relation $\models$ defined as follows:

- $\sigma \models \mathsf{true}$
- $\sigma \models \mathsf{flip}(v)$ iff $|\sigma| = 1$ and $\mathsf{hd}_\sigma(v) = \neg \mathsf{tl}_\sigma(v)$.
- $\sigma \models \mathsf{skipstable}(v)$ iff $|\sigma| = 1$ and $\mathsf{hd}_\sigma(v) = \mathsf{tl}_\sigma(v)$.
- $\sigma \models \phi \vee \phi'$ iff $\sigma \models \phi$ or $\sigma \models \phi'$.
- $\sigma \models \phi \wedge \phi'$ iff $\sigma \models \phi$ and $\sigma \models \phi'$.
- $\sigma \models \phi ; \phi'$ iff one of the following holds:
  - $|\sigma| = \omega$ and $\sigma \models \phi$.
  - there is $r \preceq |\sigma|$ such that $\sigma^r \models \phi$ and $\sigma^{(r)} \models \phi'$.
- $\sigma \models \phi^*$ iff one of the following holds:
  - $|\sigma| = 0$.
  - there are $0 = r_0 \leq r_1 \leq \cdots \leq r_{n-1} \preceq r_n = |\sigma|$ such that, for all $1 \leq l \leq n$, $\sigma_{r_{l-1}..r_l} \models \phi$.
  - $|\sigma| = \omega$ and there are infinitely many integers $0 = r_0 \leq r_1 \leq \ldots$ such that $\lim_{j \to \omega} r_j = \omega$ and for all $l \geq 1$, $\sigma_{r_{l-1}..r_l} \models \phi$.
- $\sigma \models \mathsf{inf}$ iff $|\sigma| = \omega$.

Moreover, the set of variables occurring in a formula $\phi$ will be denoted by $\mathsf{var}(\phi)$, and we will denote, for a finite set of logic variables $V' \subseteq V$:

$$\mathsf{skipstable}(V') = \begin{cases} \bigwedge_{v \in V'} \mathsf{skipstable}(v) & \text{if } V' \neq \varnothing \\ \mathsf{true} & \text{otherwise} \end{cases}$$
$$\mathsf{flip}(V') = \begin{cases} \bigvee_{v \in V'} \mathsf{flip}(v) \wedge \mathsf{skipstable}(V' \setminus \{v\}) & \text{if } V' \neq \varnothing \\ \mathsf{inf} & \text{otherwise .} \end{cases} \tag{13}$$

*Remark 1.* The logic syntax introduced above has been tailored to handle in a smooth way a subsequent translation from box expression to logic formulas. However, it is easily seen that all the non-standard constructs used above (i.e., flip, skipstable and inf) as well as true can easily be expressed in the standard ITL logic, in the following way:

$$\begin{aligned} \mathsf{true} &= v \vee \neg v \\ \mathsf{flip}(v) &= \mathsf{skip} \wedge (v = \bigcirc \neg v) \\ \mathsf{skipstable}(v) &= \mathsf{skip} \wedge (v = \bigcirc v) \\ \mathsf{inf} &= \mathsf{true}\,;\neg\mathsf{true} \end{aligned}$$

where $\bigcirc$ is the 'next' temporal operator, $\mathsf{skip} = \bigcirc\mathsf{empty}$ and $\mathsf{empty} = \neg \bigcirc \mathsf{true}$. In this way, it is possible to formulate and analyse behavioural properties of a formula resulting from the translation using the full power of ITL and the associated proof techniques and tools. □

To capture the relationship between the semantics of a box expression and a corresponding formula, with each ITL formula $\phi$ and interval $\sigma$ satisfying $\sigma \models \phi$, we associate a sequence of sets $\gamma_\sigma = \Gamma_1 \ldots \Gamma_{|\sigma|}$, where each $\Gamma_j$ is given by:

$$\Gamma_j = \{v \in \mathsf{var}(\phi) \mid \sigma_{j-1}(v) \neq \sigma_j(v)\} \ .$$

That is, each $\Gamma_j$ records all the variables which flipped their values at the point of entering the state $\sigma_j$. In this way, $\gamma_\sigma$ provides a direct interpretation of $\sigma$ in terms of sequences of steps of transitions of box nets. Formally, for any ITL formula $\phi$, we define:

$$\begin{aligned} \mathsf{steps}(\phi) &= \{\gamma_\sigma \mid \sigma \models \phi \wedge |\sigma| = \omega\} \\ \mathsf{termsteps}(\phi) &= \{\gamma_\sigma \mid \sigma \models \phi \wedge |\sigma| < \omega\} \ . \end{aligned}$$

We then obtain a number of behavioural properties of ITL formulas.

**Proposition 1.** *Let $\phi$ and $\phi'$ be two formulas with disjoint sets of variables, i.e., $\mathsf{var}(\phi) \cap \mathsf{var}(\phi') = \varnothing$. Then the following hold, where $\psi = \mathsf{skipstable}(\mathsf{var}(\phi))^*$, $\psi' = \mathsf{skipstable}(\mathsf{var}(\phi'))^*$ $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$:*

$$\begin{aligned} \mathsf{f}(\phi \wedge \psi' \ \vee \ \phi' \wedge \psi) &= \mathsf{f}(\phi) \cup \mathsf{f}(\phi') \\ \mathsf{f}(\phi \wedge \phi') &= \mathsf{f}(\phi) \parallel \mathsf{f}(\phi') \\ \mathsf{steps}((\phi \wedge \psi')\,;(\phi' \wedge \psi)) &= \mathsf{steps}(\phi) \ \cup \ \mathsf{termsteps}(\phi) \circ \mathsf{steps}(\phi') \\ \mathsf{termsteps}((\phi \wedge \psi')\,;(\phi' \wedge \psi)) &= \mathsf{termsteps}(\phi) \circ \mathsf{termsteps}(\phi') \end{aligned}$$

*Proof.* Follows directly from the basic properties of logic operators. □

**Proposition 2.** *Let $\phi_i$, for $i = 1, 2, 3$, be formulas with mutually disjoint sets of variables. Then the following hold, where $\psi_{i,j} = \mathsf{skipstable}(\mathsf{var}(\phi_i) \cup \mathsf{var}(\phi_j))^*$,*

*for* $i, j \in \{1,2,3\}$*:*

$$\begin{aligned}
&\mathsf{termsteps}((\phi_1 \wedge \psi_{2,3})\,;((\phi_2 \wedge \psi_{1,3})^*\,;(\phi_3 \wedge \psi_{1,2})) \;=\\
&\qquad \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}(\phi_2)^* \circ \mathsf{steps}(\phi_3)\\
&\mathsf{steps}((\phi_1 \wedge \psi_{2,3})\,;((\phi_2 \wedge \psi_{1,3})^*\,;(\phi_3 \wedge \psi_{1,2})) \;=\\
&\qquad \mathsf{steps}(\phi_1) \cup\\
&\qquad \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}(\phi_2)^* \circ \mathsf{steps}(\phi_2) \cup\\
&\qquad \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}(\phi_2)^* \circ \mathsf{steps}(\phi_3)\;.
\end{aligned}$$

*Proof.* We obtain, after noting that $\mathsf{var}(\phi_2^*) = \mathsf{var}(\phi_2)$ and twice using Proposition 1:

$$\begin{aligned}
&\mathsf{termsteps}((\phi_1 \wedge \psi_{2,3})\,;((\phi_2 \wedge \psi_{1,3})^*\,;(\phi_3 \wedge \psi_{1,2}))\\
&\quad = \mathsf{termsteps}((\phi_1 \wedge \psi_{2,3})\,;(((\phi_2 \wedge \psi_3)^*\,;(\phi_3 \wedge \psi_2)) \wedge \psi_1)\\
&\quad = \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}((\phi_2 \wedge \psi_3)^*\,;(\phi_3 \wedge \psi_2))\\
&\quad = \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}((\phi_2^* \wedge \psi_3)\,;(\phi_3 \wedge \mathsf{skipstable}(\mathsf{var}(\phi_2^*))^*))\\
&\quad = \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}(\phi_2^*) \circ \mathsf{termsteps}(\phi_3)\\
&\quad = \mathsf{termsteps}(\phi_1) \circ \mathsf{termsteps}(\phi_2)^* \circ \mathsf{termsteps}(\phi_3)\;.
\end{aligned}$$

The second part of the proof is similar. □

**Proposition 3.** *Let $\phi'$ be a formula obtained from an* ITL *formula $\phi$ by a consistent renaming of variables given by a bijection $\lambda$. Then $\mathsf{f}(\phi') = \lambda(\mathsf{f}(\phi))$, for $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$.*

*Proof.* Follows from the insensitivity of the semantics to the identities of logic variables used in ITL formulas. □

Finally, for every ITL formula $\phi$ and $\pi \in \Pi$, we will denote by $\pi{\cdot}\phi$ the logic formula obtained from $\phi$ by replacing each variable $a_{\pi'}$ with $a_{\pi\cdot\pi'}$.

**Proposition 4.** *Let $\pi \in \Pi$ and $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$. Then $\mathsf{f}(\pi{\cdot}\phi) = \pi{\cdot}\mathsf{f}(\phi)$.*

*Proof.* Follows from Proposition 3 and the fact that the transformation given by $\pi{\cdot}\phi$ is a consistent renaming of variables. □

## 4 From box expressions to logic formulas

To make the presentation more accessible, we will first show how to translate non-synchronised box expressions, after that we will extend the translation to the streamlined synchronised expressions, and finally we will deal with the case

of general synchronised expressions. The translation for non-synchronised expressions is as follows:

$$\begin{aligned}
\mathsf{itl}(\mathsf{stop}) &= \mathsf{inf} \\
\mathsf{itl}(a) &= \mathsf{skipstable}(a_\epsilon)^* \,;\, \mathsf{flip}(a_\epsilon) \,;\, \mathsf{skipstable}(a_\epsilon)^* \\
\mathsf{itl}(E \,;\, F) &= {;_L}{\cdot}\mathsf{itl}(E) \wedge \mathsf{skipstable}(\,{;_R}{\cdot}\mathsf{box}(F)^{\mathsf{t}})^* \;; \\
&\quad\; {;_R}{\cdot}\mathsf{itl}(F) \wedge \mathsf{skipstable}(\,{;_L}{\cdot}\mathsf{box}(E)^{\mathsf{t}})^* \\
\mathsf{itl}(E \,\square\, F) &= {\square_L}{\cdot}\mathsf{itl}(E) \wedge \mathsf{skipstable}(\,{\square_R}{\cdot}\mathsf{box}(F)^{\mathsf{t}})^* \vee \\
&\quad\; {\square_R}{\cdot}\mathsf{itl}(F) \wedge \mathsf{skipstable}(\,{\square_L}{\cdot}\mathsf{box}(E)^{\mathsf{t}})^* \\
\mathsf{itl}(E \parallel F) &= {\shortparallel_L}{\cdot}\mathsf{itl}(E) \;\wedge\; {\shortparallel_R}{\cdot}\mathsf{itl}(R) \\
\mathsf{itl}([\![E \circledast F \circledast G]\!]) &= {\circledast_L}{\cdot}\mathsf{itl}(E) \wedge \mathsf{skipstable}(\,{\circledast_M}{\cdot}\mathsf{box}(F)^{\mathsf{t}} \cup {\circledast_R}{\cdot}\mathsf{box}(G)^{\mathsf{t}})^* \;; \\
&\quad\; (\,{\circledast_M}{\cdot}\mathsf{itl}(F) \wedge \mathsf{skipstable}(\,{\circledast_L}{\cdot}\mathsf{box}(E)^{\mathsf{t}} \cup {\circledast_R}{\cdot}\mathsf{box}(G)^{\mathsf{t}})^*)^* \;; \\
&\quad\; {\circledast_R}{\cdot}\mathsf{itl}(G) \wedge \mathsf{skipstable}(\,{\circledast_L}{\cdot}\mathsf{box}(E)^{\mathsf{t}} \cup {\circledast_M}{\cdot}\mathsf{box}(F)^{\mathsf{t}})^* \,.
\end{aligned}$$

For example, if we take the box expressions $E = a \,;\, b$ and $F = c \,\square\, (a \,;\, b)$ generating the two of the boxes in Figure 1, then we obtain:

$$\begin{aligned}
\mathsf{itl}(E) &= \big(\mathsf{sfs}(a_{\,;_L}) \wedge \mathsf{skipstable}(b_{\,;_R})^*\big) \;;\; \big(\mathsf{sfs}(b_{\,;_R}) \wedge \mathsf{skipstable}(a_{\,;_L})^*\big) \\
\mathsf{itl}(F) &= \big(\mathsf{sfs}(c_{\,\square_L}) \wedge \mathsf{skipstable}(\{a_{\,\square_R\,;_L}, b_{\,\square_R\,;_R}\})^*\big) \vee \\
&\quad \Big(\big((\mathsf{sfs}(a_{\,\square_R\,;_L}) \wedge \mathsf{skipstable}(b_{\,\square_R\,;_R})^*) \;; \\
&\qquad\qquad (\mathsf{sfs}(b_{\,\square_R\,;_R}) \wedge \mathsf{skipstable}(a_{\,\square_R\,;_L})^*)\big) \wedge \mathsf{skipstable}(c_{\,\square_L})^*\Big)
\end{aligned}$$

where, for every logic variable $v \in V$:

$$\mathsf{sfs}(v) = \mathsf{skipstable}(v)^* \,;\, \mathsf{flip}(v) \,;\, \mathsf{skipstable}(v)^* \,.$$

It can easily be checked that the variables occurring in $\mathsf{itl}(E)$ are precisely the transitions of $\mathsf{box}(E)$.

**Proposition 5.** $\mathsf{var}(\mathsf{itl}(E)) = \mathsf{box}(E)^{\mathsf{t}}$, *for every non-synchronised box expression* $E$.

Crucially, however, the step semantics of a non-synchronised box expression and the corresponding ITL formula coincide.

**Theorem 1 (non-synchronised expression).** *Let* $H$ *be a non-synchronised box expression and* $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$. *Then* $\mathsf{f}(\mathsf{itl}(H)) = \mathsf{f}(\mathsf{box}(H))$.

*Proof.* Let $\phi = \mathsf{itl}(H)$ and $N = \mathsf{box}(H)$. The proof proceeds by induction of the structure of $H$.

Case 1: $H = \mathsf{stop}$. Then $\phi = \mathsf{inf}$ and $N = \mathsf{N}_{\mathsf{stop}}$. Hence we have:

$$\mathsf{steps}(\phi) = \varnothing^\infty = \mathsf{steps}(N)$$

$$\mathsf{termsteps}(\phi) = \varnothing = \mathsf{termsteps}(N) \,.$$

Case 2: $H = a$. Then $\phi = \mathsf{skipstable}(a_\epsilon)^* \,;\, \mathsf{flip}(a_\epsilon) \,;\, \mathsf{skipstable}(a_\epsilon)^*$ and $N = \mathsf{N}_a$. Hence we have:

$$\mathsf{steps}(\phi) = \varnothing^\infty \;\cup\; \varnothing^* \circ \{\{a_\epsilon\}\} \circ \varnothing^\infty = \mathsf{steps}(N)$$

$$\mathsf{termsteps}(\phi) = \varnothing^* \circ \{\{a_\epsilon\}\} \circ \varnothing^* = \mathsf{termsteps}(N)\ .$$

Case 3: $H = E \,\square\, F$. Then, by Proposition 5, we have:

$$\begin{aligned}\phi = \;& \square_L{\cdot}\mathsf{itl}(E) \wedge \mathsf{skipstable}(\,\square_R{\cdot}\mathsf{var}(\mathsf{itl}(F)))^* \;\vee \\ & \square_R{\cdot}\mathsf{itl}(F) \wedge \mathsf{skipstable}(\,\square_L{\cdot}\mathsf{var}(\mathsf{itl}(E)))^*\ .\end{aligned}$$

Hence we obtain:

$$\begin{aligned}\mathsf{f}(\phi) &= \mathsf{f}(\,\square_L{\cdot}\mathsf{itl}(E)) \;\cup \mathsf{f}(\,\square_R{\cdot}\mathsf{itl}(F)) && \text{by Prop. 1}\\ &= \square_L{\cdot}\mathsf{f}(\mathsf{itl}(E)) \;\cup\; \square_R{\cdot}\mathsf{f}(\mathsf{itl}(F)) && \text{by Prop. 4}\\ &= \square_L{\cdot}\mathsf{f}(\mathsf{box}(E)) \cup \square_R{\cdot}\mathsf{f}(\mathsf{box}(F)) && \text{by Ind. Hyp.}\\ &= \mathsf{f}(H)\ . && \text{by Fact 3}\end{aligned}$$

Case 4: $H = E \parallel F$. Then:

$$\begin{aligned}\mathsf{f}(\phi) &= \mathsf{f}(\shuffle_L{\cdot}\mathsf{itl}(E)) \;\parallel\; \mathsf{f}(\shuffle_L{\cdot}\mathsf{itl}(F)) && \text{by Prop. 1}\\ &= \shuffle_L{\cdot}\mathsf{f}(\mathsf{itl}(E)) \;\parallel\; \shuffle_L{\cdot}\mathsf{f}(\mathsf{itl}(F)) && \text{by Prop. 4}\\ &= \shuffle_L{\cdot}\mathsf{f}(\mathsf{box}(E)) \;\parallel\; \shuffle_L{\cdot}\mathsf{f}(\mathsf{box}(F)) && \text{by Ind. Hyp.}\\ &= \mathsf{f}(H)\ . && \text{by Fact 3}\end{aligned}$$

Case 5: $H = E \,;\, F$. Then, by Proposition 5, we have:

$$\begin{aligned}\phi = \;& ;_L{\cdot}\mathsf{itl}(E) \wedge \mathsf{skipstable}(\,;_R{\cdot}\mathsf{var}(\mathsf{itl}(F)))^* \;; \\ & ;_R{\cdot}\mathsf{itl}(F) \wedge \mathsf{skipstable}(\,;_L{\cdot}\mathsf{var}(\mathsf{itl}(E)))^*\ .\end{aligned}$$

Hence we obtain:

$$\begin{aligned}\mathsf{termsteps}(\phi) &= \mathsf{termsteps}(\,;_L{\cdot}\mathsf{itl}(E)) \;\circ \mathsf{termsteps}(\,;_R{\cdot}\mathsf{itl}(F)) && \text{by Prop. 1}\\ &= ;_L{\cdot}\mathsf{termsteps}(\mathsf{itl}(E)) \;\circ\; ;_R{\cdot}\mathsf{termsteps}(\mathsf{itl}(F)) && \text{by Prop. 4}\\ &= ;_L{\cdot}\mathsf{termsteps}(\mathsf{box}(E)) \circ ;_R{\cdot}\mathsf{termsteps}(\mathsf{box}(F)) && \text{by Ind. Hyp.}\\ &= \mathsf{termsteps}(H)\ . && \text{by Fact 4}\end{aligned}$$

The second part of the proof for sequence is similar.

Case 6: $H = [\![E \circledast F \circledast G]\!]$. Then, by Proposition 5, we have:

$$\begin{aligned}\phi = \;& \circledast_L{\cdot}\mathsf{itl}(E) \wedge \mathsf{skipstable}(\,\circledast_M{\cdot}\mathsf{var}(F) \cup \circledast_R{\cdot}\mathsf{var}(G))^* \;; \\ & (\,\circledast_M{\cdot}\mathsf{itl}(F) \wedge \mathsf{skipstable}(\,\circledast_L{\cdot}\mathsf{var}(E) \cup \circledast_R{\cdot}\mathsf{var}(G))^*)^* \;; \\ & \circledast_R{\cdot}\mathsf{itl}(G) \wedge \mathsf{skipstable}(\,\circledast_L{\cdot}\mathsf{var}(E) \cup \circledast_M{\cdot}\mathsf{var}(F))^*\end{aligned}$$

Hence we obtain:

$$\begin{aligned}
\mathsf{termsteps}(\phi) &= \mathsf{termsteps}(\circledast_L{\cdot}\mathsf{itl}(E)) \circ \mathsf{termsteps}(\circledast_R{\cdot}\mathsf{itl}(F))^* \\
&\quad \circ \mathsf{termsteps}(\circledast_M{\cdot}\mathsf{itl}(G)) && \text{by Prop. 2} \\
&= \circledast_L{\cdot}\mathsf{termsteps}(\mathsf{itl}(E)) \circ (\circledast_R{\cdot}\mathsf{termsteps}(\mathsf{itl}(F)))^* \\
&\quad \circ \circledast_M{\cdot}\mathsf{termsteps}(\mathsf{itl}(G)) && \text{by Prop. 4} \\
&= \circledast_L{\cdot}\mathsf{termsteps}(\mathsf{box}(E)) \circ (\circledast_R{\cdot}\mathsf{termsteps}(\mathsf{box}(F)))^* \\
&\quad \circ \circledast_M{\cdot}\mathsf{termsteps}(\mathsf{box}(G)) && \text{by Ind. Hyp.} \\
&= \mathsf{termsteps}(H)\ . && \text{by Fact 5}
\end{aligned}$$

The second part of the proof for iteration is similar. □

The above result is very strong as it basically states that the behavioural properties of non-synchronised box expressions related to the sequencing of executed actions can be re-interpreted as properties of the translated formulas, assuming that an execution of a transition is 'simulated' by a flipping of the corresponding boolean variable. Extending such a result to streamlined expressions highlights the way in which the box expression synchronisation mechanism (through merging transitions), and the ITL synchronisation mechanism (through flipping variables in different parts of a formula) match each other.

Let $F = E\,\mathsf{sco}\,\rho$ be a streamlined box expression. Then $\mathsf{itl}(F)$ is obtained from $\mathsf{itl}(E)$ by replacing each occurrence of each variable $v$ by the unique variable in $\mathsf{trans}(v)$.

**Theorem 2 (streamlined expression).** *Let $F = E\,\mathsf{sco}\,\rho$ be a streamlined box expression and $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$. Then $\mathsf{f}(\mathsf{itl}(F)) = \mathsf{f}(\mathsf{box}(F))$.*

*Proof.* By the definition of the $\mathsf{itl}$ mapping, flipping the value of any variable $v$ in $\mathsf{itl}(E)$ is due to the (unique within $\mathsf{itl}(E)$) sub-formula $\mathsf{flip}(v)$ as otherwise $v$ keeps the same value due to the presence of the $\mathsf{skipstable}(v)^*$ sub-formulas (in other words, all the variables in $\mathsf{itl}(E)$ are framed [4]).

Suppose now that $a_W$ is a variable in $\mathsf{itl}(F)$ and that $a^1_{\pi_1}, \ldots, a^k_{\pi_k}$ are the variables in $\mathsf{itl}(E)$ which in $\mathsf{itl}(F)$ are replaced by $a_W$. Then, by definition, $\pi_i | \pi_j$ for all $i \neq j$. As a consequence, there is a sub-formula $\phi \wedge \phi'$ of $\mathsf{itl}(E)$ such that $\mathsf{var}(\phi) \cap \mathsf{var}(\phi') = \varnothing$ and, without loss of generality, $a^i_{\pi_i} \in \mathsf{var}(\phi)$ and $a^j_{\pi_j} \in \mathsf{var}(\phi')$. As this observation holds for all distinct $i$ and $j$, it follows that flipping of $a_W$ in $\mathsf{itl}(F)$ must be 'agreed upon' by all the sub-formulas $\mathsf{flip}(a_W)$, each resulting from a replacement of some $a^i_{\pi_j}$ by $a_W$. The result then follows from $\mathsf{f}(\mathsf{itl}(E)) = \mathsf{f}(\mathsf{box}(E))$ (see Theorem 1) and Fact 6. □

Having discussed the case of streamlined box expressions, suppose now that $F = E\,\mathsf{sco}\,\rho$ is an arbitrary synchronised expression. Given Fact 7, we could now simply take the streamlined expression $\mathsf{stl}(F)$ defined in Section 2.6 and, after consistently renaming variables according to the bijection $\lambda$ defined in Section 2.6, obtain a generalised version of Theorem 2. Having said that, we can

proceed without pre-processing and conservatively extend the previous translation. More precisely, for any synchronised expression $F = E\,\mathsf{sco}\,\rho$ we construct $\mathsf{itl}(F)$ from $\mathsf{itl}(E)$ by replacing

- each sub-formula $\mathsf{flip}(t)$ by $\mathsf{flip}(\mathsf{trans}(t))$;
- each sub-formula $\mathsf{skipstable}(t)$ by $\mathsf{skipstable}(\mathsf{trans}(t))$; and
- each sub-formula $\mathsf{skipstable}(V')$ by $\mathsf{skipstable}(\bigcup \mathsf{trans}(V'))$.

We then obtain the main result of this paper.

**Theorem 3 (synchronised expression).** *Let $F = E\,\mathsf{sco}\,\rho$ be a synchronised box expression and $\mathsf{f} \in \{\mathsf{steps}, \mathsf{termsteps}\}$. Then $\mathsf{f}(\mathsf{itl}(F)) = \mathsf{f}(\mathsf{box}(F))$.*

*Proof.* Let us consider the streamlined expression $\mathsf{stl}(E\,\mathsf{sco}\,\rho) = E'\,\mathsf{sco}\,\rho'$ as defined in Section 2.6, and $\phi = \mathsf{itl}(E'\,\mathsf{sco}\,\rho')$ as defined for streamlined expression. Then $\mathsf{itl}(E\,\mathsf{sco}\,\rho)$ and $\phi$ are equivalent after applying a consistent renaming of variables given by the bijection $\lambda$ defined in Section 2.6. Now, we observe that if $\mathsf{trans}(a_\pi) = \{b^1_{W_1}, \ldots, b^m_{W_m}\}$ and $m \geq 2$ then the sub-formula

$$\mathsf{skipstable}(a_\pi)^* \,;\, \mathsf{flip}(a_\pi) \,;\, \mathsf{skipstable}(a_\pi)^*$$

in $\mathsf{itl}(E)$ is transformed into

$$\begin{array}{l}
\mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^m_{W_m}\})^* \,; \\
\quad (\mathsf{flip}(b^1_{W_1}) \wedge \mathsf{skipstable}(\{b^2_{W_2}, \ldots, b^m_{W_m}\}) \;\vee\; \ldots \;\vee \\
\quad \mathsf{flip}(b^m_{W_m}) \wedge \mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^{m-1}_{W_{m-1}}\})) \,; \\
\mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^m_{W_m}\})^*
\end{array}$$

within $\mathsf{itl}(F)$. This is equivalent to

$$\begin{array}{l}
\mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^m_{W_m}\})^* \,;\, \mathsf{flip}(b^1_{W_1}) \wedge \mathsf{skipstable}(\{b^2_{W_2}, \ldots, b^m_{W_m}\}) \,; \\
\qquad\qquad \mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^m_{W_m}\})^* \\
\quad \vee \;\ldots\; \vee \\
\mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^m_{W_m}\})^* \,;\, \mathsf{flip}(b^m_{W_m}) \wedge \mathsf{skipstable}(\{b^2_{W_2}, \ldots, b^m_{W_m}\}) \,; \\
\qquad\qquad \mathsf{skipstable}(\{b^1_{W_1}, \ldots, b^m_{W_m}\})^*
\end{array}$$

which in turn can be shown to be equivalent (after taking into account the correspondence given by $\lambda$) to $\mathsf{itl}(b^1_{W_1} \,\square\, \ldots \,\square\, b^m_{W_m})$. Hence the result follows from Theorem 2, Fact 7 and Proposition 3. □

To conclude, we have demonstrated that it is possible to compositionally associate a semantically equivalent ITL formula with any box expression considered in this paper.

# 5 Examples

## 5.1 Synchronisation and restriction

To illustrate action restriction as well as internal and external synchronisations, we consider a streamlined expression $F = E \,\mathsf{sco}\, \rho$, where

$$E = ((a \parallel a)\,;b) \parallel (b\,;c) \quad \text{and} \quad \rho = \{aa \mapsto A, bb \mapsto B\}\ .$$

The corresponding boxes are shown in Figure 3. In our translation, we first derive

$$\begin{aligned}\mathsf{itl}(E) = {} & ((\mathsf{sfs}(a_{\shortparallel_L\,;L\shortparallel_L}) \wedge \mathsf{sfs}(a_{\shortparallel_L\,;L\shortparallel_R})) \wedge \mathsf{skipstable}(b_{\shortparallel_L\,;R})^*)\,; \\ & \qquad (\mathsf{sfs}(b_{\shortparallel_L\,;R}) \wedge \mathsf{skipstable}(\{a_{\shortparallel_L\,;L\shortparallel_L}, a_{\shortparallel_L\,;L\shortparallel_R}\})^*) \\ & \wedge \\ & (\mathsf{sfs}(b_{\shortparallel_R\,;L}) \wedge \mathsf{skipstable}(c_{\shortparallel_R\,;R})^*)\,;(\mathsf{sfs}(c_{\shortparallel_R\,;R}) \wedge \mathsf{skipstable}(b_{\shortparallel_R\,;L})^*)\end{aligned}$$

To prepare for applying scoping we derive

$$\begin{aligned}&\mathsf{trans}(a_{\shortparallel_L\,;L\shortparallel_L}) = \mathsf{trans}(a_{\shortparallel_L\,;L\shortparallel_R}) = \{A_{\shortparallel_L\,;L\shortparallel_L,\shortparallel_L\,;L\shortparallel_R}\} = \{\alpha\} \\ &\mathsf{trans}(b_{\shortparallel_L\,;R}) \quad = \mathsf{trans}(b_{\shortparallel_R\,;L}) \quad = \{B_{\shortparallel_L\,;R,\shortparallel_R\,;L}\} \qquad = \{\beta\} \\ &\mathsf{trans}(c_{\shortparallel_R\,;R}) \quad = \varnothing\end{aligned}$$

which (after eliminating $\mathsf{true}$ in conjunctions) leads to

$$\begin{aligned}\mathsf{itl}(F) = {} & ((\mathsf{sfs}(\alpha) \wedge \mathsf{sfs}(\alpha)) \wedge \mathsf{skipstable}(\beta)^*)\,;(\mathsf{sfs}(\beta) \wedge \mathsf{skipstable}(\alpha)^*) \\ & \wedge \\ & \mathsf{sfs}(\beta)\,;(\mathsf{inf} \wedge \mathsf{skipstable}(\beta)^*)\end{aligned}$$

and so, in an equivalent form, we obtain:

$$\begin{aligned}\mathsf{itl}(F) = {} & (\mathsf{sfs}(\alpha) \wedge \mathsf{skipstable}(\beta)^*)\,;(\mathsf{sfs}(\beta) \wedge \mathsf{skipstable}(\alpha)^*) \\ & \wedge \\ & \mathsf{sfs}(\beta)\,;(\mathsf{inf} \wedge \mathsf{skipstable}(\beta)^*)\end{aligned}$$

Note that

$$\begin{aligned}\mathsf{steps}(F) &= \varnothing^\infty \ \cup\ \varnothing^* \circ \{\alpha\} \circ \varnothing^\infty \ \cup\ \varnothing^* \circ \{\alpha\} \circ \varnothing^* \circ \{\beta\} \circ \varnothing^\infty \\ \mathsf{termsteps}(F) &= \varnothing\ .\end{aligned}$$

## 5.2 Parallel composition and choice

To illustrate choice and parallel composition, we consider a streamlined expression $F = E \,\mathsf{sco}\, \rho$, where

$$E = (a \,\square\, b) \parallel b \quad \text{and} \quad \rho = \{a \mapsto a, bb \mapsto b\}\ .$$

The corresponding boxes are shown in Figure 4. In our translation, we first derive

$$\begin{aligned}\mathsf{itl}(E) = {} & \mathsf{sfs}(a_{\shortparallel_L\,\square_L}) \wedge \mathsf{skipstable}(b_{\shortparallel_L\,\square_R})^* \ \vee\ \mathsf{sfs}(b_{\shortparallel_L\,\square_R}) \wedge \mathsf{skipstable}(a_{\shortparallel_L\,\square_L})^* \\ & \wedge \\ & \mathsf{sfs}(b_{\shortparallel_R})\ .\end{aligned}$$

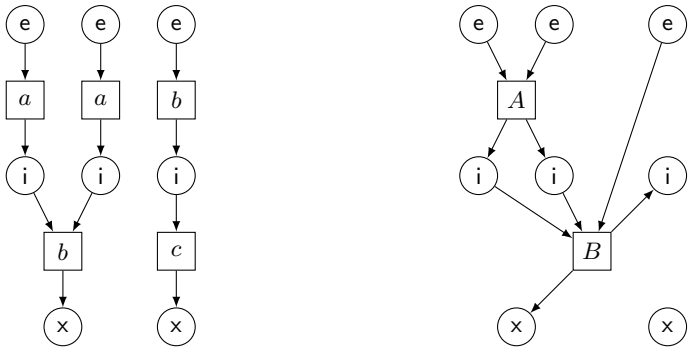


**Fig. 3.** Boxes of $E = ((a \parallel a)\,;b) \parallel (b\,;c)$ and $F = E\,\mathsf{sco}\,\{aa \mapsto A, bb \mapsto B\}$.

To prepare for applying scoping we derive

$$\begin{aligned} \mathsf{trans}(a_{\shortparallel_L \,\square_L}) &= \{a_{\shortparallel_L \,\square_L}\} = \{\alpha\} \\ \mathsf{trans}(b_{\shortparallel_L \,\square_R}) = \mathsf{trans}(b_{\shortparallel_R}) &= \{b_{\shortparallel_L \,\square_R, \shortparallel_R}\} = \{\beta\} \end{aligned}$$

which (after eliminating $\mathsf{true}$ in conjunctions) leads to

$$\mathsf{itl}(F) \;=\; \big(\mathsf{sfs}(\alpha) \wedge \mathsf{skipstable}(\beta)^* \;\vee\; \mathsf{sfs}(\beta) \wedge \mathsf{skipstable}(\alpha)^*\big) \;\wedge\; \mathsf{sfs}(\beta)\;.$$

Hence, in an equivalent form, we obtain:

$$\mathsf{itl}(F) \;=\; \mathsf{sfs}(\alpha) \wedge \mathsf{skipstable}(\beta)^* \wedge \mathsf{inf} \;\vee\; \mathsf{sfs}(\beta) \wedge \mathsf{skipstable}(\alpha)^*\;.$$

Note that

$$\begin{aligned} \mathsf{steps}(F) &= \varnothing^\infty \;\cup\; \varnothing^* \circ \{\alpha\} \circ \varnothing^\infty \;\cup\; \varnothing^* \circ \{\beta\} \circ \varnothing^\infty \\ \mathsf{termsteps}(F) &= \varnothing^* \circ \{\beta\} \circ \varnothing^*\;. \end{aligned}$$

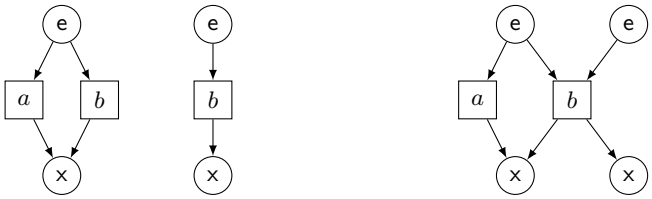


**Fig. 4.** Boxes of $E = (a \,\square\, b) \parallel b$ and $F = E\,\mathsf{sco}\,\{a \mapsto a, bb \mapsto b\}$.

### 5.3 Iteration and scoping

To illustrate synchronisation inside iteration, we consider a non-streamlined expression $F = E \,\mathsf{sco}\, \{a \mapsto G, bc \mapsto A, c \mapsto C\}$ where

$$E = [\![ a \circledast (b \parallel c) \circledast d ]\!] .$$

Note that boxes corresponding to $E$ and $F$ are shown in Figure 2. We first derive

$$\begin{aligned}\mathsf{itl}(E) = {} & \mathsf{sfs}(a_{\circledast_L}) \wedge \mathsf{skipstable}(\{b_{\circledast_{M\shortparallel L}}, c_{\circledast_{M\shortparallel R}}, d_{\circledast_R}\})^* \;; \\ & (\mathsf{sfs}(b_{\circledast_{M\shortparallel L}}) \wedge \mathsf{sfs}(c_{\circledast_{M\shortparallel R}}) \wedge \mathsf{skipstable}(\{a_{\circledast_L}, d_{\circledast_R}\})^*)^* \;; \\ & \mathsf{sfs}(d_{\circledast_R}) \wedge \mathsf{skipstable}(\{a_{\circledast_L}, b_{\circledast_{M\shortparallel L}}, c_{\circledast_{M\shortparallel R}}\})^* .\end{aligned}$$

To prepare for applying scoping we derive

$$\begin{aligned}&\mathsf{trans}(a_{\circledast_L}) = \{G_{\circledast_L}\} = \{\gamma\} \quad && \mathsf{trans}(b_{\circledast_{M\shortparallel L}}) = \{A_{\circledast_{M\shortparallel L}}, \circledast_{M\shortparallel R}\} && = \{\alpha\} \\ &\mathsf{trans}(d_{\circledast_R}) = \varnothing && \mathsf{trans}(c_{\circledast_{M\shortparallel R}}) = \{A_{\circledast_{M\shortparallel L}}, \circledast_{M\shortparallel R}, C_{\circledast_{M\shortparallel R}}\} && = \{\alpha, \zeta\}\end{aligned}$$

which leads to

$$\begin{aligned}\mathsf{itl}(F) = {} & \mathsf{sfs}(\gamma) \wedge \mathsf{skipstable}(\{\alpha, \zeta\})^* \;; \\ & (\mathsf{sfs}(\alpha) \wedge \mathsf{sfs}(\{\alpha, \zeta\}) \wedge \mathsf{skipstable}(\gamma)^*)^* \;; \\ & \mathsf{inf} \wedge \mathsf{skipstable}(\{\gamma, \alpha, \zeta\})^* .\end{aligned}$$

Hence, in an equivalent form, we obtain:

$$\begin{aligned}\mathsf{itl}(F) = {} & \mathsf{sfs}(\gamma) \wedge \mathsf{skipstable}(\{\alpha, \zeta\})^* \;; \\ & \big(\mathsf{sfs}(\alpha) \wedge \\ & \quad \big(\mathsf{skipstable}(\{\alpha, \zeta\})^* \,;\, \mathsf{flip}(\alpha) \wedge \mathsf{skipstable}(\zeta) \,;\, \mathsf{skipstable}(\{\alpha, \zeta\})^* \\ & \quad \vee \\ & \quad \mathsf{skipstable}(\{\alpha, \zeta\})^* \,;\, \mathsf{flip}(\zeta) \wedge \mathsf{skipstable}(\alpha) \,;\, \mathsf{skipstable}(\{\alpha, \zeta\})^*\big) \\ & \wedge \mathsf{skipstable}(\gamma)^*\big)^* \;; \\ & \mathsf{inf} \wedge \mathsf{skipstable}(\{\gamma, \alpha, \zeta\})^* .\end{aligned}$$

Note that

$$\begin{aligned}\mathsf{steps}(F) = {} & \varnothing^\infty \;\cup \\ & \varnothing^* \circ \{\gamma\} \circ \varnothing^\infty \;\cup \\ & \varnothing^* \circ \{\gamma\} \circ (\varnothing^* \circ \{\alpha\})^* \varnothing^\infty \;\cup \\ & \varnothing^* \circ \{\gamma\} \circ (\varnothing^* \circ \{\alpha\})^* \circ \varnothing^* \circ \{\zeta\} \circ \varnothing^\infty \\ \mathsf{termsteps}(F) = {} & \varnothing .\end{aligned}$$

## 6 Conclusions

In the past, different kinds of logics have been used as formalism for expressing correctness properties of systems specified using Petri nets. When it comes to

the relationship between logics and Petri net, we feel that the work on the connections between linear logic [8] and Place Transition nets was the closest one. However, the main concern there was the handling of multiple token occurrences in net places whereas here nets can hold at most two tokens in a single place ever. Another way in which logics and Petri nets were discussed was reported in [17] which provided a characterisation of Petri net languages in terms of second-order logical formulas.

The results presented in this paper demonstrate that one can develop a very close structural connection between BA and ITL. It is therefore important to further investigate the extent to which such a connection could be generalised and exploited. In particular, we plan to investigate what is the subset of ITL which can be modelled by BA. A longer time goal is the development of a hybrid verification methodology combining ITL and BA techniques. For example, sequential algorithms and infinite data structures could be treated by ITL techniques [2, 9, 14], while intensive parallel or communicating aspects of systems could be treated by net unfoldings [7, 10] or other Petri net techniques [18].

**Acknowledgement**

This research was supported by the 973 Program Grant 2010CB328102, NSFC Grant 61133001, ANR SYNBIOTIC and EPSRC GAELS and UNCOVER projects.

## References

1. E.Best, R.Devillers and M.Koutny: Petri Net Algebra. Monographs in Theoretical Computer Science, Springer (2001)
2. A.Cau and H.Zedan: Refining Interval Temporal Logic Specifications. Lecture Notes in Computer Science 1231 (1997) 79–94
3. Z.Duan, H.Klaudel, M.Koutny: ITL Semantics of Composite Petri Nets. The Journal of Logic and Algebraic Programming (2012) http://dx.doi.org/10.1016/j.jlap.2012.12.001
4. Z.Duan, X.Yang and M.Koutny: Framed Temporal Logic Programming. Science of Computer Programming 70 (2008) 31–61
5. J.Desel and G.Juhás: "What Is a Petri Net?" Lecture Notes in Computer Science 2128 (2001) 1–25
6. E.A.Emerson: Temporal and Modal Logic. In: Handbook of Theoretical Computer Science, Elsevier Science (1990) 995–1072
7. J.Esparza: Model Checking Using Net Unfoldings. Science of Computer Programming 23 (1994) 151–195
8. J.-Y.Girard: Linear Logic. Theoretical Computer Science 50 (1987) 1–102
9. H.Janicke, A.Cau, F.Siewe and H.Zedan: Dynamic Access Control Policies: Specification and Verification. The Computer Journal (2012) doi:10.1093/comjnl/bxs102
10. V.Khomenko and M.Koutny: Towards An Efficient Algorithm for Unfolding Petri Nets. Lecture Notes in Computer Science 2154 (2001) 366–380
11. Z.Manna and A.Pnueli: Verification of Concurrent Programs: The Temporal Framework. In: The Correctness Problem in Computer Science, Academic Press (1981) 215–273

12. R.Milner: A Calculus of Communicating Systems. Springer Verlag (1980)
13. B.Moszkowski: Compositional Reasoning About Projected and Infinite Time. Proceedings of ICECCS (1995) 238–245
14. B.Moszkowski: Executing Temporal Logic Programs. Cambridge University Press (1986)
15. B.Moszkowski and Z.Manna: Reasoning in Interval Temporal Logic. Lecture Notes in Computer Science 164 (1984) 371–382
16. T.Murata: Petri Nets: properties, Analysis and Applications. Proceedings of the IEEE 77 (1989) 541–80
17. M.Parigot and E.Pelz: A Logical Approach of Petri Net Languages. Theoretical Computer Science 39 (1985) 155–169
18. Petri net tools homepage: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html
19. M.Silva, E.Teruel and J.-M.Colom: Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. Lecture Notes in Computer Science 1491 (1998) 309–373
20. A.Valmari: Stubborn Sets for Reduced State Space Generation. Lecture Notes in Computer Science 483 (1989) 491–515