University of Newcastle upon Tyne
School of Computing Science

An Algebra of Petri Nets with Arc-based Timing Restrictions

by
Apostolos Niaouris

PhD Thesis

December 2005

Contents

Acknowledgements

To begin with, I would like to thank my supervisor, Maciej Koutny, for inspiring me to work on Petri nets and process algebras. Actually, I cannot find the right words to express my gratitude for him and I am pretty sure that without his constant guidance and support the work presented in this thesis would not be possible. I am also very grateful to the members of my research committee, Alex Yakovlev and Victor Khomenko for our fruitful discussions and for their comments on the improvement of this thesis. I want to express my gratitude to the School of Computing Science at Newcastle and EPSRC for giving me the opportunity and the necessary funds to complete this research. Furthermore, I am grateful to my parents Nikolaos and Evanthia for their unconditional love and continuous moral and financial support. Finally, I want to thank, from the bottom of my heart, my partner in life Tina for always being there for me.

Abstract

Human beings from the moment they understood the power of their brain tried to create things to make their life easier and satisfy their needs either physical or mental. Inventions became more and more complicated, covering almost every aspect of human life and satisfying the never ending human curiosity. One of the reasons for this complexity is that an increasing number of systems exhibit concurrency. The development of concurrent systems is generally challenging since it is more difficult to fully understand their exact behaviour. In this thesis we present and investigate two of the most widely used and well studied theories to capture concurrent behaviour. Based on the results of PBC, we develop two algebras, one based on term re-writing and the other on Petri nets, aimed at the specification and analysis of concurrent systems with timing information. The former is based on process expressions (at-expressions) and employs a set of SOS rules providing their operational semantics. The latter is based on a class of Petri nets with time restrictions associated with their arcs, called at-boxes, and the corresponding transition firing rule. We relate the two algebras through a compositionally defined mapping which for a given at-expression returns an at-box with behaviourally equivalent transition system. The resulting framework consisting of the two algebras is called the Timed-Arc Petri Box Calculus, or *atPBC*.

Introduction

Human beings from the moment they understood the power of their brain tried to create things to make their life easier and satisfy their needs either physical or mental. They were building all sort of contraptions, starting for example from simple and common nowadays but really fundamental things like the lever or the wheel. But the human mind did not stop when the need to raise something with a lever was satisfied or when it became possible to carry big piles of stone in order to build a shelter. Inventions became more and more complicated, covering almost every aspect of human life and satisfying the never ending human curiosity. It is impossible to measure the increase in complexity of systems from the simple 'sort of round' wheel till the latest space exploration shuttle, the state of the art electronic microscope possible to reach subatomic levels or even the internet. One of the reasons for the increased complexity is that the number of systems that work concurrently is increasing extremely fast.

The meaning of the term 'concurrently' is that the system can perform a number of its specified actions at the same time. An external observer may not be able to distinguish any particular order of these actions. Amongst other reasons that led to concurrent systems' design is the need for extra speed. In an oversimplified example, let us assume that there are four numbers that need to be added and there is a calculator with one 'computing' element and another one with two 'computing' elements. The first one will take the first two numbers add them together, then add the third to the existing sum and finally add the forth number to the sum and complete the computation. It is obvious that there is a specific order of events in the first calculator and three steps are necessary to complete the computation. The second, more advanced, calculator can take the first two numbers in its first processing unit, the other two in the second processing unit, compute the two sums and then add these two sums. Again, three computations are necessary but the first two computations can happen in any order or even in parallel. Let us consider that each addition consumes one time unit. In the latter case, the elapsed time for the complete computation will be two time units instead of three. In an optimum situation (constantly feeding data and no dependencies between computations), a computer that uses two microprocessors in parallel may be able to finish its computations in half time compared to a computer that uses only one microprocessor. On the other hand, extra care is need from the modeler when building such concurrent systems to compensate the massive increase in complexity. It is generally more difficult to

fully understand the exact behaviour of a concurrent system, even in a relatively simple one, since there exists no specific order in its actions. Therefore, avoiding bugs in system's design is a challenging process and several techniques and tools, for example [19,29], have been introduced in order to identify and capture these errors.

The need to build correct and reliable concurrent systems is one of the main reasons that fueled this research. It can be understood from this informal presentation that concurrency theory is one of the most challenging and open areas of research in computing science. In the past years, several theories have been introduced in order to capture concurrent behaviour and computation. Two of the most widely used and well studied are *process algebras* and *Petri nets*. Process algebras, e.g., ACP [4], CCS [45,46] and CSP[32], provide a formal framework for dealing with large and complex concurrent computing systems by employing specific operators corresponding to commonly used programming constructs. The way of representing a system's structure is given through suitably defined set of process expressions, and their behaviour is typically captured by a (structured) set of sequences of executed actions. Furthermore, since process algebras are compositional by definition it is possible to compose large systems from smaller ones in a structured way. A variety of logics is present in process algebras helping the modeler to reason about the properties of the system. Finally, process algebras come with a wide selection of algebraic laws which can be used to prove correctness with respect to the specification. On the other hand, Petri nets [47,57] represent a natural framework for capturing concurrent behaviours. There is a clear distinction between (local) states and changes of states (local actions) through the distinction between places and transitions. The global state of the system is not shown explicitly but it can be derived from their local counterparts. Although formal, they support a graphical representation of concurrent systems which is simpler to understand compared to other approaches and therefore Petri nets can be easily adopted by practitioners. Petri nets are based on the theory of partial orders and as a result it is possible to capture explicit asynchrony. For example, the simultaneous execution of several actions can be easily modeled and there is no need for interleaving semantics. Finally, since they are closely related to graph theory and linear algebra, they provide an additional means to verify the correctness of the modeled system efficiently and a way of expressing properties related to causality and concurrency in system behaviour.

We can get back to the simple calculator example to visualise some of the advantages of Petri nets and understand the increased complexity in a system's behaviour. In figure 1 we have the Petri nets corresponding to the two calculators together with their reachability graphs. It can be seen in the reachability graphs that there is only one execution scenario for the first calculator but, even though we are not considering alternative feeding of the four numbers, there are three different scenarios (every scenario is giving the same result) for the calculator with the two processing elements.

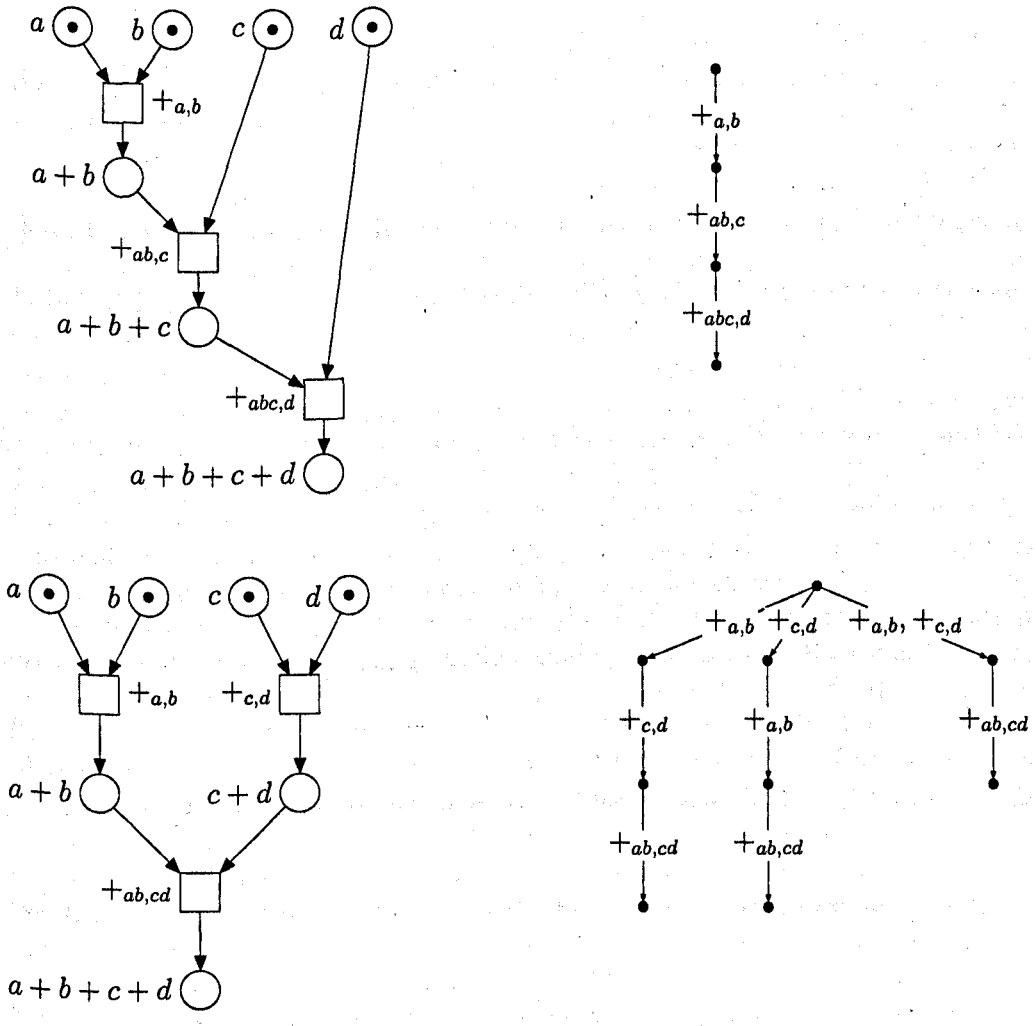


Figure 1: Petri nets corresponding to two different calculators and their reachability graphs.

These two kinds of formalisms treat the structure and semantics of concurrent systems in different ways, which in the past meant that it was almost impossible to take full advantage of their relative advantages when used in isolation. Several approaches have been proposed in order to resolve this situation by providing a translation of process algebras into nets, for example in [12, 13, 21, 22, 25–27, 65]. This list is not complete and a more precise list of the previous research in the area can be found in [18]. This thesis is based on a different approach presented in the Box Algebra [8–10, 38] and its precursor, the Petri Box Calculus (PBC) [7]. To a significant extent, these two research proposals addressed the problem at hand. Both models provided a framework where Petri nets and process algebras could co-exist, and thus established a bridge between these two approaches.

A large number of real world applications can be considered where most actions are associated with some kind of time restrictions. In standard Petri nets, there was no consideration for time variables. As a result, such an extension was necessary to accurately model this type of concurrent systems. Furthermore, since its conception, the original PBC has been also extended towards the direction of timing restrictions. In particular, it was necessary to cover concurrent systems with timing restrictions [37,39], where the timing restrictions were associated with transitions, effectively specifying for how long an enabled action (or transition) can delay/prolong its execution as well as what a minimum delay or execution time is. Another way in which timing assumption could be introduced is to associate clocks (or age) with the resources (or tokens). More precisely, one can specify how old/young a given resource consumed by an action must be. This approach has been extensively studied in the past, see, e.g., [1,15,51], both as a model for dealing with complex concurrent systems such as communication protocols, and as a framework for verifying their properties. It is precisely this kind of time modeling which has been adopted in this thesis.

We will introduce and investigate two different models for the specification of concurrent systems including explicit timing information. Both models have an algebraic structure based on operators present in the standard PBC. The first algebra is based on process expressions, called at-expressions, and a system of rewriting rules providing structural operational semantics of at-expressions in the style of [54]. The second algebra is based on a class of Petri nets with arc-based timing restrictions, called at-boxes, and their execution rules. This means, in particular, that: (i) each arc from a place p to a transition is given two time bounds, e and l, representing the *earliest consuming time* and the *latest consuming time*, respectively, for a token which has arrived at place p; (ii) the local clock of a token is started at the very moment it has been created; and (iii) time is discrete. It is important to point out that property (i) suits particularly well the intended compositional setting we are aiming at since the handshake synchronisation of two transitions basically amounts to gluing them together, and no special consideration of their timing restrictions is needed. On the other hand, gluing two transitions in the other time framework we mentioned requires combining their timing intervals which can be done in several different ways.

The two algebras are related through a compositionally defined mapping which, for at-expression returns a corresponding at-box (its denotational semantics). The main result is that the denotational and operational semantics of an at-expression are behaviourally equivalent. The resulting framework, first reported in [48] and further developed in [49], consisting of two algebras is called the Timed-Arc Petri Box Calculus, or *atPBC*.

Although there will be a concise presentation of the basic concepts of PBC and the Box Algebra, throughout this thesis we assume that the reader is somehow familiar with the work presented in [7–10, 38] on which the compositional treatment of nets is based.

Organisation of the Thesis

The thesis is organized as follows.

Chapter 1 provides the basic notions concerning Petri nets and a presentation of basic concepts of Petri Box Calculus and the Box Algebra.

Chapter 2 provides a presentation of possible time extensions of Petri nets and the existing research achievements on the combination of this type of Petri nets with process algebras.

Chapter 3 describes the syntax of atPBC and the operational semantics of process expressions corresponding to at-boxes.

Chapter 4 extends the box algebra to at-boxes by the definition of a compositional mapping from at-expressions to at-boxes.

Chapter 5 introduces a new type of timed-arc Petri nets together with a translation from at-expressions to this new type of boxes.

Chapter 6 presents the main results of this thesis which have to do with the behavioural relationships between expressions and the two different type of timed-arc Petri nets.

Chapter 7 describes several possible extensions of the proposed framework that can increase the modeling power of atPBC.

Chapter 1

Basic notions

In this chapter, we present the basic notions which will be used throughout the thesis.

1.1 Multisets

Throughout this thesis, $\mathbb{N}$ denotes the set of non-negative integers, $\mathbb{Z}$ denotes the set of integers, $\mathbb{N}^\infty \stackrel{\text{df}}{=} \mathbb{N} \cup \{\infty\}$ and $\mathbb{Z}^\infty \stackrel{\text{df}}{=} \mathbb{Z} \cup \{\infty\}$. A *multiset* over a set X is a function $\mu : X \to \mathbb{N}$. Note that any subset of X may be viewed (through its characteristic function) as a multiset over X. We denote $x \in \mu$ if $\mu(x) \geq 1$, and for two multisets over X, μ and μ', we write $\mu \leq \mu'$ if $\mu(x) \leq \mu'(x)$ for all $x \in X$. We will use $\varnothing$ to denote the *empty multiset* defined as $\varnothing(x) \stackrel{\text{df}}{=} 0$, for all $x \in X$. Moreover, a finite multiset may be represented by explicitly listing its elements between the $\{\ldots\}$ brackets. For example $\{a, a, b\}$ denotes a multiset μ such that, for every $x \in X$,

$$\mu(x) = \begin{cases} 2 & \text{if } x = a \\ 1 & \text{if } x = b \\ 0 & \text{otherwise.} \end{cases}$$

The *sum of two multisets* μ' and μ'' over X is given by $(\mu' + \mu'')(x) \stackrel{\text{df}}{=} \mu'(x) + \mu''(x)$, the *difference* by $(\mu' - \mu'')(x) \stackrel{\text{df}}{=} \max\{0, \mu'(x) - \mu''(x)\}$, and the *intersection* by $(\mu' \cap \mu'')(x) \stackrel{\text{df}}{=} \min\{\mu'(x), \mu''(x)\}$, for all $x \in X$. A multiset μ is *finite* if there are finitely many $x \in X$ such that $\mu(x) \geq 1$. In such a case, the *cardinality* of μ is defined as $|\mu| \stackrel{\text{df}}{=} \sum_{x \in X} \mu(x)$. The set of all finite multisets over a set Z will be denoted by $\mathsf{mult}(Z)$.

The notation $\{P(x) \mid x \in \mu\}$, where μ is a multiset and $P(x)$ is constructed from $x \in X$ and will be used to denote the multiset μ' such that

$$\mu'(y) \stackrel{\text{df}}{=} \sum_{x \in X \wedge P(x) = y} \mu(x) \cdot y \,,$$

where $\mu(x) \cdot y$ is the multiset consisting of exactly $\mu(x)$ copies of y. Furthermore, for a mapping $h : X \to Y$ and a multiset μ over X, we denote $h\{\mu\} \stackrel{\text{df}}{=} \{h(x) \mid x \in \mu\}$. For example, $\{x^2+1 \mid x \in \{-1,0,0,1\}\} = \{1,1,2,2\}$.

If $f : X \to \mathbb{Z}$ is a function and μ is a multiset over X then

$$\sum_{x \in \mu} f(x) \stackrel{\text{df}}{=} \sum_{x \in X} \mu(x) \cdot f(x)$$

if the latter sum is defined.

1.2 Elements of an algebra of Petri nets

We will introduce Petri nets as in [17,47,57], and present their semantics as in [8, 10] as necessary, choosing from concurrency semantics such as: step semantics [24], trace semantics [42], process semantics [6, 28], or partial word semantics [30, 64, 66].

Furthermore, there will be a concise description of the general compositionality mechanism for combining nets. Composition of nets will be driven by labellings of both places and transitions. Such labellings indicate the border (interface) between a net and its (potential) surroundings, the resulting combinable objects will be called *boxes*.

1.2.1 Labelled nets and their semantics

A marked net with place and transition labels (*labelled net*, for short) is a tuple

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

such that: P and T are disjoint sets of respectively *places* and *transitions*; W is a *weight function* from the set $(P \times T) \cup (T \times P)$ to the set of natural numbers $\mathbb{N}$; λ is a *labelling* function for places and transitions such that $\lambda(s) \in \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$, for every place $p \in P$, and $\lambda(t)$ is a relabelling ϱ of the form

$$\varrho \subseteq \mathsf{mult}(\mathcal{A}) \times \mathcal{A}$$

such that $(\varnothing, \alpha) \in \varrho$ if and only if $\varrho = \{(\varnothing, \alpha)\}$ for every transition $t \in T$. Moreover, $\mathcal{A}$ is a fixed set of *communication actions* (serving as transition labels) such that for every $a \in \mathcal{A}$, there exists its *conjugate*, $\widehat{a} \in \mathcal{A}$, satisfying $a \neq \widehat{a}$ and $\widehat{\widehat{a}} = a$. Also, there is a *silent* (or internal) action $\imath \notin \mathcal{A}$. In the algebra of nets (as well as in the process algebra), it will be assumed that a *synchronisation* of two conjugate communication actions always gives rise to the silent action $\imath$. Finally M is a *marking*, i.e., a multiset over P.

Nets can be represented as directed graphs. We adopt the standard rules about drawing nets, viz. places are represented as circles, transitions as boxes, the weight function by arcs with the indicated weight (we do not draw arcs

whose weight is 0, and we do not indicate the weight if it is 1), and markings are shown by placing tokens within circles. To avoid ambiguity, we will sometimes decorate the various components of Σ with the index Σ; thus, T_Σ denotes the set of transitions of Σ, etc. A net is *finite* if both P and T are finite sets.

If the labelling of a place p in a labelled net Σ is e then p is an *entry* place, if i then p is an *internal* place, and if x then p is an *exit* place. By convention, $^\circ\Sigma$, Σ° and $\ddot{\Sigma}$ denote respectively the entry, exit and internal places of Σ. For every place (transition) x, we use $^\bullet x$ to denote is *pre-set*, i.e., the set of all transitions (places) y such that there is an arc from y to x, $W(y,x) > 0$. The *post-set* $x^\bullet$ is defined in a similar way. The pre- and post-set notation extends in the usual way to sets R of places and transitions, e.g., $^\bullet R = \bigcup\{^\bullet r \mid r \in R\}$. In what follows, all nets are assumed to be T-*restricted*, i.e., the pre- and post-sets of each transition are non-empty.

A labelled net Σ is *ex-restricted* if there is at least one entry and at least one exit place, $^\circ\Sigma \neq \varnothing \neq \Sigma^\circ$. Σ is *e-directed* (*x-directed*) if the entry (respectively, exit) places are free from incoming (respectively, outgoing) arcs, i.e., $^\bullet(^\circ\Sigma) = \varnothing$ (respectively, $(\Sigma^\circ)^\bullet = \varnothing$). Σ is *ex-directed* if it is both e-directed and x-directed. Σ is *marked* if $M_\Sigma \neq \varnothing$, and *unmarked* otherwise.

We will use three explicit ways of modifying the marking of $\Sigma = (P, T, W, \lambda, M_\Sigma)$. We define $\lfloor\Sigma\rfloor$ as $(P, T, W, \lambda, \varnothing)$; typically, this operation is used when $M_\Sigma \neq \varnothing$, since it erases all tokens. Moreover, we define $\overline{\Sigma}$ and $\underline{\Sigma}$ as, respectively, $(P, T, W, \lambda, {}^\circ\Sigma)$ and $(P, T, W, \lambda, \Sigma^\circ)$. We will call $^\circ\Sigma$ the *entry marking*, and Σ° the *exit marking* of Σ. Note also that $\lfloor.\rfloor$, $\overline{(.)}$ and $\underline{(.)}$ are syntactic operations having nothing to do with derivability (reachability) in the sense of the step sequence semantics defined next.

Step sequence semantics

We adopt finite step sequence semantics for a labelled net $\Sigma = (P, T, W, \lambda, M)$, in order to capture the potential concurrency in the behaviour of the system modelled by Σ. A finite multiset of transitions U, called a *step*, is *enabled* by Σ if for every place $p \in P$,

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

We denote this by $\Sigma[U\rangle$, or $M[U\rangle$ if the net is understood from the context. An enabled step U can be *executed*, leading to a follower marking M' defined, for every place $p \in P$, by

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

We denote this by $M[U\rangle M'$ or $\Sigma[U\rangle\Theta$, where Θ is the labelled net (P, T, W, λ, M'). Transition labelling may be extended to steps, through the formula

$$\lambda(U) \;=\; \sum_{t \in U} U(t) \cdot \{\lambda(t)\} \;\in\; \mathsf{mult}(\mathcal{A})\;.$$

Although we will use the same term 'step' to refer both to a finite set of transitions and to a finite multiset of labels, it will always be clear from the context which one is meant. The notation for label based steps will be $\Sigma\ [\Gamma\rangle_{\mathsf{lab}}\ \Theta$, where $\Gamma = \lambda(U)$.

A *finite step sequence* of Σ is a finite (possibly empty) sequence $\sigma = U_1 \ldots U_k$ of steps for which there are labelled nets $\Sigma_0, \ldots, \Sigma_k$ such that $\Sigma = \Sigma_0$ and for every $1 \leq i \leq k$, $\Sigma_{i-1}[U_i\rangle\Sigma_i$. Depending on the need, we shall then use one of the following notations:

$$\Sigma[\sigma\rangle\Sigma_k \qquad M_\Sigma[\sigma\rangle M_{\Sigma_k} \qquad \Sigma_k \in [\Sigma\rangle \qquad M_{\Sigma_k} \in [M_\Sigma\rangle\ .$$

Moreover, the marking M_{Σ_k} will be called *reachable* from M_Σ, and Σ_k *derivable* from Σ. The empty step will always be enabled, but it can be ignored when one considers a step sequence, since the empty step always relates a net to itself, i.e., $\Sigma[\varnothing\rangle\Theta$ if and only if $\Sigma = \Theta$.

Safeness, cleanness and exclusiveness

The marking M of Σ is *safe* if for all $p \in P$, $M(p) \in \{0, 1\}$. As already indicated, a safe marking can and will often be identified with the set of places to which it assigns 1. A marking is *clean* if it is not a proper super-multiset of ${}^\circ\Sigma$ or Σ°, i.e., if ${}^\circ\Sigma \subseteq M$ or $\Sigma^\circ \subseteq M$ implies ${}^\circ\Sigma = M$ or $\Sigma^\circ = M$, respectively. A marking M is *ex-exclusive* if it does not simultaneously mark an entry place and an exit place, i.e., if $M \cap {}^\circ\Sigma = \varnothing$ or $M \cap \Sigma^\circ = \varnothing$. A labelled net is called *safe* (respectively, *clean*) if all its reachable markings are safe (respectively, clean).

Partial order semantics

To model partial order behaviours of nets and expressions, we use Mazurkiewicz traces [42].

Let A be a set and $\mathsf{ind} \subseteq A \times A$ be an irreflexive symmetric relation on A. The idea here is that A represents the set of all possible events in a concurrent system, and ind is an *independence* relation which asserts which events can be executed concurrently. With every sequence $\sigma = A_1 \ldots A_k$, where each A_i is a finite subset of A such that $(a, b) \in \mathsf{ind}$ for all distinct $a, b \in A_i$, we associate a partial order (poset), denoted by $\mathsf{poset}_{\mathsf{ind}}(\sigma)$, in the following way.

The set of *event occurrences* of σ, occ_σ, comprises all pairs $(a, l) \in A \times \mathbb{N}$ such that $a \in A_1 \cup \ldots \cup A_k$ and l is less or equal to the number of times a occurs within σ. Moreover, we denote by $\mathsf{idx}_{(a,l)}$ the index m such that A_m contains the l-th occurrence of a in σ, and define a *precedence* relation on occ_σ, $\prec_\sigma$, by stipulating that $(a, l) \prec_\sigma (b, n)$ whenever $(a, b) \notin \mathsf{ind}$ and $\mathsf{idx}_{(a,l)} < \mathsf{idx}_{(b,n)}$. Then $\mathsf{poset}_{\mathsf{ind}}(\sigma) = (\mathsf{occ}_\sigma, \prec_\sigma^*)$ where $\prec_\sigma^*$ is the transitive reflexive closure of $\prec_\sigma$.

For a labelled net Σ, let ind_Σ be a symmetric relation on its transitions, defined by:

$$\mathsf{ind}_\Sigma\ =\ \{(t, u) \in T_\Sigma \times T_\Sigma \mid ({}^\bullet t \cup t^\bullet) \cap ({}^\bullet u \cup u^\bullet) = \varnothing\}\ .$$

This relation is called the *independence relation*, because two distinct transitions belonging to ind_Σ have no impact on their respective environments. If they are both enabled individually, then they are also enabled simultaneously. Then, with every finite step sequence σ of a safe labelled net Σ, $\Sigma[\sigma\rangle\Theta$, we can associate a partial order, $\mathsf{poset}_{\mathsf{ind}_\Sigma}(\sigma)$, in the way described above, and after taking A to be the transition set, $A = T_\Sigma$. The presence of a path between two nodes is interpreted as *causality*, and the lack of ordering as *concurrency*. Whenever $\Sigma[\sigma\rangle\Theta$, we will write $\Sigma\ [\mathsf{poset}_{\mathsf{ind}_\Sigma}(\sigma)\rangle_{\mathsf{po}}\ \Theta$ to indicate that Θ arises from Σ through the execution of the poset between the brackets $[\ldots\rangle$.

1.2.2 Equivalence notions

One may consider various behavioural equivalences for labelled nets. It may first be observed that the whole set of step sequences of a labelled net may be specified by defining its *full reachability graph*, whose nodes are all reachable markings (or equivalently, all derivable nets) and whose arcs are labelled with steps which transform one marking into another.

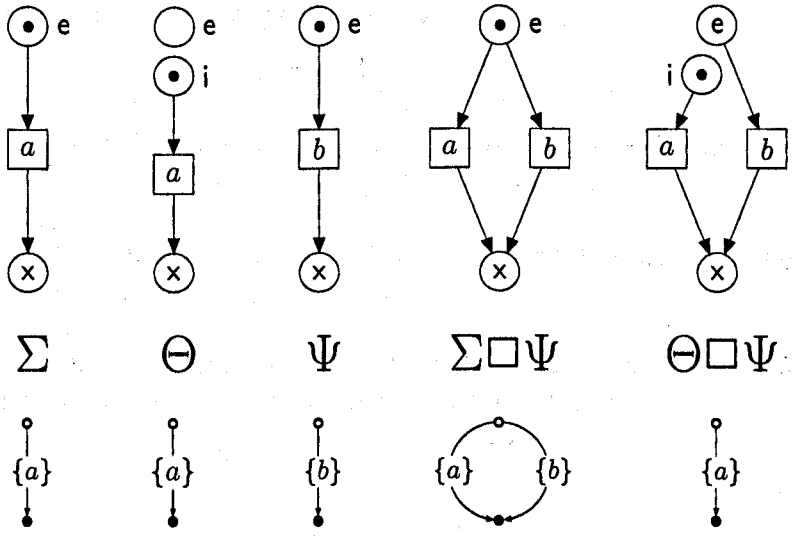


Figure 1.1: Five nets and the corresponding (labelled) full reachability graphs demonstrating that isomorphism of reachability graphs is not preserved by choice composition.

Transition systems

Using the full reachability graph to represent the behaviour of a labelled net leads to problems in a compositional setting. In particular, isomorphism of reachability graphs is not preserved by, e.g., choice composition of nets, as demonstrated

informally in figure 1.1. One can address this problem by augmenting the behaviour of a labelled net Σ with two auxiliary moves, skip and redo, which can transform its initial state into the terminal state, and vice versa. The desired effect is achieved by adding to Σ two auxiliary transitions, skip and redo, in such a way that: $^\bullet\mathsf{skip} = \mathsf{redo}^\bullet = {}^\circ\Sigma$, $\mathsf{skip}^\bullet = {}^\bullet\mathsf{redo} = \Sigma^\circ$, $\lambda(\mathsf{skip}) = \mathsf{skip}$, $\lambda(\mathsf{redo}) = \mathsf{redo}$, $\mathsf{redo}, \mathsf{skip} \notin \mathcal{A}$, and all arcs adjacent to skip and redo have weight 1. The net Σ augmented with skip and redo will be denoted by Σ_{sr}.

The *transition system* of a marked net Σ is defined as $\mathsf{ts}_\Sigma = (V, L, A, v_0)$ where: $V = \{\Theta \mid \mathsf{skip}, \mathsf{redo} \notin T_\Theta \wedge \Theta_{\mathsf{sr}} \in [\Sigma_{\mathsf{sr}}\rangle\}$ is the set of states; $v_0 = \Sigma$ is the initial state; $L = \mathsf{mult}(\mathcal{A} \cup \{\mathsf{redo}, \mathsf{skip}\})$ is the set of arc labels; and

$$A = \{(\Theta, \Gamma, \Psi) \in V \times L \times V \mid \Theta_{\mathsf{sr}}\ [\Gamma\rangle_{\mathsf{lab}}\ \Psi_{\mathsf{sr}}\}$$

is the set of arcs. In other words, ts_Σ is the labelled reachability graph of Σ_{sr} with all references to skip and redo in the nodes of the graph erased. The transition system of an unmarked net Σ is defined as $\mathsf{ts}_\Sigma = \mathsf{ts}_{\overline{\Sigma}}$.

The *full transition system* of a marked net Σ is defined as $\mathsf{fts}_\Sigma = (V, L', A', v_0)$ where: V is the set of states and v_0 is the initial state, both defined as above; L' is the set of all finite multisets of transitions of Σ_{sr}; and

$$A' = \{(\Theta, U, \Psi) \in V \times L' \times V \mid \Theta_{\mathsf{sr}}[U\rangle\Psi_{\mathsf{sr}}\}$$

is the set of arcs. In other words, fts_Σ is the reachability graph of Σ_{sr} with all references to skip and redo in the nodes of the graph erased. For an unmarked net Σ, $\mathsf{fts}_\Sigma = \mathsf{fts}_{\overline{\Sigma}}$.

Figure 1.2 shows how augmenting labelled nets with the redo and skip transitions allows one to discriminate between the nets Σ and Θ depicted in figure 1.1. In general, skip and redo allow for distinguishing the entry and exit states from the other ones, and modelling the fact that if a net is left (through the exit state), it may later be possible to re-enter it (through the entry state).

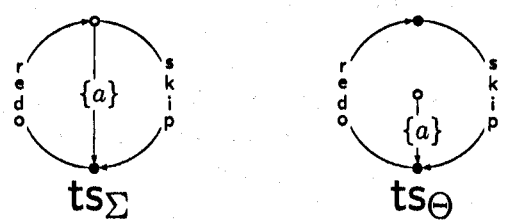


Figure 1.2: Discriminating transition systems for the nets in figure 1.1.

Behavioural equivalences

Let Σ and Θ be two labelled nets which are either both unmarked, or both marked. The nets are *ts-isomorphic*, denoted by $\Sigma \cong \Theta$, if ts_Σ and ts_Θ are

isomorphic transition systems, and *strongly equivalent (or bisimilar)*, denoted by $\Sigma \approx \Theta$, if ts_Σ and ts_Θ are strongly equivalent transition systems, where two transition systems, (V, L, A, v_0) and (V', L', A', v_0'), are *strongly equivalent* if there is a relation $R \subseteq V \times V'$, itself called a *strong bisimulation*, such that $(v_0, v_0') \in R$, and if $(v, v') \in R$ then

$$\begin{array}{lcl} (v,l,w) \in A & \Longrightarrow & \exists w' \in V' \ : \ (v',l,w') \in A' \ \wedge \ (w,w') \in R \\ (v',l,w') \in A' & \Longrightarrow & \exists w \in V \ : \ (v,l,w) \in A \ \wedge \ (w,w') \in R. \end{array}$$

Figure 1.3 shows two strongly equivalent labelled nets which are, however, not ts-isomorphic.

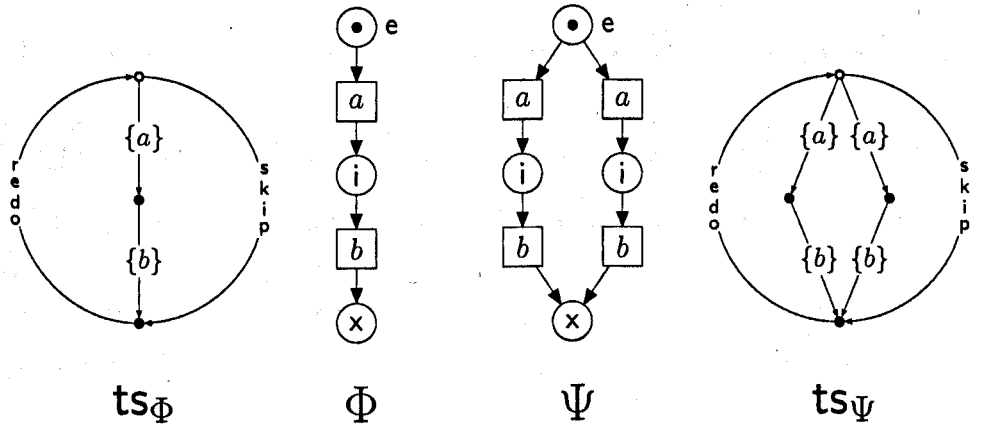


Figure 1.3: Two strongly equivalent, but not ts-isomorphic, labelled nets.

Structural equivalences

As ts-isomorphism and strong equivalence are behavioural notions, it may be difficult to check whether two nets are indeed equivalent. But, since we are working with nets, it is also possible to define equivalences based on their graph theoretic structure of nets rather than on their behaviour.

Arguably the strongest structural equivalence, other than equality, is net isomorphism. Two labelled nets, Σ and Θ, are *isomorphic* if there is a bijective mapping ψ: $S_\Sigma \cup T_\Sigma \rightarrow S_\Theta \cup T_\Theta$ such that for all $s \in S_\Sigma$ and $t \in T_\Sigma$, $\psi(s) \in S_\Theta$, $\psi(t) \in T_\Theta$, $\lambda_\Theta(\psi(s)) = \lambda_\Sigma(s)$, $\lambda_\Theta(\psi(t)) = \lambda_\Sigma(t)$, $M_\Theta(\psi(s)) = M_\Sigma(s)$, $W_\Theta(\psi(s), \psi(t)) = W_\Sigma(s,t)$ and $W_\Theta(\psi(t), \psi(s)) = W_\Sigma(t,s)$. We will denote this by Σ iso Θ, and call ψ an *isomorphism* for Σ and Θ.

A weaker equivalence is obtained by allowing the two nets differ only by duplicating places and transitions. Two places s and s' are *duplicating* in a labelled net Σ if $\lambda_\Sigma(s) = \lambda_\Sigma(s')$, $M_\Sigma(s) = M_\Sigma(s')$, and for every transition t, $W_\Sigma(s,t) = W_\Sigma(s',t)$ and $W_\Sigma(t,s) = W_\Sigma(t,s')$; then, in any evolution of the net, the two places do not add/remove anything with respect to each other. Similarly, two transitions t and t' are *duplicating* if $\lambda_\Sigma(t) = \lambda_\Sigma(t')$, and for every place s,

$W_\Sigma(s,t) = W_\Sigma(s,t')$ and $W_\Sigma(t,s) = W_\Sigma(t',s)$; then, in any label based evolution of the net, the two transitions do not add/remove anything with respect to each other.

The relation of *duplication* is an equivalence relation for places and transitions, and two labelled nets, Σ and Θ, are *duplication equivalent* if they lead to isomorphic nets when their places and transitions are replaced by their duplication equivalence classes which inherit the labels, connectivity and markings of their elements. We will denote this by Σ iso_ST Θ. For the nets in figure 1.4, we have Σ_1 iso_ST Σ_2 iso_ST Σ_3. Moreover, Σ_4 is not duplication equivalent to any of the other three nets since, intuitively, Σ_4 has no reachable terminal state (the x-places cannot be marked concurrently).

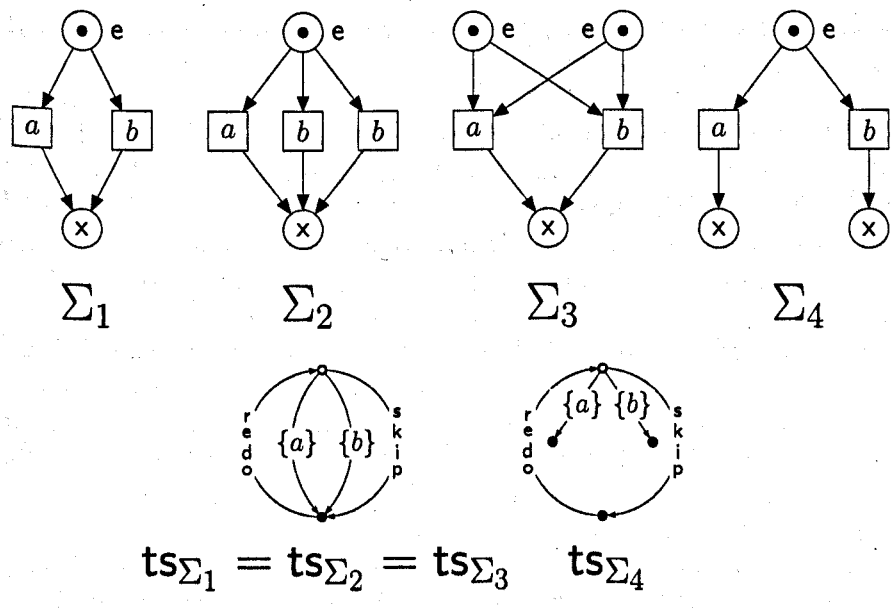


Figure 1.4: Four labelled nets and their transition systems.

1.2.3 Plain boxes

A *box* is an ex-restricted (and T-restricted) labelled net Σ. A box is *plain* if its transitions are all labelled by constant relabellings (or labels, according to our convention of identifying constant relabellings with labels). There are two main classes of plain boxes.

An unmarked plain box Σ will be called *static* if each marking reachable from ${}^\circ\Sigma$ or Σ° is safe and clean.

A marked plain box Σ is *dynamic* if each marking reachable from M_Σ or ${}^\circ\Sigma$ or Σ° is safe and clean. Note that if Σ is a static box and Θ is derivable from $\overline{\Sigma}$ then Θ is a dynamic box.

When a box Σ or Σ_{sr} is marked, its reachable markings are always non-empty (due to T-restrictedness and ex-restrictedness). On the other hand, the empty marking of a box has no successor markings except itself. Thus, the distinction between static and dynamic boxes is invariant over behaviour. Moreover, if Σ is a static box then both $\overline{\Sigma}$ and $\underline{\Sigma}$ are dynamic boxes, and if Σ is a dynamic box then $\lfloor\Sigma\rfloor$ is a static box.

Proposition 1.1. *If Σ is a dynamic box then every net derivable from it is a dynamic box. Moreover, if U is a step enabled by the marking of Σ then $U \times U \subseteq \mathsf{ind}_\Sigma \cup id_{T_\Sigma}$ and $W_\Sigma(U \times S_\Sigma) \cup W_\Sigma(S_\Sigma \times U) \subseteq \{0,1\}$.*

Moreover, there are two special classes of dynamic boxes, called the *entry* and *exit* boxes, which comprise all dynamic boxes Σ such that M_Σ is, respectively, ${}^\circ\Sigma$ and Σ°. The sets of plain static, dynamic, entry and exit boxes will, respectively, be denoted by Box^s, Box^d, Box^e, and Box^x. The entire set of plain boxes will be denoted by Box.

Static boxes Σ (or, equivalently, the entry boxes $\overline{\Sigma}$) provide the denotational semantics of the static expressions. Two behavioural conditions were imposed on the markings M reachable from the entry or exit marking of Σ. First, M is required to be safe in order to ensure that the semantics of the boxes is as simple as possible, in order to directly use the partial order semantics of Petri nets in the style of Mazurkiewicz (cf. section 1.2.1). The second condition, that M is always a clean marking, is a consequence of the first condition in order to use iterative constructs in the algebra of nets. Finally dynamic boxes are necessary to represent intermediate markings. By definition, they include all marked nets Θ such that Θ is derivable from $\overline{\Sigma}$, for some static box Σ. Dynamic boxes will provide the denotational semantics for the dynamic expressions.

A box Σ will be called *ex-exclusive* if each marking reachable from M_Σ or ${}^\circ\Sigma$ or Σ° is ex-exclusive. Moreover, for every pair of non-empty disjoint sets of places S_e and S_x, the *ex-box* $\mathsf{ex}(S_e, S_x)$ is a box Σ such that ${}^\circ\Sigma = S_e$, $\Sigma^\circ = S_x$ and $\ddot{\Sigma} = T_\Sigma = \varnothing$. The simplest ex-box has two places and no transitions.

1.2.4 Net refinement

The mechanism for providing plain boxes with an algebraic structure is a general *simultaneous refinement and relabelling meta-operator* (*net refinement*, for short). It is defined for an operator box Ω (see the following subsection) with n transitions. The transition refinement part of net refinement serves as a pattern for gluing together an n-tuple of plain boxes Σ – one plain box for every transition in Ω – along their e and x interfaces. The relabelling part of net refinement combines (synchronises) transitions from Σ and changes the interface of the resulting transition(s) according to the transformations prescribed in Ω's transition labels.

Operator boxes

An *operator box* is a simple finite unmarked box Ω all of whose transition labellings are transformational. We will assume that the set of transitions of Ω is implicitly ordered, $T_\Omega = \{v_1, \ldots, v_n\}$. Then any tuple $\Sigma = (\Sigma_{v_1}, \ldots, \Sigma_{v_n})$ consisting of plain boxes is called an *Ω-tuple*. For $T \subseteq T_\Omega$, we will denote $\Sigma_T = \{\Sigma_v \mid v \in T\}$.

1.2.5 Place and transition names of operator and plain boxes

Example of net refinement

Although this may be the right place to include a section about net substitution and the formal definition of net refinement, we decided against this. Instead, in figure 1.5 we give two small examples of net refinement together with the linear notation for the place and transition tree names and with a graph for a place and a transition tree. This decision was driven for the following reasons. First of all, we want to avoid several repetitions of the basic theory. Since, this research is actually an extension of Petri Box Calculus it is obvious that a large portion of the theory and ideas will be common in both cases. In order to increase the readability of this thesis, we decided to put the net substitution section in the beginning of chapter 4 and the formal definition of net refinement can be found in section 4.1. Furthermore, net refinement for standard boxes can be easily obtained from the definition of net refinement for at-boxes.

1.3 An algebra of process expressions

In this section we present the syntax and operational semantics of the Petri Box Calculus, according to [9].

1.3.1 Syntax

Static expressions

A *(standard) static PBC expression* is a word generated by the syntax

$$\begin{aligned} E \;::=\; & \alpha \mid E \| E \mid E \Box E \mid E ; E \mid [E * E * E] \mid \\ & \mid E \,\mathsf{sy}\, a \mid E[f] \mid E \,\mathsf{rs}\, a \mid E \,\mathsf{sc}\, a \end{aligned} \tag{1.1}$$

possibly with parentheses, used - if needed - to resolve ambiguities. In the syntax, α is a constant called *basic action* or *multiaction* and is an element of the set of labels $\mathsf{Lab}_{\mathsf{PBC}} = \mathsf{mult}(\mathcal{A})$ as defined in [9]; a is an element of $\mathcal{A}$ and $[f]$ is a

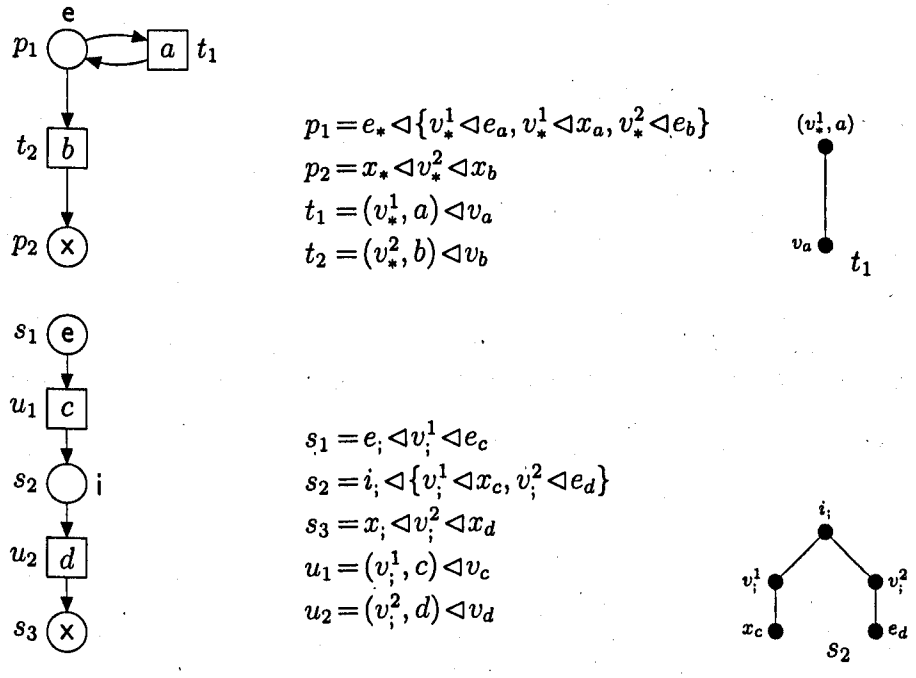


Figure 1.5: Two net refinements.

relabelling function from $\mathcal{A}$ to $\mathcal{A}$. It is sometimes required, as in the CCS framework, that such a function preserves conjugates. Moreover such an f commutes with $\widehat{\ }$ and that $\widehat{\ }$ itself is a relabelling function preserving conjugates.

PBC operators can be split into two groups. The three binary operators, i.e., $\|$ (disjoint parallelism), $\Box$ (choice), and ' ; ' (sequence), together with the ternary operator $[**]$ (iteration with explicit initialisation and termination) are *control flow operators.* The two unary (classes of) operators , i.e., $[f]$ (basic relabelling), and $\mathsf{sc}\, a$ (scoping, a combination of synchronisation and restriction) are *communication interface operators.* Finally, static boxes are the net counterparts of static expressions.

To avoid excessive bracketing, we will often replace in expressions and the corresponding Petri nets, a singleton multiaction $\{a\} \in \mathcal{A}$ by a.

Dynamic Expressions

A *(standard) dynamic PBC expression* is a word generated by the syntax:

$$\begin{array}{lllllll} G & ::= & \overline{E} & | & \underline{E} & | & G \Box E \quad | \quad E \Box G \quad | \\ & & G[f] & | & G\,\mathsf{sc}\,a & | & G;E \quad | \quad E;G \quad | \\ & & G \| G & | & [G*E*E] & | & [E*G*E] \quad | \quad [E*E*G] \end{array} \qquad (1.2)$$

where E stands for a static expression defined by the syntax 1.1.

The first two clauses of 1.2 concern a static expression E in its entry (or initial) state, $\overline{E}$, and exit (or terminal) state, $\underline{E}$. Both are valid dynamic expressions, irrespective of whether or not the final state is reachable from the initial state. An expression $G;H$ is not dynamic since dynamic composition does not allow both its components to be active at the same time. The markings of the corresponding boxes are assumed to be safe and this is already built into the theory. The shape of dynamic expressions depends on that assumption. The syntax for sequence is therefore split into two clauses. The first one, $G;E$ means that the first component of the sequential composition is currently active, while the second component is currently dormant. The second clause, $E;G$ means exactly the opposite. Similar remarks hold for choice and iteration; both require syntactically that only one of the parts of choice or iteration expression is ever active. Note that in parallel composition both components are required to be active. Finally, dynamic boxes are the net counterparts of dynamic expressions.

1.3.2 Operational semantics

In this section, we present the operational semantics of dynamic expressions, as defined in [9]. Operational semantics are divided in structural equivalence relation on expressions and rules of structural operational semantics (SOS) in the style of [54].

Structural Equivalence

The structural equivalence relation on expression aims to capture the most fundamental correspondence between expressions. For example, $\underline{E};F \equiv E;\overline{F}$ states that a sequential system in which the first component terminated is the same as that in which the second component is in its initial state. Formally, $\equiv$ is the least equivalence relation on dynamic expressions such that rules in table 1.1 are satisfied.

SOS Rules

The rules of the label and transition based operational semantics, are presented in table 1.2.

$\overline{E \| F} \equiv \overline{E} \| \overline{F}$	$\underline{E} \| \underline{F} \equiv \underline{E \| F}$
$\overline{E \Box F} \equiv \overline{E} \Box F$	$\underline{E} \Box F \equiv \underline{E \Box F}$
$\overline{E \Box F} \equiv E \Box \overline{F}$	$E \Box \underline{F} \equiv \underline{E \Box F}$
$\overline{E \,\mathsf{sc}\, a} \equiv \overline{E} \,\mathsf{sc}\, a$	$\underline{E} \,\mathsf{sc}\, a \equiv \underline{E \,\mathsf{sc}\, a}$
$\overline{E ; F} \equiv \overline{E} ; F$	$E ; \underline{F} \equiv \underline{E ; F}$
$\underline{E} ; F \equiv E ; \overline{F}$	$[D * \underline{E} * F] \equiv [D * E * \overline{F}]$
$\overline{[D * E * F]} \equiv [\overline{D} * E * F]$	$[D * E * \underline{F}] \equiv \underline{[D * E * F]}$
$[\underline{D} * E * F] \equiv [D * \overline{E} * F]$	$[D * \overline{E} * F] \equiv [D * \underline{E} * F]$
$\overline{E[f]} \equiv \overline{E}[f]$	$\underline{E}[f] \equiv \underline{E[f]}$

Table 1.1: Rules of the structural equivalence.

$\overline{\alpha} \xrightarrow{\{\alpha\}} \underline{\alpha}$	$\overline{\alpha} \xrightarrow{\{v_\alpha\}} \underline{\alpha}$
$\dfrac{G \xrightarrow{\Gamma} G',\ H \xrightarrow{\Delta} H'}{G\|H \xrightarrow{\Gamma+\Delta} G'\|H'}$	$\dfrac{G \xrightarrow{U} G',\ H \xrightarrow{W} H'}{G\|H \xrightarrow{v_\|^1 \blacktriangleleft U \cup v_\|^2 \blacktriangleleft W} G'\|H'}$
$\dfrac{G \xrightarrow{\Gamma} H}{G\Box E \xrightarrow{\Gamma} H\Box E}$	$\dfrac{G \xrightarrow{U} H}{G\Box E \xrightarrow{v_\Box^1 \blacktriangleleft U} H\Box E}$
$\dfrac{G \xrightarrow{\Gamma} H}{E\Box G \xrightarrow{\Gamma} E\Box H}$	$\dfrac{G \xrightarrow{U} H}{E\Box G \xrightarrow{v_\Box^2 \blacktriangleleft U} E\Box H}$
$\dfrac{G \xrightarrow{\Gamma} H}{G;E \xrightarrow{\Gamma} H;E}$	$\dfrac{G \xrightarrow{U} H}{G;E \xrightarrow{v_;^1 \blacktriangleleft U} H;E}$
$\dfrac{G \xrightarrow{\Gamma} H}{E;G \xrightarrow{\Gamma} E;H}$	$\dfrac{G \xrightarrow{U} H}{E;G \xrightarrow{v_;^2 \blacktriangleleft U} E;H}$
$\dfrac{G \xrightarrow{\Gamma} H}{[G*E*F] \xrightarrow{\Gamma} [H*E*F]}$	$\dfrac{G \xrightarrow{U} H}{[G*E*F] \xrightarrow{v_*^1 \blacktriangleleft U} [H*E*F]}$
$\dfrac{G \xrightarrow{\Gamma} H}{[E*G*F] \xrightarrow{\Gamma} [E*H*F]}$	$\dfrac{G \xrightarrow{U} H}{[E*G*F] \xrightarrow{v_*^2 \blacktriangleleft U} [E*H*F]}$
$\dfrac{G \xrightarrow{\Gamma} H}{[E*F*G] \xrightarrow{\Gamma} [E*F*H]}$	$\dfrac{G \xrightarrow{U} H}{[E*F*G] \xrightarrow{v_*^3 \blacktriangleleft U} [E*F*H]}$
$\dfrac{G \xrightarrow{\Gamma} H}{G[f] \xrightarrow{f(\Gamma)} H[f]}$	$\dfrac{G \xrightarrow{U} H}{G[f] \xrightarrow{(v_{[f]},f) \blacktriangleleft U} H[f]}$
$\dfrac{G \xrightarrow{\Gamma_1+\ldots+\Gamma_k} H}{[G\,\mathsf{sc}\,a] \xrightarrow{\{\alpha_1,\ldots,\alpha_k\}} [H\,\mathsf{sc}\,a]}$ where $(\Gamma_i,\alpha_i) \in \varrho_{\mathsf{sc}\,a}$	$\dfrac{G \xrightarrow{U_1 \uplus \ldots \uplus U_k} H}{[G\,\mathsf{sc}\,a] \xrightarrow{\{(v_{\mathsf{sc}\,a},\alpha_1) \lhd U_1,\ldots,(v_{\mathsf{sc}\,a},\alpha_k) \lhd U_k\}} [H\,\mathsf{sc}\,a]}$ where $(\mathsf{lab}(U_i),\alpha_i) \in \varrho_{\mathsf{sc}\,a}$

Table 1.2: PBC operational semantics rules.

Chapter 2

Petri nets with time restrictions

The main ingredients of this chapter are Petri nets that have been extended with different time restrictions. There will be a concise presentation of several kinds of possible time extensions and a more extensive analysis of the timed-arc Petri nets and their applications since this type of Petri nets will be used throughout this thesis. Finally, a section of this chapter will cover the existing research achievements on the combination of process algebras with Petri nets that have time restrictions in the sense of Petri Box Calculus.

2.1 Time Petri nets

Petri nets is a strong and efficient tool used in the description and analysis of concurrent systems. They support a graphical representation of concurrent systems and since they are based on a theory of partial orders (capturing explicit asynchrony), provide an additional means of verifying their correctness efficiently. Furthermore, they can be used to express properties related to causality and concurrency in system behaviour. On the other hand, in the real world concurrent systems, most of the actions are time related. There was no consideration for timing variables in the standard Petri net model and it was necessary to introduce some to precisely model concurrent systems. Several models that use timing assumptions have been presented in the literature, for a survey see [16, 68]. According to [68] they have the same form as standard Petri nets but their labelling consists of assigning numerical values or intervals to their places, transitions, arcs or even on combination of these. The newly introduced time variables (restrictions) will affect the enableness and execution of transitions and the firing rules of standard Petri nets must be altered to reflect these changes. Moreover, since these time variables have been introduced, it is necessary to actually being able to count the passage of time. Clocks can be global (counting the age of the whole system) or local (for example, counting the age of tokens). All possible combinations are feasible and the modeler can decide the best possible combination according to the modeling needs. Some interesting approaches will be mentioned here without getting into many details. One of the first attempts to introduce

time variables in Petri nets was made on [56]. In this approach, time labels were placed at each transition, denoting the fact that actions are not instantaneous. This type of Petri nets are called *Timed Petri nets.* A different approach for timing restrictions on transitions was proposed by Merlin and Farber in [44] and this time, two values have been attached to each transition that correspond to the minimum and maximum time an enabled transition has to wait before it can actually fire. These extended Petri nets are called *Time Petri nets.* Time Petri nets have been been used in protocol and real-time system modelling and verification in [67]. An approach with time labels on places was made in [20,63], modelling processes that consume time. This type of nets was used to analyze the time dependence of all places in the class of "time-driven" systems. *Timed-Arc Petri nets* [15,31,50] is timed extension of Petri nets where a time interval is associated with each place outgoing arcs and time passing affects the age of tokens. It is precisely this kind of time modelling which has been adopted in this thesis and will be presented in the following section.

Basic definitions

To model timing restrictions throughout this thesis, we use the following notation:

$$\begin{aligned}
\mathbb{D}^{\infty} &\stackrel{\text{df}}{=} \{el \mid e \in \mathbb{N} \wedge l \in \mathbb{N}^{\infty} \wedge e \leq l\} \\
\mathbb{D} &\stackrel{\text{df}}{=} \{el \in \mathbb{D}^{\infty} \mid l \neq \infty\} \\
\mathbb{D}^{\perp} &\stackrel{\text{df}}{=} \mathbb{D} \cup \{\perp\} \\
\mathbb{N}^{\perp} &\stackrel{\text{df}}{=} \mathbb{N} \cup \{\perp\}.
\end{aligned}$$

Let $n \in \mathbb{N}$, $\mathbb{EL} \in \mathbb{D}$ and $el \in \mathbb{D}^{\infty}$. Then n satisfies the timing restriction el if $e \leq n \leq l$, and $\mathbb{EL}$ satisfies the timing restriction el if $e \leq \mathbb{E}$ and $\mathbb{L} \leq l$. We denote this by n tsat el and $\mathbb{EL}$ tsat el, respectively. Moreover, for every pair $\xi, \nu \in \mathbb{D}^{\perp}$, we denote

$$\xi \oplus \nu \stackrel{\text{df}}{=} \begin{cases} \perp & \text{if } \xi = \nu = \perp \\ \mathbb{EL} & \text{if } \{\xi, \nu\} = \{\perp, \mathbb{EL}\} \\ \min\{\mathbb{E}, \mathbb{E}'\} \max\{\mathbb{L}, \mathbb{L}'\} & \text{if } \xi = \mathbb{EL} \wedge \nu = \mathbb{E}'\mathbb{L}'. \end{cases}$$

2.2 Petri nets with arc-based time restrictions

A *timed-arc Petri net* (or at-net) is a tuple $\Sigma \stackrel{\text{df}}{=} (P, T, F, \lambda, M)$ such that:

- (P, T, F) is a net and F is a flow relation such that $F \subseteq (P \times T) \cup (T \times P)$.
- λ is a mapping with the domain $P \cup T \cup ((P \times T) \cap F)$ such that, for every place $p \in P$ and transition $t \in T$, $\lambda(p)$ is a symbol in $\{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$, $\lambda(t)$ is an action in $\mathcal{A} \cup \{\imath\}$, and if $(p, t) \in F$ then $\lambda(p, t)$ is a time constraint in $\mathbb{D}^{\infty}$.
- $M : P \to \mathbb{N}$ is a marking.

Essentially this is a Petri net with time annotations in the place outgoing arcs but its marking is the same as in standard Petri nets.

Since the main topic of this research is the translation of 'timed' expressions into 'timed' boxes in the sense of PBC and in order to improve the readability of the thesis, it is necessary to present a simplified definition for *timed-arc Petri boxes*. A more detailed definition will be presented in a following chapter.

A *timed-arc box* (or at-box) is an at-net with interfaces for applying composition operators and is defined as a tuple $\Theta \stackrel{\text{df}}{=} (P, T, F, \lambda, \mu)$ such that:

- P, T and F are as in the definition of a PT-net.
- λ is a mapping with the domain $P \cup T \cup ((P \times T) \cap F)$. For every place $p \in P$ and transition $t \in T$, we have the following: $\lambda(p)$ is a symbol in $\{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$; $\lambda(t)$ is an action in $\mathcal{A} \cup \{\imath\}$; and if $(p, t) \in F$ then $\lambda(p, t) \in \mathbb{D}^{\infty}$.
- $\mu : P \to \mathbb{N}^{\perp}$ is the *initial* token timing mapping of Θ (in general, any such mapping is a token timing of Θ).

Note that token timing mappings of at-boxes are interpreted differently from markings of PT-nets, namely, $\mu(p) = k$ means that p holds a single token which is k units of time old, and $\mu(p) = \perp$ means that p is empty.

As before, we adopt the standard rules concerning the drawing of diagrams. In the diagrams, the empty local state $\perp$ will not be represented, and otherwise $\mu(p)$ will be displayed. Other drawing conventions are the same as for the standard Petri nets.

The 'time-less' version of Θ is defined as a PT-net $\lfloor\Theta\rfloor \stackrel{\text{df}}{=} (P, T, F, \lfloor\mu\rfloor)$ such that, for every $p \in P$,

$$\lfloor\mu\rfloor(p) \stackrel{\text{df}}{=} \begin{cases} 1 & \text{if } \mu(p) \in \mathbb{N} \\ 0 & \text{if } \mu(p) = \perp. \end{cases}$$

In what follows, $\lfloor\Theta\rfloor$ will be called the *underlying* net of Θ, and we will assume that it is always safe. It is worth stressing out that at-boxes are nothing but ordinary nets with time annotations on the input arcs and a different kind of token mapping. Since the underlying net is essentially the same, every property concerning nets (e.g., ex-directedness) presented in the previous chapter for standard Petri nets will continue to hold.

In the at-box model, time restrictions are associated with the arcs incoming to transitions. For example, if $\lambda(p, t) = el$, then the interval $el = (e, l)$ gives the waiting time for the tokens flowing from place p to transition t. This interval identifies the time for which a token has to wait in place p before it can be used to fire transition t on this occasion. The left bound, e, is called the *minimum* waiting time and the right bound, l, the *maximum* waiting time. A token on p cannot be used to fire t when it is younger than the minimum waiting time and must be used to fire an enabled transition before the maximum waiting time has finished (unless the transition has been disabled in the meantime). If t is not

enabled and the maximum waiting time has passed, the token can no longer be used to fire transition t. The age of tokens is represented through a token timing which returns, for each place containing a token, its age ($\perp$ is returned if a given place is empty). When a token arrives to a place, its age is set to zero. After that the age can be increased due to the passage of time. It should be emphasized that a token does not need to enable any transition in order for its clock to start 'ticking'.

A finite set of transitions $U = \{t_1, \ldots, t_k\}$, called a *step*, is *enabled* by a token timing μ if it is enabled at the marking $\lfloor\!\lfloor \mu \rfloor\!\rfloor$ in the safe underlying PT-net and, moreover, if $t \in U$ and $p \in {}^\bullet t$ then $\mu(p)$ tsat $\lambda(p,t)$. Such a step may *fire* leading to a *follower* token timing ν such that, for every place $p \in P$,

$$\nu(p) \stackrel{\text{df}}{=} \begin{cases} \perp & \text{if } p \in {}^\bullet U \setminus U^\bullet \\ 0 & \text{if } p \in U^\bullet \\ \mu(p) & \text{otherwise .} \end{cases}$$

We denote this by $\mu[U\rangle\nu$.

Another kind of dynamic changes is effected by time moves. A token timing μ can change into token timing ν by the passage of one time unit if, for every transition t enabled at μ and for every place $p \in {}^\bullet t$ we have $\mu(p) < l$, where $el = \lambda(p,t)$. The change results in a new token timing ν such that, for every place $p \in P$,

$$\nu(p) \stackrel{\text{df}}{=} \begin{cases} \mu(p) + 1 & \text{if } \mu(p) \in \mathbb{N} \\ \mu(p) & \text{otherwise .} \end{cases}$$

We denote this by $\mu[\surd\rangle\nu$. Intuitively, at-boxes' time deadlines are assumed to be *hard*, i.e., when a transition is ready to fire and even if only one of its input tokens has reached the maximum waiting time, then this transition must fire (or become disabled) before further passage of time.

The overall behaviour of Θ is captured by its *reachability tree* with nodes labelled by token timings and arcs annotated by labelled moves, denoted by RT_Θ. More precisely, the root node is labelled by the initial token timing and, if a node is labelled by μ, then for every move $\mu[x\rangle\nu$ there is a unique descendant labelled by ν; the arc leading to it is labelled by $\surd$ if $x = \surd$, and by the multiset of communication labels

$$\lambda(U) \stackrel{\text{df}}{=} \sum_{t \in U} U(t) \cdot \{\lambda(t)\}$$

if $x = U$ is an executed transition step. Figure 2.1 shows an at-box Θ and the corresponding reachability tree RT_Θ. The use of reachability trees instead of reachability graphs may be quite surprising at the moment but will be explained later in this thesis together with the considerations that led to this decision.

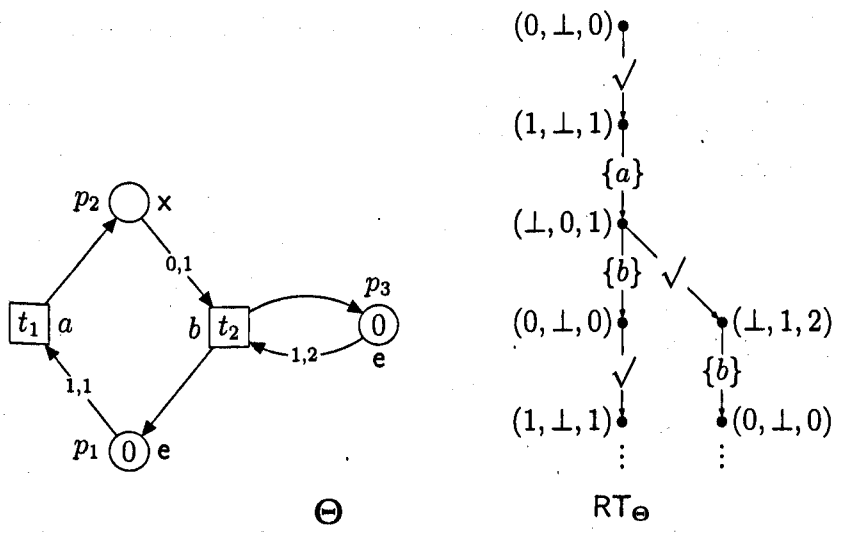


Figure 2.1: An at-box Θ and a part of its reachability tree RT_Θ.

2.3 Research on Petri nets with time restrictions

2.3.1 Timed-arc Petri nets

From the definition of timed-arc Petri nets it is clear that timing variables has been introduced to associate clocks (or age) with resources. More precisely, one can specify how old/young a given resource consumed by an action must be. This approach has been extensively studied in the past, both as a model for dealing with complex concurrent systems such as communication protocols, and as a framework for verifying their properties.

Standard Petri nets cannot simulate a Turing machine and in [14], it is proved that the same holds for timed-arc Petri nets, since they cannot simulate a counter with zero testing. With such a result in mind, showing rather 'weak expressiveness', one could expect that the reachability problem would be decidable like for the standard Petri nets. But in [59] it has been proven that reachability is undecidable for this type of nets. On the other hand, coverability and boundedness is decidable as shown in [1, 23]. The suitability of timed-arc Petri nets for the description of concurrent systems and the preference of software engineers to work with formalism close to programming languages led to an interesting approach presented in [60]. In this paper, a timed algebraic language (TPAL) [52] was automatically translated into timed-arc Petri nets. The specifications of a complete train-gate controller was presented using TPAL and the resulting timed-arc Petri net is presented in figure 2.2. In a different context, timed-arc Petri nets have been used for the performance analysis of a real life algorithm for video

processing, the MPEG-2. The MPEG-2 [33] is a standard intended for a wide range of applications such as Video-on-Demand, High Definition TV(HDTV) and video communications via broadband networks. In [53, 61, 62], the specification of the algorithm has been modelled with timed-arc Petri nets. The analysis of the algorithm showed that the performance of the encoding algorithm for a single Group-of-Pictures (GoP) can be improved by taking into account the potential parallelism in the encoding process of GoPs (a video sequence can be considered as a sequence of GoPs, for more info see [33]). The results showed that performance has increased by 50% when two processors were used. A further increase around 90% was also possible when the necessary number of processors was used to exploit all possible parallelism. The timed-arc Petri nets that models the MPEG-2 algorithm is presented in figure 2.3. In this figure, the time intervals will not be shown for arcs that have interval $(0, \infty)$.

2.3.2 Relation with process algebras

Petri Box Calculus provided the necessary framework to combine and take full advantage of the relative advantages of process algebras and Petri nets. The original framework didn't allow to specify timing restrictions for the actions employed by a concurrent system. In order to explicitly represent this kind of information, two extensions of the PBC with (discrete) time restrictions have been defined, namely tPBC [37] and TPBC [39]. These two approaches are different in several aspects. The most important difference is the way that timing information is captured in these two models. In [37], time Petri Nets [44] have been used. In this type of nets, the timing information is represented with an interval attached to each transition of the net. This interval represents the earliest and latest firing time for the given transition and the local clock for every transition starts the moment this transition becomes enabled in the usual Petri nets sense. In [39], the model is based on timed Petri nets [56]. The timing restrictions in timed Petri nets are again attached to each transition but they are different than before. Instead of a firing interval, in this case each transition has been associated with a number denoting the fact that transitions are used to represent actions and actions take time to complete. The duration of actions makes this model somehow more complicated but this was necessary to avoid the *illegal action occurrences* that were present in [37]. Moreover, TPBC has been extended further in [40, 41]. Beside duration, now each action is associated with an interval, that constrain the execution to start within the given interval (like in [37]). New constructs have been introduced to support time-out and delays, which allows to give an alternative continuation to processes where time bounds have been exceeded as well as semantics for maximal parallelism, that forces all actions to be executed the very moment they become available. In both approaches there is a strong result about the consistency between denotational and operational semantics. In [37] there is strong bisimulation between the transition systems of *t-expressions* and *ct-boxes*, while in [39–41] the main result states that every step

sequence of the operational semantics of an expression is also a step sequence of any time box that corresponds to the expression and vice versa. Although these two approaches seem to be in competition, they are complementary to each other and provide a more complete solution of the problem at hand.

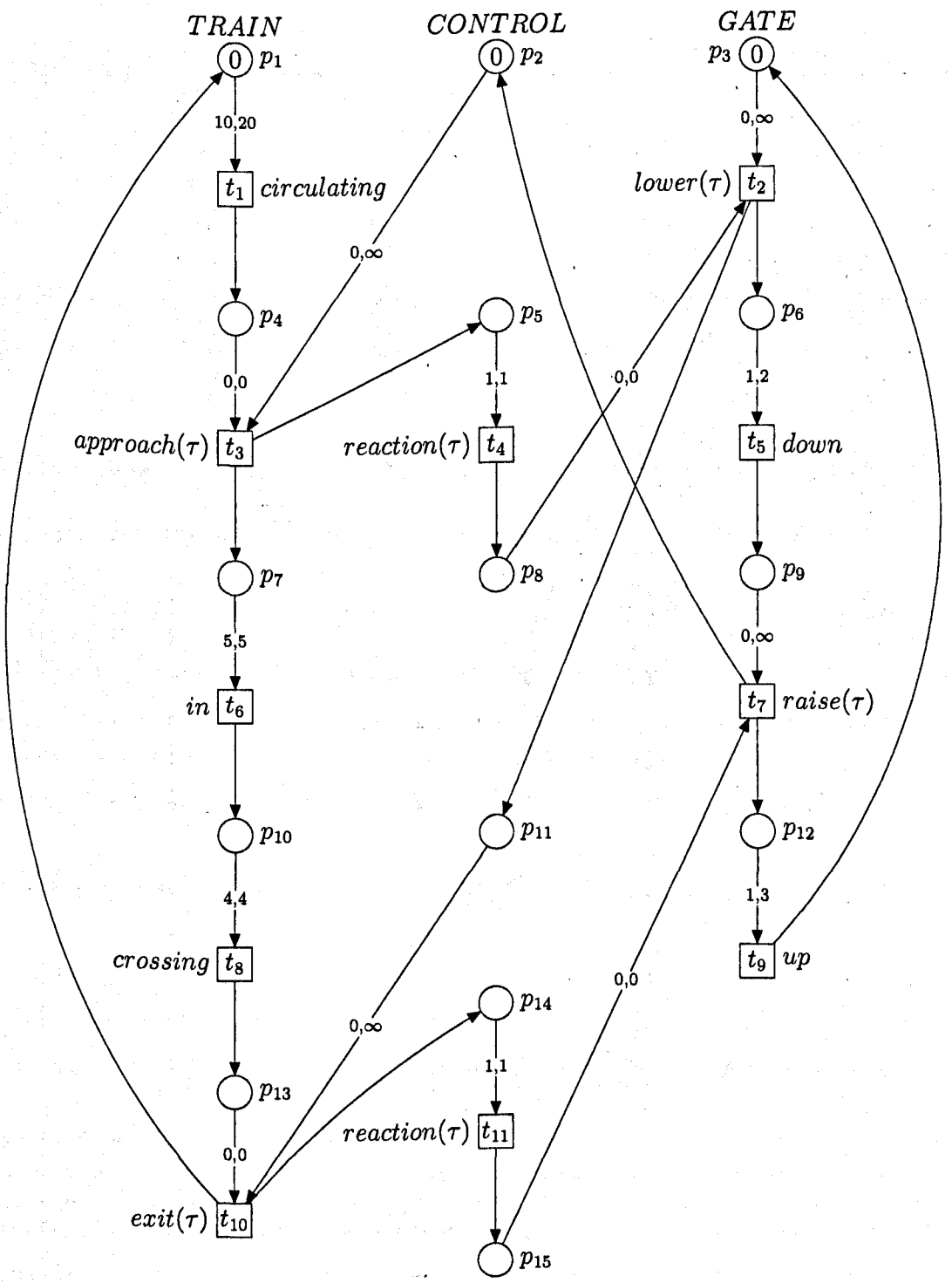


Figure 2.2: Marked timed-arc Petri net model of the Train-Gate Controller.

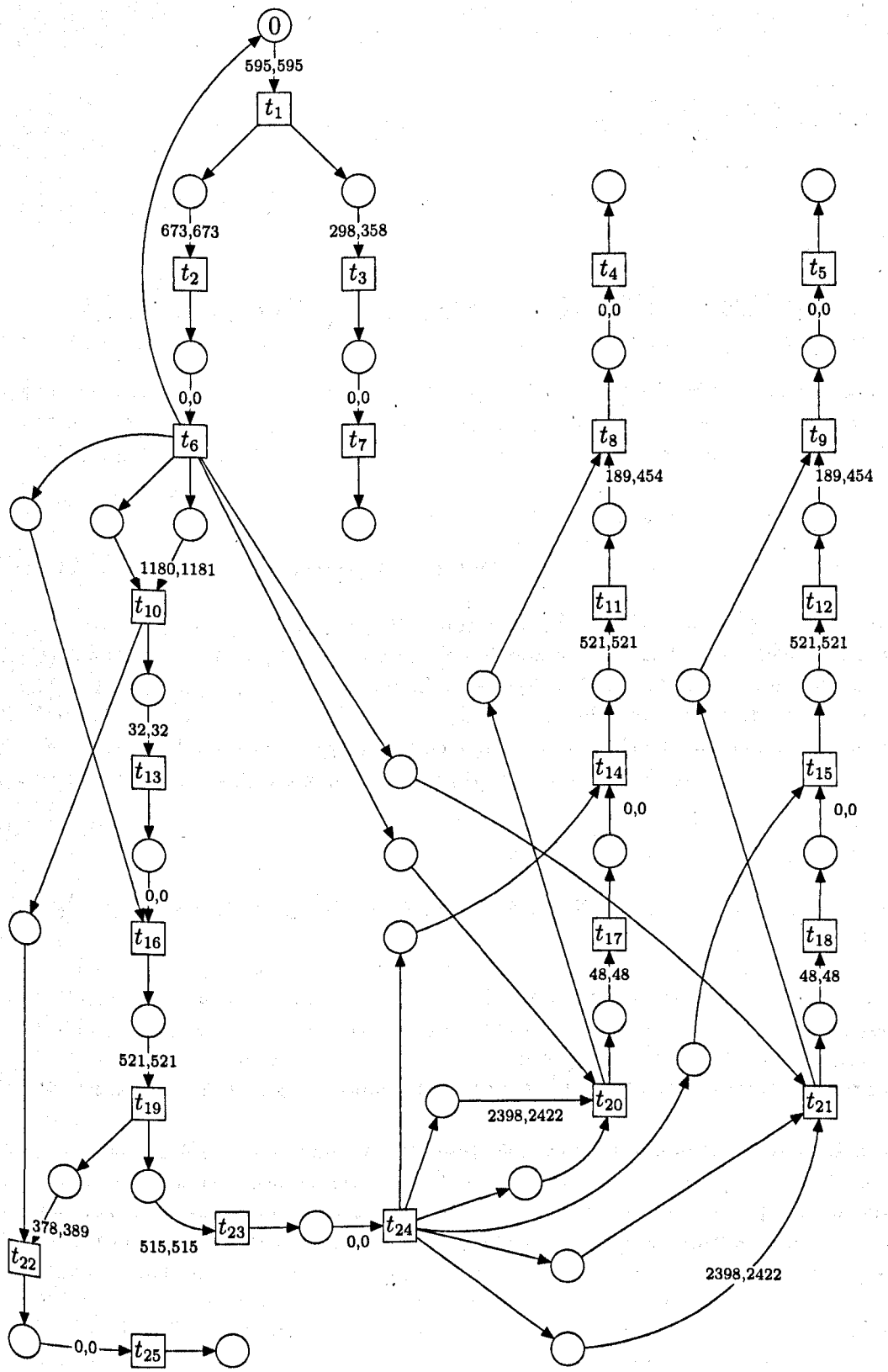


Figure 2.3: Timed-arc Petri net model of the MPEG-2 encoding process of a GoP

Chapter 3

Algebra of process expressions

In this chapter, we begin our contribution by defining the syntax of the new atPBC followed by the operational semantics of process expressions corresponding to at-boxes.

3.1 Static at-expressions

The following is the syntax for the *static timed-arc box expressions* (or *static at-expressions*), E, which correspond to at-boxes without tokens (below A is a finite subset of $\mathcal{A}$, and Z is an auxiliary subset of static at-expressions which is needed to ensure that the nets corresponding to *at-expressions* are always ex-exclusive, and due to the standard box algebra theory, the at-nets corresponding to the expressions E are safe and clean (see proposition 4.1).

$$\begin{aligned} E &::= \alpha el \mid E \,\mathsf{sc}\, A \mid E \Box E \mid E \| E \mid E ; E \mid \langle\!\langle E \circledast Z \circledast E \rangle\!\rangle \\ Z &::= Z \,\mathsf{sc}\, A \mid Z \Box Z \mid E ; E \mid \langle\!\langle E \circledast Z \circledast E \rangle\!\rangle \,. \end{aligned} \tag{3.1}$$

The only real modification, when compared with the standard PBC syntax, is that a different type of constant expression is used, viz. αel where $\alpha \in \mathcal{A} \cup \{\imath\}$ is a basic action $el \in \mathbb{D}^{\infty}$ is a timing restriction. Moreover, the actions employed by the syntax allow two-way rather than multi-way synchronisation. Similarly as in the case of at-boxes, e denotes the *minimum*, and l the *maximum* waiting time.

Sequence $E ; F$ and choice $E \Box F$ compositions are standard; the $\Box$ is used to denote what is essentially the $+$ in CCS [46] and the comma (,) in COSY [34]. The iterative construct $\langle\!\langle D \circledast E \circledast F \rangle\!\rangle$ means 'perform D once, then perform zero or more repetitions of E, then perform F once'. The basic expression αel means 'upon its activation, execute a single action with communication capabilities α and terminate, waiting at least e units of time and no more than l units of time to do so'. The concurrent composition operator is basically a disjoint union

and hence differs from its counterparts in CCS and COSY, and is similar to the $\|_\emptyset$ in TCSP [58]. For instance, $a00\|\widehat{a}00$ can perform the $\{a\}$ and $\{\widehat{a}\}$ actions individually (as well as a two-action step $\{a, \widehat{a}\}$), but no synchronised action (in contrast to $a.nil|\widehat{a}.nil$ in CCS). Finally, scoping $E\,\mathsf{sc}\,A$ implements a combination of synchronisation and restriction. In essence, it applies the CCS synchronisation mechanism over all the concurrently enabled pairs $(a, \widehat{a})$, for $a \in A$, of conjugate action names but it prevents the individual actions a and $\widehat{a}$ from occurring.

Static expressions describe structural characteristics of concurrent systems. Their behaviour will be modelled using dynamic at-expressions, introduced next.

Example

Following the introduction of the syntax for the static at-expression, it is now possible to model the timed-arc Petri net depicted in figure 2.2 and show the usefulness of the process algebra part of our approach. In this example, there are three components, namely the *Train*, the *Control* and the *Gate*. These three components work in parallel and there is also some communication between them. This communication will be achieved by the usage of the scoping operator in actions t_n and their conjugates $\widehat{t_n}$. The silent (in the standard case) action coming from the synchronisation of t_n and $\widehat{t_n}$ is denoted in the timed-arc Petri net by t_n. Furthermore, all of the components are in a constant loop. To model this, the use of the iteration operator is necessary. The choice we made about the iteration operator in our syntax forces us to have an entry (set of) action(s) and also an exit (set of) action(s). In the *'train-gate controller'* example as presented in figure 2.2 only the iterative part is present. The entry action will be a silent τ action that is not adding anything to the model and the exit action will be an action *stop* that is scoped and thus cannot be executed. To begin with, we will provide the at-expression for each of the three components and then we'll present the final model for the *'train-gate controller'*.

- *Train*

$$\langle\langle \tau \circledast t_1\langle 10, 20\rangle ; t_3\langle 0, 0\rangle ; t_6\langle 5, 5\rangle ; t_8\langle 4, 4\rangle ; t_{10}\langle 0, 0\rangle \circledast stop\rangle\rangle$$

- *Control*

$$\langle\langle \tau \circledast \widehat{t_3}\langle 0, \infty\rangle ; t_4\langle 1, 1\rangle ; t_2\langle 0, 0\rangle ; \widehat{t_{10}}\langle 0, \infty\rangle ; t_{11}\langle 1, 1\rangle ; t_7\langle 0, 0\rangle \circledast stop\rangle\rangle$$

- *Gate*

$$\langle\langle \tau \circledast \widehat{t_2}\langle 0, \infty\rangle ; t_5\langle 1, 2\rangle ; \widehat{t_7}\langle 0, \infty\rangle ; t_9\langle 1, 3\rangle \circledast stop\rangle\rangle$$

- *Train-Gate Controller*

$$\begin{aligned}&(\langle\langle \tau \circledast t_1\langle 10, 20\rangle ; t_3\langle 0, 0\rangle ; t_6\langle 5, 5\rangle ; t_8\langle 4, 4\rangle ; t_{10}\langle 0, 0\rangle \circledast stop\rangle\rangle \;\|\\ &\langle\langle \tau \circledast \widehat{t_3}\langle 0, \infty\rangle ; t_4\langle 1, 1\rangle ; t_2\langle 0, 0\rangle ; \widehat{t_{10}}\langle 0, \infty\rangle ; t_{11}\langle 1, 1\rangle ; t_7\langle 0, 0\rangle \circledast stop\rangle\rangle \;\|\\ &\langle\langle \tau \circledast \widehat{t_2}\langle 0, \infty\rangle ; t_5\langle 1, 2\rangle ; \widehat{t_7}\langle 0, \infty\rangle ; t_9\langle 1, 3\rangle \circledast stop\rangle\rangle)\,\mathsf{sc}\{t_3, t_{10}, t_2, t_7, stop\}\end{aligned}$$

3.2 Dynamic at-expressions

The syntax of (standard) dynamic PBC expressions is changed by adding time related annotations to the over- and underbars. Each such annotation is a pair of two non-negative integers that correspond to the *age* of the 'youngest' and 'oldest' token that might be consumed. For example, $\overline{E}^{00}$ is an expression E which is in its *initial state* and all tokens present are zero time units old. Another example, $\underline{E}_{35}\,;F$, is a sequential composition where the first component has terminated, and produced some tokens. The exact number (and clock values) of these tokens is not represented by the annotation, but what is represented is the age of the youngest token (3 time units), and the age of of the oldest one (5 time units). Effectively, this means that the annotation gives an *age range* for the tokens in the state which is represented by the expression. This, in general, provides less information than that conveyed by the token timings provided by at-boxes. However, it will turned out that this reduced (or abstracted) view is sufficient to reason about the behaviour. We will re-visit this issue later on.

The *dynamic at-expressions*, G, are defined below, where E and Z denote static at-expression as in the syntax (3.1) and $\mathbb{EL} \in \mathbb{D}$.

$$
\begin{array}{rcl}
G & ::= & \overline{E}^{\mathbf{EL}} \mid G;E \mid E;G \mid \langle\!\langle E \circledast K \circledast E \rangle\!\rangle \mid \langle\!\langle E \circledast Z \circledast G \rangle\!\rangle \mid \\
 & & \underline{E}_{\mathbf{EL}} \mid G \Box E \mid E \Box G \mid \langle\!\langle G \circledast Z \circledast E \rangle\!\rangle \mid G \| G \\
 & & G \,\mathsf{sc}\, A \\
K & ::= & \overline{Z}^{\mathbf{EL}} \mid G;E \mid E;G \mid \langle\!\langle G \circledast Z \circledast E \rangle\!\rangle \mid \langle\!\langle E \circledast Z \circledast G \rangle\!\rangle \mid \\
 & & \underline{Z}_{\mathbf{EL}} \mid K \Box Z \mid Z \Box K \mid \langle\!\langle E \circledast K \circledast E \rangle\!\rangle \mid K \,\mathsf{sc}\, A
\end{array}
\tag{3.2}
$$

Given that we are primarily interested in at-expressions that can be derived from expressions of the form $\overline{E}^{00}$, the above syntax may appear to be too permissive. For example, it admits expressions like $\overline{a03}^{55}$ which has an inconsistent timing information (the enabled action cannot wait for more than 3 time units before being executed, yet the age of the enabling tokens is already 5). However, such an expression may be a part of another, fully consistent expression, e.g., $(\overline{a03}^{55})\,\mathsf{sc}\{a\}$, and thus cannot be excluded. In addition, for any dynamic at-expression G, we denote by $\lfloor G \rfloor$, the static at-expression obtained from G by removing all the overbars and underbars. As a final comment we have to stress that parallel composition is not allowed inside the iterated part of the iteration operator. The reason for this choice is that we want to preserve safeness in our expressions in order to keep our time semantics as clean as possible.

3.3 Operational semantics of at-expressions

We follow the way through which the semantics of the standard PBC was defined, with appropriate modifications in order to address timing restrictions. We

$$
\begin{array}{rcl@{\qquad}rcl}
\overline{E\|F}^{\mathbb{EL}} & \equiv & \overline{E}^{\mathbb{EL}}\|\overline{F}^{\mathbb{EL}} & \underline{E}_{\mathbb{EL}}\|\underline{F}_{\mathbb{E'L'}} & \equiv & \underline{E\|F}_{\min\{\mathbb{E},\mathbb{E'}\}\max\{\mathbb{L},\mathbb{L'}\}} \\
\overline{E\Box F}^{\mathbb{EL}} & \equiv & \overline{E}^{\mathbb{EL}}\Box F & \underline{E}_{\mathbb{EL}}\Box F & \equiv & \underline{E\Box F}_{\mathbb{EL}} \\
\overline{E\Box F}^{\mathbb{EL}} & \equiv & E\Box\overline{F}^{\mathbb{EL}} & E\Box\underline{F}_{\mathbb{EL}} & \equiv & \underline{E\Box F}_{\mathbb{EL}} \\
\overline{E\,\mathsf{sc}\,A}^{\mathbb{EL}} & \equiv & \overline{E}^{\mathbb{EL}}\,\mathsf{sc}\,A & \underline{E}_{\mathbb{EL}}\,\mathsf{sc}\,A & \equiv & \underline{E\,\mathsf{sc}\,A}_{\mathbb{EL}} \\
\overline{E;F}^{\mathbb{EL}} & \equiv & \overline{E}^{\mathbb{EL}};F & E;\underline{F}_{\mathbb{EL}} & \equiv & \underline{E;F}_{\mathbb{EL}} \\
\underline{E}_{\mathbb{EL}};F & \equiv & E;\overline{F}^{\mathbb{EL}} & & & \\
\overline{\langle\!\langle D\circledast E\circledast F\rangle\!\rangle}^{\mathbb{EL}} & \equiv & \langle\!\langle\overline{D}^{\mathbb{EL}}\circledast E\circledast F\rangle\!\rangle & & & \\
\langle\!\langle\underline{D}_{\mathbb{EL}}\circledast E\circledast F\rangle\!\rangle & \equiv & \langle\!\langle D\circledast\overline{E}^{\mathbb{EL}}\circledast F\rangle\!\rangle & & & \\
\langle\!\langle D\circledast\underline{E}_{\mathbb{EL}}\circledast F\rangle\!\rangle & \equiv & \langle\!\langle D\circledast E\circledast\overline{F}^{\mathbb{EL}}\rangle\!\rangle & & & \\
\langle\!\langle D\circledast\overline{E}^{\mathbb{EL}}\circledast F\rangle\!\rangle & \equiv & \langle\!\langle D\circledast\underline{E}_{\mathbb{EL}}\circledast F\rangle\!\rangle & & & \\
\langle\!\langle D\circledast F\circledast\underline{F}_{\mathbb{EL}}\rangle\!\rangle & \equiv & \underline{\langle\!\langle D\circledast E\circledast F\rangle\!\rangle}_{\mathbb{EL}} & & &
\end{array}
$$

Table 3.1: Rules of the structural equivalence for at-expressions.

first define a structural equivalence relation on at-expressions which aims to capture the most fundamental correspondence between expressions. For example, $\underline{E}_{\mathbb{EL}};F \equiv E;\overline{F}^{\mathbb{EL}}$ states that a sequential system in which its first component has terminated is the same as the system in which the second component is ready to begin its operation. The time annotations are not changed since the entire state produced by the first component is passed to the second one. Formally, $\equiv$ is the least equivalence relation on dynamic at-expressions such that the rules in table 3.1 are satisfied. Note that we do not give any rule for $\overline{E}^{\mathbb{EL}}\|\overline{F}^{\mathbb{E'L'}}$ with $\mathbb{EL} \neq \mathbb{E'L'}$ as such an expression can never be derived from initially marked static expressions, which are the only at-expressions we are interested in.

Proposition 3.1. *Assuming that we treat the rules in table 3.1 as term rewriting rules, if $G \equiv H$ and G is an at-expression, then so is H.*

Proof. Follows from a similar result in the standard box algebra. □

3.4 SOS rules

Similarly as at-boxes, at-expressions can perform two kinds of operational semantics moves, namely *action* moves and *time* moves. A time move has the form

$$G \xrightarrow{\surd} H$$

and an action move has the form

$$G \xrightarrow{\Gamma} H$$

where Γ is a finite multiset of communication actions. We now define various types of moves of the structural operational semantics of dynamic at-expressions.

Empty moves

The following rules deal with the empty action moves.

$$\frac{G \equiv H}{G \xrightarrow{\varnothing} H} \qquad \frac{G \xrightarrow{\varnothing} J \xrightarrow{\Gamma} H}{G \xrightarrow{\Gamma} H} \qquad \frac{G \xrightarrow{\Gamma} J \xrightarrow{\varnothing} H}{G \xrightarrow{\Gamma} H}$$

Basic action

A basic action can occur if its timing restrictions are satisfied by the age range of its overbar:

$$\frac{\mathbb{EL} \text{ tsat } el}{\overline{\alpha el}^{\mathbb{EL}} \xrightarrow{\{\alpha\}} \underline{\alpha el}_{00}}$$

Note that the age range of a newly created underbar is always set to (00).

Scoping

There is a single rule for scoping:

$$\frac{G \xrightarrow{\{a_1,\widehat{a}_1\}+\cdots+\{a_k,\widehat{a}_k\}+\Gamma} H \ , \ (A \cup \widehat{A}) \cap \Gamma = \varnothing \ , \ a_1, \ldots, a_k \in A}{G \,\mathsf{sc}\, A \xrightarrow{k\cdot\{\iota\}+\Gamma} H \,\mathsf{sc}\, A}$$

Other operators

There is no real difference in the rules for the remaining operators when compared with the standard PBC [8,9].

$$\frac{G \xrightarrow{\Gamma} G' \;,\; H \xrightarrow{\Gamma'} H'}{G \| H \xrightarrow{\Gamma+\Gamma'} G' \| H'} \qquad \frac{G \xrightarrow{\Gamma} H}{\begin{array}{l} \langle\!\langle G \circledast E \circledast F \rangle\!\rangle \xrightarrow{\Gamma} \langle\!\langle H \circledast E \circledast F \rangle\!\rangle \\ \langle\!\langle E \circledast G \circledast F \rangle\!\rangle \xrightarrow{\Gamma} \langle\!\langle E \circledast H \circledast F \rangle\!\rangle \\ \langle\!\langle E \circledast F \circledast G \rangle\!\rangle \xrightarrow{\Gamma} \langle\!\langle E \circledast F \circledast H \rangle\!\rangle \end{array}}$$

$$\frac{G \xrightarrow{\Gamma} H}{\begin{array}{l} E \Box G \xrightarrow{\Gamma} E \Box H \\ G \Box E \xrightarrow{\Gamma} H \Box E \end{array}} \qquad \frac{G \xrightarrow{\Gamma} H}{\begin{array}{l} G;E \xrightarrow{\Gamma} H;E \\ E;G \xrightarrow{\Gamma} E;H \end{array}}$$

3.4.1 Urgent labels of at-expressions

To identify cases when time moves can be applied, we need the notion of *urgent* labels which can be executed by an at-expression. Urgent labels of dynamic at-expressions are defined by

$$\mathsf{urgent}_{lab}(H) \stackrel{\text{df}}{=} \{\alpha \mid \alpha^0 \in \mathsf{enabled}_{aux}(H)\},$$

where $\mathsf{enabled}_{aux}(H)$ is a set defined by induction on the structure of H. There are two kinds of objects which $\mathsf{enabled}_{aux}(H)$ can contain, namely α^δ and a, where $\alpha \in \mathcal{A} \cup \{\imath\}$, $a \in \mathcal{A}$ and $\delta \in \{0,1\}$. Intuitively, α^0 means that the label α is enabled and urgent in the expression H, α^1 means that the label α is enabled but non-urgent, and a means that there is a pair of conjugate labels $(a, \widehat{a})$ enabled simultaneously and at least one of these labels is urgent. In more detail, for the base case, we have:

$$\mathsf{enabled}_{aux}(\overline{\alpha el}^{\mathbb{EL}}) \stackrel{\text{df}}{=} \begin{cases} \{\alpha^0\} & \text{if } \mathbb{EL} \text{ tsat } el \text{ and } l = \mathbb{L} \\ \{\alpha^1\} & \text{if } \mathbb{EL} \text{ tsat } el \text{ and } l > \mathbb{L} \\ \varnothing & \text{otherwise .} \end{cases}$$

$$\mathsf{enabled}_{aux}(\underline{\alpha el}_{\mathbb{EL}}) \stackrel{\text{df}}{=} \varnothing$$

For more complicated expressions H, we define $\mathsf{enabled}_{aux}(H)$ as the smallest set such that, whenever $H \equiv G$ then

$$\mathsf{enabled}_{aux}(G) = \mathsf{enabled}_{aux}(H)$$

and then the following hold for individual cases of composition operators. For scoping, if $a \in \mathsf{enabled}_{aux}(G)$ and $a \in (A \cup \widehat{A})$ then:

$$\imath^0 \in \mathsf{enabled}_{aux}(G \,\mathsf{sc}\, A)\,,$$

as well as

$$\{\alpha^\delta \in \mathsf{enabled}_{aux}(G) \mid \alpha \notin (A \cup \widehat{A})\} \subseteq \mathsf{enabled}_{aux}(G \,\mathsf{sc}\, A)$$
$$\{a \in \mathsf{enabled}_{aux}(G) \mid a \notin (A \cup \widehat{A})\} \subseteq \mathsf{enabled}_{aux}(G \,\mathsf{sc}\, A)\,.$$

For concurrent composition,

$$\mathsf{enabled}_{aux}(G) \cup \mathsf{enabled}_{aux}(J) \subseteq \mathsf{enabled}_{aux}(G \| J)$$
$$\{a \mid a^\delta \in \mathsf{enabled}_{aux}(G) \wedge \widehat{a}^{\delta'} \in \mathsf{enabled}_{aux}(J) \wedge \delta \cdot \delta' = 0\} \subseteq \mathsf{enabled}_{aux}(G \| J)\,.$$

For the remaining operators, we have that:

$$\begin{aligned}\mathsf{enabled}_{aux}(G) \subseteq\ & \mathsf{enabled}_{aux}(\langle\!\langle G \circledast E \circledast F \rangle\!\rangle) \cap \mathsf{enabled}_{aux}(\langle\!\langle E \circledast G \circledast F \rangle\!\rangle) \\ & \cap \mathsf{enabled}_{aux}(\langle\!\langle E \circledast F \circledast G \rangle\!\rangle)\end{aligned}$$
$$\mathsf{enabled}_{aux}(G) \subseteq \mathsf{enabled}_{aux}(G \Box E) \cap \mathsf{enabled}_{aux}(E \Box G)$$
$$\mathsf{enabled}_{aux}(G) \subseteq \mathsf{enabled}_{aux}(G ; E) \cap \mathsf{enabled}_{aux}(E ; G)\,.$$

Now one can consider the following example expression $(E ; (\overline{a} \parallel \underline{b}))$ to enhance the intuition behind the calculation of objects contained in the equivalence class of $\mathsf{enabled}_{aux}$.

$$\mathsf{enabled}_{aux}((E ; (\overline{a} \parallel \underline{b})) = \mathsf{enabled}_{aux}(\overline{a} \parallel \underline{b}) =$$
$$\mathsf{enabled}_{aux}(\overline{a}) \cup \mathsf{enabled}_{aux}(\underline{b}) = \{a\} \cup \varnothing = \{a\}.$$

3.4.2 Time moves

There is a single time rule:

$$\frac{\mathsf{urgent}_{lab}(G) = \varnothing}{G \xrightarrow{\surd} G^{\surd}}$$

where $G^{\surd}$ is G with each time annotation $\mathbb{EL}$ at an over- or underbar changed to $(\mathbb{E}+1)(\mathbb{L}+1)$. Notice that a time move can only be applied at the topmost level of an expression as it cannot be 'propagated' through the expression using action rules. This ensures that time progresses uniformly.

Note also that to capture the urgency of enabled label, one cannot use a definition of the following kind: $\alpha \in \mathsf{urgent}_{lab}(G)$ if α is enabled by G but not by $G^{\surd}$. The reason is that enabling alone cannot find out precisely which action cannot wait any longer. Take for, instance, the following at-expression: $\overline{a00 \Box a01}^{00}$. We have here two possible occurrences of a leading to the the same expression $\underline{a00 \Box a01}_{00}$. However, one of them should be considered urgent, even though we still have that a is enabled by $\overline{a00 \Box a01}^{11}$.

It can be seen that the rules of the operational semantics do not lead outside the set of dynamic at-expressions.

Proposition 3.2. *Assuming that we treat the rules of the operational semantics as term rewriting rules, and H has been derived from an at-expression, then H is also an at-expression.*

Proof. Follows from a similar result in the standard box algebra. □

3.5 Reachability trees of at-expressions

As already mentioned, we are ultimately interested in those at-expressions that can be reached, through the rules of the structural operational semantics, from static at-expressions started at zero time, i.e., we are interested in at-expressions of the form $G = \overline{E}^{00}$ executed using the operational semantics rules defined earlier in this section. The representation that we will use to capture the behaviour of G will again be a *reachability tree*, denoted by RT_G. Its nodes are labelled by equivalence classes of dynamic expressions reachable from G, and arcs are labelled by multisets over $\mathcal{A} \cup \{\imath\}$ or the $\surd$ symbol. The root node is labelled by $[G]_{\equiv}$ and, if a node is labelled by $[H]_{\equiv}$, then: for every move

$$H \xrightarrow{\Gamma} J,$$

there is a unique descendant labelled by $[J]_{\equiv}$ and the arc leading to it is labelled by Γ, and if the time move is possible for H then there is a unique descendant labelled by $[H^{\surd}]_{\equiv}$ and the arc leading to it is labelled by $\surd$.

Note that we base reachability trees (and later transition systems) of at-expressions on the equivalence classes of $\equiv$, rather than on at-expressions themselves, since we may have $G \xrightarrow{\varnothing} G'$ for two different expressions G and G', whereas in the domain of at-boxes, $\Theta[\varnothing\rangle\Xi$ always implies $\Theta = \Xi$.

3.6 Examples

Our first example, in figure 3.1, shows an at-expression with two sequential actions a, c in parallel with two other sequential actions $b, \widehat{c}$ and scoping on action c. Different execution scenarios can be followed. We choose, in line (2), to execute action a followed by a time move in line (3) which is the only possible move at this stage. Action b becomes then urgent and in line (4) b is executed. After three time moves, in line (6), the c part of enabled synchronisation action is urgent, and so time move is disallowed. Synchronisation takes place in line (7), by executing the silent synchronisation action $\imath$.

The second example, in figure 3.2, shows an at-expression consisting of an action a in parallel with two sequential actions $b, \widehat{a}$ and scoping on action a. In line (2), we cannot execute a due to the restriction imposed by the scoping operator (as well as the timing age) and b is not ready to fire. In line (3), after one time move, action b is urgent and must be executed immediately. In line

$$
\begin{array}{lll}
(1) & \overline{((a02\,;c44) \parallel (b11\,;\widehat{c}14))\,\mathsf{sc}\{c\}}^{00} & \equiv \\
(2) & ((\overline{a02}^{00}\,;c44) \parallel (\overline{b11}^{00}\,;\widehat{c}14))\,\mathsf{sc}\{c\} & \xrightarrow{\{a\}} \\
(3) & ((\underline{a02}_{00}\,;c44) \parallel (\overline{b11}^{00}\,;\widehat{c}14))\,\mathsf{sc}\{c\} & \xrightarrow{\surd} \\
(4) & ((\underline{a02}_{11}\,;c44) \parallel (\overline{b11}^{11}\,;\widehat{c}14))\,\mathsf{sc}\{c\} & \xrightarrow{\{b\}} \\
(5) & ((\underline{a02}_{11}\,;c44) \parallel (\underline{b11}_{00}\,;\widehat{c}14))\,\mathsf{sc}\{c\} & \equiv \\
(6) & ((a02\,;\overline{c44}^{11}) \parallel (b11\,;\overline{\widehat{c}14}^{00}))\,\mathsf{sc}\{c\} & \xrightarrow{\surd\surd\surd} \\
(7) & ((a02\,;\overline{c44}^{44}) \parallel (b11\,;\overline{\widehat{c}14}^{33}))\,\mathsf{sc}\{c\} & \xrightarrow{\{c\}} \\
(8) & ((a02\,;\underline{c44}_{00}) \parallel (b11\,;\underline{\widehat{c}14}_{00}))\,\mathsf{sc}\{c\} & \equiv \\
(9) & \underline{((a02\,;c44) \parallel (b11\,;\widehat{c}14))\,\mathsf{sc}\{c\}}_{00} &
\end{array}
$$

Figure 3.1: An evolution of the expression $\overline{((a02\,;c44) \parallel (b11\,;\widehat{c}14))\,\mathsf{sc}\{c\}}^{00}$.

(5), action a is urgent, but its counterpart $\widehat{a}$ is not enabled due to the time restrictions. As a result, the synchronisation action of the scoping operator is not possible and there are no other possible action moves after that.

$$
\begin{array}{lll}
(1) & \overline{(a11 \parallel (b11\,;\widehat{a}11))\,\mathsf{sc}\{a\}}^{00} & \equiv \\
(2) & (\overline{a11}^{00} \parallel (\overline{b11}^{00}\,;\widehat{a}11))\,\mathsf{sc}\{a\} & \xrightarrow{\surd} \\
(3) & (\overline{a11}^{11} \parallel (\overline{b11}^{11}\,;\widehat{a}11))\,\mathsf{sc}\{a\} & \xrightarrow{\{b\}} \\
(4) & (\overline{a11}^{11} \parallel (\underline{b11}_{00}\,;\widehat{a}11))\,\mathsf{sc}\{a\} & \equiv \\
(5) & (\overline{a11}^{11} \parallel (b11\,;\overline{\widehat{a}11}^{00}))\,\mathsf{sc}\{a\} &
\end{array}
$$

Figure 3.2: An evolution of the expression $\overline{(a00 \parallel (b11\,;\widehat{a}01))\,\mathsf{sc}\{a\}}^{00}$.

Chapter 4

Algebra of at-boxes

In this chapter, we extend the box algebra to at-boxes, by defining compositionally a mapping which, for static at-expressions, returns at-boxes.

Net substitution

The identities of places and transitions will play a key role, especially when defining the transition based SOS semantics of process expressions. As in the standard box algebra, place and transition identities will come in the form of finite labelled trees retracing the operators used to construct a box.

We shall assume that there are two disjoint sets of *basic* place and transition names, $\mathsf{P}_{\mathsf{root}}$ and $\mathsf{T}_{\mathsf{root}}$. Each name $\eta \in \mathsf{P}_{\mathsf{root}} \cup \mathsf{T}_{\mathsf{root}}$ can be viewed as a special tree with a single node labelled with η, which is both a root and a leaf. (All the transitions in figure 4.1 are assumed to be of that kind.) We shall also employ more complex trees as transition and place names, and use a linear notation to express such trees. To this end, an expression $x \lhd \mathsf{T}$, where x is a basic name in $\mathsf{P}_{\mathsf{root}} \cup \mathsf{T}_{\mathsf{root}}$, and T is a set of trees, denotes a tree where the trees of the multiset are appended to an x-labelled root. Moreover,

- if $\mathsf{T} = \{\mathsf{t}\}$ is a singleton then $x \lhd \mathsf{T}$ will be denoted by $x \lhd \mathsf{t}$.
- $x \blacktriangleleft \mathsf{T}$ denotes the set of trees $\{x \lhd \mathsf{t} \mid \mathsf{t} \in \mathsf{T}\}$.
- $x \lhd (v_1 \blacktriangleleft \mathsf{T}_1, \ldots, v_k \blacktriangleleft \mathsf{T}_k)$ denotes the set of trees

$$\{x \lhd \{v_1 \lhd \mathsf{t}_1, \ldots, v_k \lhd \mathsf{t}_k\} \mid \mathsf{t}_1 \in \mathsf{T}_1 \wedge \ldots \mathsf{t}_k \in \mathsf{T}_k\}\,.$$

4.1 Net refinement

The net algebra employs operators directly corresponding to (and denoted as) those used in the algebra of static at-expressions. All the net operators are similar to those in the standard PBC with two important modifications: (i) changing the definition of the basic net corresponding to a single action, and

(ii) taking care of the time restrictions associated with transition input arcs. Essentially, the latter means that if p and t are a place and transition which are 'carried forward' by a net operator, then the associated time constraint $\lambda(p,t)$ is also carried forward. Moreover, in the scoping operation, if t and t' are fused together to yield a z-labelled synchronisation transition u, then we assume that ${}^\bullet t \cap {}^\bullet t' = \varnothing$ and $t^\bullet \cap t'^\bullet = \varnothing$. The full definitions of the composition operators for at-nets are given below. The relevant operator boxes are shown in figure 4.1.

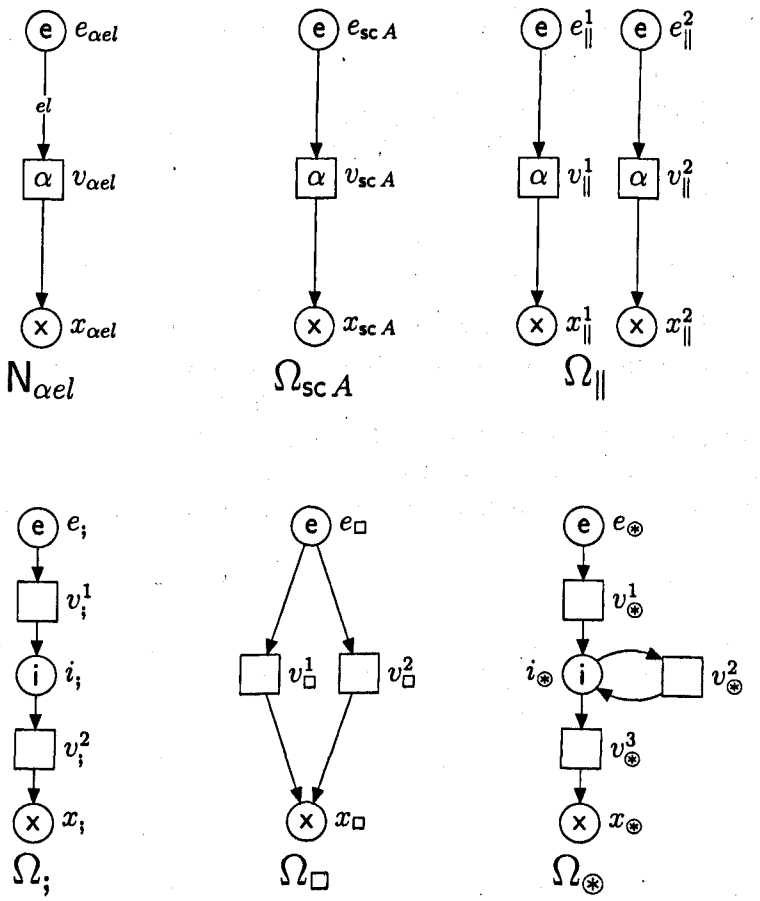


Figure 4.1: An at-net $\mathsf{N}_{\alpha el}$ and five operator boxes.

Scoping

Let $A \subseteq \mathcal{A}$ and Σ be an ex-restricted and ex-directed at-net. The result of a substitution of the transition $v_{\mathsf{sc}\,A}$ in $\Omega_{\mathsf{sc}\,A}$ by Σ is an at-net $\Phi = \Omega_{\mathsf{sc}\,A}(\Sigma)$ whose components are defined as follows.

Places. There are three kinds of places in Φ:

- For every entry place p in Σ, $q = e_{\mathrm{sc}\,A} \lhd v_{\mathrm{sc}\,A} \lhd p$ is an entry place in Φ with the marking $M_\Sigma(p)$.
- For every exit place p in Σ, $q = x_{\mathrm{sc}\,A} \lhd v_{\mathrm{sc}\,A} \lhd p$ is an exit place in Φ with the marking $M_\Sigma(p)$.
- For every internal place p in Σ, $q = v_{\mathrm{sc}\,A} \lhd p$ is an internal place in Φ with the marking $M_\Sigma(p)$.

Transitions, arcs and timing constraints. There are two kinds of transitions in Φ:

- For every transition t in Σ with a label not belonging to $A \cup \widehat{A}$, $w = v_{\mathrm{sc}\,A} \lhd t$ is a transition in Φ with the same label as t.
  There is an arc from a place q to w iff there was an arc from p to t; moreover, in such a case, $\lambda_\Phi(q, w) = \lambda_\Sigma(p, t)$.
  There is an arc from w to a place q iff there was an arc from t to p.
- For all pairs of transitions t, u in Σ, one with a label $a \in A$ and the other with the label $\widehat{a}$, as well as with disjoint sets of pre- and post-places, $w = v_{\mathrm{sc}\,A} \lhd \{t, u\}$ is a transition in Φ with the label $\imath$.
  There is an arc from a place q to w iff there was an arc from p to t (or u); moreover, in such a case, $\lambda_\Phi(q, w) = \lambda_\Sigma(p, t)$ (or $\lambda_\Phi(q, w) = \lambda_\Sigma(p, u)$).[1]
  There is an arc from w to a place q iff there was an arc from t or u to p.

Other operators

Let $\Omega_{op} \in \{\Omega_\Box, \Omega_\circledast, \Omega_;, \Omega_\|\}$ be any n-unary ($n \geq 2$) operator box and $\Sigma = (\Sigma_1, \ldots, \Sigma_n) = (\Sigma_{v^1_{op}}, \ldots, \Sigma_{v^n_{op}})$ be an n-tuple of ex-restricted and ex-directed at-nets. The result of a simultaneous substitution of the transitions v^i_{op} in Ω_{op} by the at-nets $\Sigma_{v^i_{op}}$ is a net $\Omega_{op}(\Sigma) = \Phi$ whose components are defined as follows.

Transitions. There is one kind of transition in Φ:

- For all transitions v in Ω_{op} and t in Σ_v, $w = v \lhd t$ is a transition in Φ with the same label as t.

Places, arcs and timing constraints. There are two kinds of places in Φ:

- For every transition z in Ω_{op} and every internal place p in Σ_z, $q = z \lhd p$ is an internal place in Φ with the marking $M_{\Sigma_z}(p)$.
  There is an arc from q to a transition w iff $v = z$ and there was an arc from p to t; moreover, in such a case, $\lambda_\Phi(q, w) = \lambda_{\Sigma_z}(p, t)$.
  There is an arc from a transition w to q iff $v = z$ and there was an arc from t to p.

[1] Note that the definition is well-formed since the pre-sets of t and u are disjoint.

- For every place s in Ω_{op} with ${}^\bullet s = \{u_1, \ldots, u_k\}$ and $s^\bullet = \{u_{k+1}, \ldots, u_{k+m}\}$, we construct in Φ all the places

$$q = s \lhd (u_1 \lhd p_1, \ldots, u_{k+m} \lhd p_{k+m}) ,$$

where each p_i (for $i \leq k$) is an exit place of Σ_{u_i}, and each p_j (for $j > k$) is an entry place of Σ_{u_j}.
The label of q is that of s and the marking is equal to

$$M_{\Sigma_{u_1}}(p_1) + \ldots + M_{\Sigma_{u_{k+m}}}(p_{k+m}) .$$

There is an arc from q to a transition w iff $w = u_j$ (for some j) and there was an arc from p_j to t; moreover, in such a case, $\lambda_\Phi(q, w) = \lambda_{\Sigma_w}(p_j, t)$.[2]
There is an arc from a transition w to q iff $w = u_j$ (for some j) and there was an arc from t to p_j.

As in the standard box algebra, we will use the following notations:

$$\begin{aligned}
\Omega_{\mathsf{sc}\,A}(\Sigma) &= \Sigma\,\mathsf{sc}\,A \\
\Omega_{\|}(\Sigma, \Sigma') &= \Sigma \| \Sigma' \\
\Omega_{;}(\Sigma, \Sigma') &= \Sigma ; \Sigma' \\
\Omega_{\Box}(\Sigma, \Sigma') &= \Sigma \Box \Sigma' \\
\Omega_{\circledast}(\Sigma, \Sigma', \Sigma'') &= \langle\!\langle \Sigma \circledast \Sigma' \circledast \Sigma'' \rangle\!\rangle .
\end{aligned}$$

4.1.1 Composite at-nets

To be able to take advantage of the results developed for the standard box algebra, we introduce semantics of at-expressions into at-nets which are the same as that in the standard box algebra if we ignore all time annotations (nets are marked with black tokens). The mapping Box from at-expressions to at-nets is defined so that

$$\mathsf{Box}(\alpha el) \stackrel{\mathrm{df}}{=} \mathsf{N}_{\alpha el}$$

where $\mathsf{N}_{\alpha el}$ is shown in figure 4.1,

$$\begin{aligned}
\mathsf{Box}(\overline{E}^{\mathbf{EL}}) &\stackrel{\mathrm{df}}{=} \overline{\mathsf{Box}(E)} \\
\mathsf{Box}(\underline{E}_{\mathbf{EL}}) &\stackrel{\mathrm{df}}{=} \underline{\mathsf{Box}(E)}
\end{aligned}$$

and for the remaining static and dynamic at-expressions:

$$\begin{aligned}
\mathsf{Box}(H\,\mathsf{sc}\,A) &\stackrel{\mathrm{df}}{=} \mathsf{Box}(H)\,\mathsf{sc}\,A \\
\mathsf{Box}(H \Box J) &\stackrel{\mathrm{df}}{=} \mathsf{Box}(H) \Box \mathsf{Box}(J) \\
\mathsf{Box}(H \| J) &\stackrel{\mathrm{df}}{=} \mathsf{Box}(H) \| \mathsf{Box}(J) \\
\mathsf{Box}(H ; J) &\stackrel{\mathrm{df}}{=} \mathsf{Box}(H) ; \mathsf{Box}(J) \\
\mathsf{Box}(\langle\!\langle H \circledast J \circledast I \rangle\!\rangle) &\stackrel{\mathrm{df}}{=} \langle\!\langle \mathsf{Box}(H) \circledast \mathsf{Box}(J) \circledast \mathsf{Box}(I) \rangle\!\rangle .
\end{aligned}$$

[2]Note that the definition is well-formed since the operand at-nets are ex-directed.

Any at-net obtained through the Box() from some at-expression will be called *composite*. Note that the above at-nets semantics of at-expressions are the standard black token semantics, with all time constraint being simply ignored.

Proposition 4.1. *For every static (or dynamic) at-expression H, $\mathsf{Box}(H)$ is a static (resp. dynamic) at-net which is both ex-directed and ex-restricted. Moreover, if H conforms to the syntax for Z or K then $\mathsf{Box}(H)$ is ex-exclusive.*

Proof. Follows from similar results in the standard box algebra. □

4.2 Transition based operational semantics of at-expressions

To prove our main results, we will need another semantics of at-expressions, based on the transitions present in the corresponding composite at-nets. More precisely, at-expressions can perform two kinds of operational semantics moves, namely *action* moves and *time* moves. A time move has the form

$$G \xrightarrow{\surd} H$$

and an action move has the form

$$G \xrightarrow{U} H$$

where $U = \{t_1, \ldots, t_k\}$ $(k \geq 0)$ is a set of transitions in the composite at-net $\mathsf{Box}(E)$, where E is obtained from G by deleting all overbars and underbars.

We now define various types of moves of the structural operational semantics of dynamic at-expressions (note that the relation $\equiv$ below is defined as in table 3.1).

Empty moves

The following rules deal with the empty action moves.

$$\frac{G \equiv H}{G \xrightarrow{\varnothing} H} \qquad \frac{G \xrightarrow{\varnothing} J \xrightarrow{U} H}{G \xrightarrow{U} H} \qquad \frac{G \xrightarrow{U} J \xrightarrow{\varnothing} H}{G \xrightarrow{\Gamma} H}$$

Basic action

A basic action can occur if its timing restrictions are satisfied by the age range of its overbar:

$$\frac{\mathbb{EL}\ \mathsf{tsat}\ el}{\overline{\alpha el}^{\mathbb{EL}} \xrightarrow{\{v_{\alpha el}\}} \underline{\alpha el}_{00}}$$

Scoping

There is a single rule for scoping:

$$\frac{G \xrightarrow{\{t_1,u_1\}\uplus\cdots\uplus\{t_k,u_k\}\uplus U} H \ , \ (\forall i) a_i = \widehat{c_i} \in A \ , \ (A \cup \widehat{A}) \cap L = \varnothing}{G \,\mathsf{sc}\, A \xrightarrow{\{v_{\mathsf{sc}\,A} \lhd \{t_1,u_1\},\ldots,v_{\mathsf{sc}\,A} \lhd \{t_k,u_k\}\} \cup v_{\mathsf{sc}\,A} \blacktriangleleft U} H \,\mathsf{sc}\, A}$$

where $L = \lambda_{\mathsf{Box}(\lfloor G \rfloor)}(U)$, $a_i = \lambda_{\mathsf{Box}(\lfloor G \rfloor)}(t_i)$ and $c_i = \lambda_{\mathsf{Box}(\lfloor G \rfloor)}(u_i)$, for $i = 1, \ldots, k$.

Other operators

There is no real difference in the rules for the remaining operators when compared with the standard box algebra [8, 9].

$$\frac{G \xrightarrow{U} G' \ , \ H \xrightarrow{U'} H'}{G \| H \xrightarrow{v_{\|}^1 \blacktriangleleft U \cup v_{\|}^2 \blacktriangleleft U'} G' \| H'}$$

$$\frac{G \xrightarrow{U} H}{\begin{array}{l} \langle\!\langle G \circledast E \circledast F \rangle\!\rangle \xrightarrow{v_{\circledast}^1 \blacktriangleleft U} \langle\!\langle H \circledast E \circledast F \rangle\!\rangle \\ \langle\!\langle E \circledast G \circledast F \rangle\!\rangle \xrightarrow{v_{\circledast}^2 \blacktriangleleft U} \langle\!\langle E \circledast H \circledast F \rangle\!\rangle \\ \langle\!\langle E \circledast F \circledast G \rangle\!\rangle \xrightarrow{v_{\circledast}^3 \blacktriangleleft U} \langle\!\langle E \circledast F \circledast H \rangle\!\rangle \end{array}}$$

$$\frac{G \xrightarrow{U} H}{\begin{array}{l} G \Box E \xrightarrow{v_{\Box}^1 \blacktriangleleft U} H \Box E \\ E \Box G \xrightarrow{v_{\Box}^2 \blacktriangleleft U} E \Box H \end{array}}$$

$$\frac{G \xrightarrow{\Gamma} H}{\begin{array}{l} G ; E \xrightarrow{v_{;}^1 \blacktriangleleft U} H ; E \\ E ; G \xrightarrow{v_{;}^2 \blacktriangleleft U} E ; H \end{array}}$$

4.2.1 Urgent transitions of at-expressions

Urgent transitions of dynamic at-expressions are defined by induction on their structure, as follows. For the base case, we have:

$$\begin{array}{rcl} \mathsf{urgent}(\overline{\alpha el}^{\mathbb{EL}}) & \stackrel{\mathrm{df}}{=} & \begin{cases} \{v_{\alpha el}\} & \text{if } \mathbb{EL} \text{ tsat } el \text{ and } l = \mathbb{L} \\ \varnothing & \text{otherwise .} \end{cases} \\ \mathsf{urgent}(\underline{\alpha el}_{\mathbb{EL}}) & \stackrel{\mathrm{df}}{=} & \varnothing \end{array}$$

For more complicated expressions H, we define $\mathsf{urgent}(H)$ as the smallest set such that, whenever $H \equiv G$ then

$$\mathsf{urgent}(G) = \mathsf{urgent}(H)$$

and then the following hold for individual cases of composition operators. For scoping, if $v_{\mathsf{sc}\,A} \lhd U \in \mathsf{enabled}(G)$ and $U \cap \mathsf{urgent}(G) \neq \varnothing$ then:

$$v_{\mathsf{sc}\,A} \lhd U \in \mathsf{urgent}(G \,\mathsf{sc}\, A) \, .$$

Note: $\mathsf{enabled}(H)$ comprises all t such that there is an at-expression J satisfying

$$H \xrightarrow{\{t\}} J\,.$$

For the remaining operators, if $t \in \mathsf{urgent}(G)$ then:

$$\begin{array}{rcl}
v^1_{\|} \lhd t & \in & \mathsf{urgent}(G \| J) \\
v^2_{\|} \lhd t & \in & \mathsf{urgent}(J \| G) \\
v^1_{\circledast} \lhd t & \in & \mathsf{urgent}(\langle\!\langle G \circledast E \circledast F \rangle\!\rangle) \\
v^2_{\circledast} \lhd t & \in & \mathsf{urgent}(\langle\!\langle E \circledast G \circledast F \rangle\!\rangle) \\
v^3_{\circledast} \lhd t & \in & \mathsf{urgent}(\langle\!\langle E \circledast F \circledast G \rangle\!\rangle) \\
v^1_{\Box} \lhd t & \in & \mathsf{urgent}(G \Box E) \\
v^2_{\Box} \lhd t & \in & \mathsf{urgent}(E \Box G) \\
v^1_{;} \lhd t & \in & \mathsf{urgent}(G\,;E) \\
v^2_{;} \lhd t & \in & \mathsf{urgent}(E\,;G)\,.
\end{array}$$

4.2.2 Time moves

There is a single time rule:

$$\boxed{\frac{\mathsf{urgent}(G) = \varnothing}{G \xrightarrow{\surd} G^{\surd}}}$$

Note that $\mathsf{urgent}(G)$ is the set of all transitions enabled by G but not by $G^{\surd}$ and, in fact, it could be defined like that. However, we preferred to give a definition closer to that used in the label based presentation in the previous chapter of this thesis. Note also that the example motivating a rather complicated definition of urgent labels there, $\overline{a00 \Box a01}^{11}$, no longer works. The reason is that in case of the transition based semantics, the two a labels correspond to executing $v^1_{\Box} \lhd v_{a00}$ and $v^2_{\Box} \lhd v_{a01}$, respectively, and so they can be distinguished by the enabling relation.

It can be seen that the rules of the operational semantics do not lead outside the set of dynamic at-expressions.

Proposition 4.2. *Assuming that we treat the rules of the transition based operational semantics as term rewriting rules, and H has been derived from an at-expression, then H is also an at-expression.*

Proof. Follows from a similar result in the standard box algebra. □

Representing global behaviour of at-expressions

There are different, though closely related, representations capturing the overall behaviour of an at-expression H. The first one we already introduced is that of reachability tree, RT_H. We will also need the following.

- A *full reachability tree* of a dynamic at-expression H, denoted by fRT_H, is a tree whose nodes are labelled by equivalence classes of dynamic expressions reachable from H using the rules defined in this section, and arcs are labelled by steps of transitions or the $\surd$ symbol. The root node is labelled by $[H]_{\equiv}$ and, if a node is labelled by $[G]_{\equiv}$, then: for every move

$$G \xrightarrow{U} J,$$

there is a unique descendant labelled by $[J]_{\equiv}$ and the arc leading to it is labelled by U, and if the time move is possible for G then there is a unique descendant labelled by $[G^{\surd}]_{\equiv}$ and the arc leading to it is labelled by $\surd$. For a static at-expression H, $\mathsf{fRT}_H \stackrel{\text{df}}{=} \mathsf{fRT}_{\overline{H}^{00}}$.

- Let H be a dynamic at-expression. We will use $[H\rangle$ to denote all the at-expressions derivable from H using the operational semantics defined in this section, i.e., the least set of expressions containing H such that if $H' \in [H\rangle$ and $H' \xrightarrow{U} H''$, for some step U of transitions in $\llbracket \mathsf{cBox}(H) \rrbracket$, then $H'' \in [H\rangle$. Moreover, $[H]_{\equiv}$ will denote the equivalence class of $\equiv$ containing H.
The *full transition system* of H is then defined as $\mathsf{fTS}_H \stackrel{\text{df}}{=} (V, Arcs, init)$, where $V \stackrel{\text{df}}{=} \{[H']_{\equiv} \mid H' \in [H\rangle\}$ is the set of states with $init \stackrel{\text{df}}{=} [H]_{\equiv}$ being the initial state, and $Arcs$ is the set of labelled arcs of the form $([H']_{\equiv}, U, [H'']_{\equiv})$ such that $H', H'' \in [H\rangle$ and $H' \xrightarrow{U} H''$.
For a static at-expression H, $\mathsf{fTS}_H \stackrel{\text{df}}{=} \mathsf{fTS}_{\overline{H}^{00}}$.

- Let H be a dynamic at-expression. We will use $[H\rangle_{lab}$ to denote all the at-expressions derivable from H using the operational semantics introduced in the main body of the paper, i.e., the least set of expressions containing H such that if $H' \in [H\rangle_{lab}$ and $H' \xrightarrow{\Gamma} H''$, for some multiset of communication labels Γ, then $H'' \in [H\rangle_{lab}$.
The *transition system* of H is then defined as $\mathsf{TS}_H \stackrel{\text{df}}{=} (V, Arcs, init)$, where $V \stackrel{\text{df}}{=} \{[H']_{\equiv} \mid H' \in [H\rangle_{lab}\}$ is the set of states with $init \stackrel{\text{df}}{=} [H]_{\equiv}$ being the initial state, and $Arcs$ is the set of labelled arcs of the form $([H']_{\equiv}, \Gamma, [H'']_{\equiv})$ such that $H', H'' \in [H\rangle_{lab}$ and $H' \xrightarrow{\Gamma} H''$.
For a static at-expression H, $\mathsf{TS}_H \stackrel{\text{df}}{=} \mathsf{TS}_{\overline{H}^{00}}$.

4.3 Interface regions

The standard boxes have quite regular internal structure which then has a significant impact on their behaviour. We will capture some aspects of this structure through the notion of interface regions, which will form a partition of the set of internal places.

The set of *interface regions* $\mathbb{IR}(\Sigma)$ of a composite at-net Σ is defined by induction on the structure of the at-net, in the following way.

Basic Net: $\Sigma = \mathsf{N}_{\alpha el}$. Then $\mathbb{IR}(\Sigma) \stackrel{\text{df}}{=} \varnothing$.

Parallel composition: $\Sigma = \Sigma_1 \| \Sigma_2$. Then

$$\mathbb{IR}(\Sigma) \stackrel{\text{df}}{=} \bigcup_{k=1}^{2} \{v_{\|}^{k} \blacktriangleleft Q \mid Q \in \mathbb{IR}(\Sigma_k)\} .$$

Sequential composition: $\Sigma = \Sigma_1 ; \Sigma_2$. Then

$$\mathbb{IR}(\Sigma) \stackrel{\text{df}}{=} \{i_; \triangleleft (v_;^1 \blacktriangleleft \Sigma_1^\circ, v_;^2 \blacktriangleleft {}^\circ\Sigma_2)\} \cup \bigcup_{k=1}^{2} \{v_;^k \blacktriangleleft Q \mid Q \in \mathbb{IR}(\Sigma_k)\} .$$

Choice operator: $\Sigma = \Sigma_1 \Box \Sigma_2$. Then

$$\mathbb{IR}(\Sigma) \stackrel{\text{df}}{=} \bigcup_{k=1}^{2} \{v_{\Box}^{k} \blacktriangleleft Q \mid Q \in \mathbb{IR}(\Sigma_k)\} .$$

Iteration: $\Sigma = \langle\!\langle \Sigma_1 \circledast \Sigma_2 \circledast \Sigma_3 \rangle\!\rangle$. Then

$$\mathbb{IR}(\Sigma) \stackrel{\text{df}}{=} \{i_{\circledast} \triangleleft (v_{\circledast}^1 \blacktriangleleft \Sigma_1^\circ, v_{\circledast}^2 \blacktriangleleft {}^\circ\Sigma_2, v_{\circledast}^2 \blacktriangleleft \Sigma_2^\circ, v_{\circledast}^3 \blacktriangleleft {}^\circ\Sigma_3)\} \cup \bigcup_{k=1}^{3} \{v_{\circledast}^{k} \blacktriangleleft Q \mid Q \in \mathbb{IR}(\Sigma_k)\} .$$

Scoping: $\Sigma = \Sigma_1 \,\mathsf{sc}\, A$. Then $\mathbb{IR}(\Sigma) \stackrel{\text{df}}{=} \{v_{\mathsf{sc}\,A} \blacktriangleleft Q \mid Q \in \mathbb{IR}(\Sigma_1)\}$.

Proposition 4.3. *Let Σ be a composite at-net. Then*

$$\ddot{\Sigma} = \biguplus_{Q \in \mathbb{IR}(\Sigma)} Q .$$

Proof. Follows by a straightforward induction on the way Σ has been constructed. □

A crucial property of an interface region is that its marking behaves in a monotone way, as captured by the following result.

Proposition 4.4. *Let Σ be an initial composite at-net, $Q \in \mathbb{IR}(\Sigma)$ one of its interface regions, and $M_1 U_1 M_2 U_2 \ldots M_n U_n M_{n+1}$ be a sequence of markings and steps such that $M_1 = M_\Sigma = {}^\circ\Sigma$ and $M_i[U_i\rangle M_{i+1}$, for $i = 1, \ldots, n$. Moreover, let $M_i' = M_i \cap Q$, for $i = 1, \ldots, n+1$.*

1. *There are indices* $k_1 < k_2 < \ldots < k_m$ *such that* $k_1 = 1$, $k_m = n+2$ *and, for each* $j < m$, *one of the following holds:*

$$\textit{Case 1:} \quad \varnothing = M'_{k_j} \subseteq M'_{k_j+1} \subseteq \cdots \subseteq M'_{k_{j+1}-1} \cdot$$
$$\textit{Case 2:} \quad Q = M'_{k_j} \supseteq M'_{k_j+1} \supseteq \cdots \supseteq M'_{k_{j+1}-1} \cdot$$

Moreover the two cases strictly alternate, beginning with Case 1.

2. *If* M'_i *occurs in Case 1 sequence then* ${}^\bullet U_i \cap Q = \varnothing$, *and otherwise* $U_i^\bullet \cap Q = \varnothing$.

Proof. (1) This is a property of the standard box algebra. It can be shown, for instance, by considering the isomorphism between the reachability graphs of such boxes and the the corresponding process expressions. One also needs the following property $\varnothing \in \{{}^\bullet U_i \cap Q, U_i^\bullet \cap Q\}$, for all i, which holds due to the syntaxes (3.1,3.2); in particular, since the way in which the syntax for Z was given guarantees that the corresponding at-net is ex-exclusive.

(2) Follows directly from part (1) and the above property. □

4.3.1 Example

After the definition of interface regions and their technical details it seems useful to clarify their notion and functionality and also present the interface regions for the following example in figure 4.2. For our purposes, we need a more refined view of the structure of algebraically defined nets. In particular, according to their definition interface regions are sets of internal places. These places 'behave' as if they were a single place in the sense that they start from being all empty, they can be subsequently filled in a 'monotonic' fashion (no token removal is allowed before they are completely full), and after that emptied in a 'monotonic' fashion. Essentially, this is described in cases 1,2 in proposition 4.4. This allows us to have a good insight into the manner composite nets evolve. Interface regions are created by both sequential composition and iteration, and carried forward by every operation. Now, let us consider the following at-expression

$$\langle\!\langle (a\,;b) \parallel c \circledast d \circledast e \Box (f\,;g) \rangle\!\rangle$$

and the at-box that corresponds to this expression is presented in figure 4.2. Notice that time annotations have been omitted from the at-expression and the corresponding at-box since they are not necessary for this example. We will now compute the interface regions for this at-box. In the first part of the iteration operator, we have the parallel composition of the sequence of action a and b with action c. Like mentioned before, interface regions are created when we have sequential composition. Following the definition, place p_4 is an interface region. Likewise in the third part of the iteration operator, we have the choice between action e and the sequence of actions f and g. Place p_6 is also an interface

region. Finally, places $\{p2, p5\}$ are also an interface region. In this case, we can also observe the monotonic filling and emptying of an interface region. At the beginning both p_2 and p_5 are empty and then either p_2 or p_5 will get a token. This token cannot be removed before both places are filled.

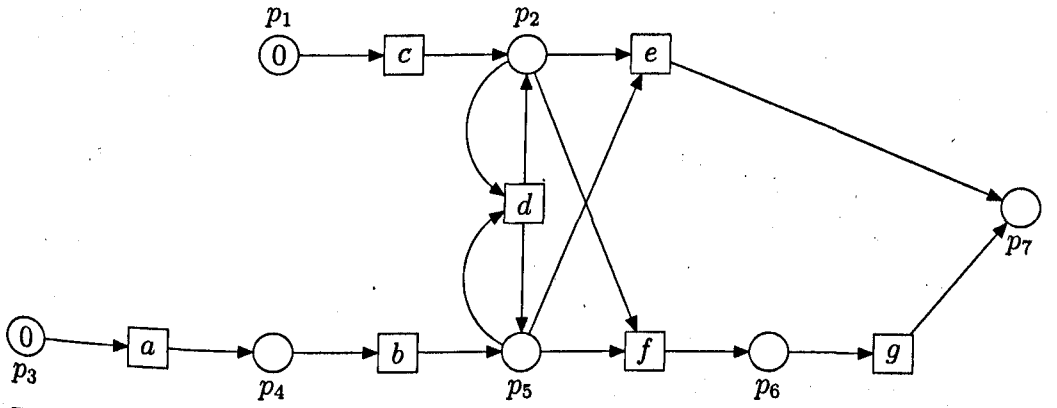


Figure 4.2: An at-box corresponding to the expression $\langle\langle (a\,;b) \parallel c \circledast d \circledast e \square (f\,;g) \rangle\rangle$.

Chapter 5

A new type of timed-arc petri nets

In this chapter, we will introduce a different kind of timed-arc Petri boxes, together with the reasons for this introduction. Furthermore, there will be a presentation of the translation of at-expressions to this new type of timed-arc boxes.

5.1 Token based timed-arc Petri nets

An *at-box* is a pair $\Theta \stackrel{\text{df}}{=} (\Sigma, \mu)$ such that $\Sigma = \text{box}(J)$, for some static or dynamic at-expression J given by the syntax (3.1,3.2) and

$$\mu : P_\Sigma \to \mathbb{N}^\perp$$

is a *token timing* mapping (a state) such that the following consistency conditions hold:

- For every $p \in P_\Sigma$, $\mu(p) = \perp$ iff $M_\Sigma(p) = 0$.
- For all $p, p' \in {}^\circ\Sigma$, if $\mu(p) \neq \perp \neq \mu(p')$ then $\mu(p) = \mu(p')$.

We say that Θ is static/dynamic if so is J and denote $\Theta \in \mathfrak{T}_J$. We then introduce some useful notations:

- $[\![\Theta]\!] \stackrel{\text{df}}{=} \Sigma$ and $\lfloor\Theta\rfloor \stackrel{\text{df}}{=} (\lfloor\Sigma\rfloor, \nu)$, where ν always returns $\perp$.
- The state $\mu^\surd$ is defined so that, for every $p \in P_\Sigma$,

$$\mu^\surd(p) \stackrel{\text{df}}{=} \begin{cases} \mu(p) + 1 & \text{if } \mu(p) \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

  and the at-box $\Theta^\surd$ is then defined as $(\Sigma, \mu^\surd)$.

- Θ is *input-reachable* if it is reachable from the at-box $(\overline{\lfloor\Sigma\rfloor}, \nu)$, where ν returns 0 for all the entry places, and otherwise $\perp$. We will be interested only in those at-boxes which are input-reachable.

The above notions are well-defined. Indeed, it is clear that the two consistency conditions are satisfied in each case.

Proposition 5.1. *Let Θ be an at-box in $\mathfrak{T}_J$.*

1. *$\lfloor\Theta\rfloor$ is a static at-box in $\mathfrak{T}_{\lfloor J\rfloor}$.*
2. *If Θ is static, then $\lfloor\Theta\rfloor = \Theta$.*

Proof. Follows from the properties of the standard box algebra. □

A set of transitions $U \subseteq T_\Sigma$ is *enabled* by Θ if it is enabled by Σ and, for every $t \in U$ and every place $p \in {}^\bullet t$, we have that $\mu(p)$ tsat $\lambda_\Sigma(p,t)$. We denote this by $U \in \mathsf{enabled}(\Theta)$. This enabling is *urgent*, denoted $U \in \mathsf{urgent}(\Theta)$, if U is not enabled by $\Theta^{\surd}$.

An enabled step may be *executed* and yield a *follower* at-box $\Xi = (\Sigma', \nu)$ such that $\Sigma[U\rangle\Sigma'$ and, for every place $p \in P_\Sigma$,

$$\nu(p) \stackrel{\text{df}}{=} \begin{cases} \bot & \text{if } p \in {}^\bullet U \\ 0 & \text{if } p \in U^\bullet \\ \mu(p) & \text{otherwise .} \end{cases}$$

We denote this by $\Theta[U\rangle\Xi$. Note that due to proposition 4.4 and the ex-directedness of Σ, we do not need to consider the case when $p \in {}^\bullet U \cap U^\bullet$. A similar comment applies also to the formula for marking execution in the cat-boxes introduced in the next section.

A time move is enabled if there is no urgent enabled step; it then can be executed and yield a follower at-box: $\Theta[\surd\rangle\Theta^{\surd}$.

Proposition 5.2. *Let Θ be an at-box and $\Theta[U\rangle\Xi$ or $\Theta[\surd\rangle\Xi$.*

1. *If Θ is static, then $U = \varnothing$ and $\Theta = \Xi$.*
2. *If Θ is dynamic then so is Ξ.*

Proof. Follows from the properties of the standard box algebra and, additionally, we need to check that the two consistency conditions from the definition of at-boxes are satisfied. The latter is straightforward (ex-directedness of at-nets is important here). □

Proposition 5.3. *Let Θ be an input-reachable at-box, $\Theta[U\rangle\Xi$, where U is a step consisting of transitions $t_1, \ldots, t_k$. Then there are at-boxes $\Theta_0, \ldots, \Theta_k$ such that $\Theta_0 = \Theta$, $\Theta_k = \Xi$ and $\Theta_{i-1}[t_i\rangle\Theta_i$, for $i = 1, \ldots, k$.*

Proof. Follows from the standard properties of safe Petri nets and proposition 4.4 which ensures that for each t_i no time token involved in the enabling of t_i is involved in the firing of the preceding transitions $t_1, \ldots, t_{i-1}$. □

5.1.1 Representing global behaviour of at-boxes

As in the case of at-expressions, there are different representations capturing the overall behaviour of an at-box Θ. The first one we already introduced is that of reachability tree, RT_Θ. We will also need the following.

- A *full reachability tree* of an at-box $\Theta = (\Sigma, \mu)$, denoted by fRT_Θ, has nodes labelled by token timings and arcs annotated by executed transition steps or time moves. More precisely, the root node is labelled by the initial token timing μ and, if a node is labelled by μ', then for every move $\mu'[x\rangle\mu''$ there is a unique descendant labelled by μ''; the arc leading to it is labelled by $\surd$ if $x = \surd$, and by U if $x = U$ is an executed transition step.

- A *full transition system* of an at-box Θ is $\mathsf{fTS}_\Theta \stackrel{\text{df}}{=} (V, Arcs, init)$, where $V \stackrel{\text{df}}{=} [\Theta\rangle$ is the set of states with $init \stackrel{\text{df}}{=} \Theta$ being the initial state, and $Arcs$ is the set of all labelled arcs of the form (Θ', U, Θ'') and $(\Theta', \surd, \Theta'')$ such that $\Theta', \Theta'' \in [\Theta\rangle$ and, respectively, $\Theta'[U\rangle\Theta''$ and $\Theta'[\surd\rangle\Theta''$.

- A *transition system* of an at-box Θ, denoted by ts_Θ, is obtained from fTS_Θ by replacing each arc $\Theta'[U\rangle\Theta''$ by $\Theta'[\Gamma\rangle\Theta''$, where Γ is the multiset of communication labels of the transitions in U.

5.2 Preparing for the main result proof

5.2.1 Why reachability trees?

In the original PBC the main result stated that the reachability graphs of expressions and the corresponding boxes are isomorphic. Unfortunately such result cannot hold in this case as it can be seen in theorem 6.4. The main result of this thesis is formulated in terms of reachability trees because the reachability graphs are not isomorphic, though they are strongly bisimilar [46]. Isomorphism of reachability graphs fails to hold because, in general, there is no one-to-one correspondence between the expressions reachable from G and the token timings reachable from the initial token timing of $\mathsf{Box}(G)$. To illustrate this, we consider the at-expression $G = \overline{((a00 \parallel b01) \parallel c11)\,;d01}^{00}$ and the corresponding at-box $\mathsf{Box}(G)$ shown in figure 5.1.

It may be easily checked that this at-box allows the following two sequences

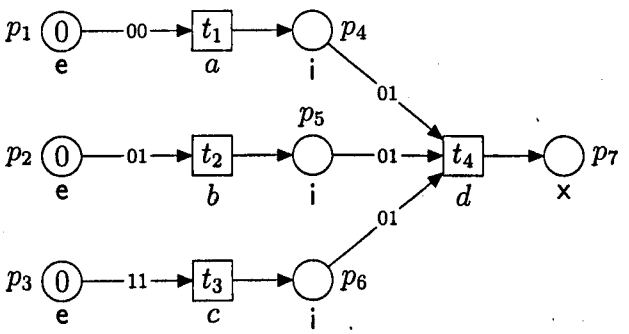


Figure 5.1: An at-box corresponding to the expression $\overline{((a00 \parallel b01) \parallel c11)\,;d01}^{00}$.

of moves, both starting from the initial token timing:

	scenario1		*scenario2*	
(1)	$(0,0,0,\bot,\bot,\bot,\bot)$	$[\{t_1,t_2\}\rangle$	$(0,0,0,\bot,\bot,\bot,\bot)$	$[\{t_1\}\rangle$
(2)	$(\bot,\bot,0,0,0,\bot,\bot)$	$[\surd\rangle$	$(\bot,0,0,0,\bot,\bot,\bot)$	$[\surd\rangle$
(3)	$(\bot,\bot,1,1,1,\bot,\bot)$	$[\{t_3\}\rangle$	$(\bot,1,1,1,\bot,\bot,\bot)$	$[\{t_2,t_3\}\rangle$
(4)	$(\bot,\bot,\bot,1,1,0,\bot)$	$[\{t_4\}\rangle$	$(\bot,\bot,\bot,1,0,0,\bot)$	$[\{t_4\}\rangle$
(5)	$(\bot,\bot,\bot,\bot,\bot,\bot,0)$		$(\bot,\bot,\bot,\bot,\bot,\bot,0)$	

The two corresponding execution sequences for the expression G are shown in figure 5.2. One may further observe that the left marking in line (4) above corresponds to the expressions in lines $(4')$ and $(4a')$, and that the right marking in line (4) above corresponds to the expressions in lines $(4'')$ and $(4a'')$. However, the two markings are different yet we have $(4') \equiv (4a') = (4a'') \equiv (4'')$, which indicates that the expressions in lines $(4', 4a', 4'', 4a'')$ represent the same state of the system. It is therefore impossible to show that the reachability graphs of G and $\mathsf{Box}(G)$ are isomorphic. This should not be treated as a cause for concern the main results of this thesis still establish very strong relationship between the behaviours of the at-expressions and the corresponding at-boxes. The above discussion also means that a proof of the main results presented in this research cannot be obtained by a simple adaptation of that used in [9] since dynamic at-expressions cannot be unambiguously mapped to at-boxes. In the following sections we will explain how we cope with this problem.

5.2.2 Clusters

For every composite at-net Σ, its *clusters* are defined as:

$$CL(\Sigma) \stackrel{\text{df}}{=} \{^\circ\Sigma, \Sigma^\circ\} \cup \mathsf{cl}_\mathsf{e}(\Sigma) \cup \mathsf{cl}_\mathsf{i}(\Sigma),$$

where the entry clusters $\mathsf{cl}_\mathsf{e}(\Sigma)$, and the internal clusters $\mathsf{cl}_\mathsf{i}(\Sigma)$, are defined compositionally below.

scenario1

$$
\begin{array}{lll}
(1') & \overline{((a00 \parallel b01) \parallel c11)\,;d01}^{\,00} & \xrightarrow{\{a,b\}} \\
(2') & ((\underline{a00}_{00} \parallel \underline{b01}_{00}) \parallel \overline{c11}^{\,00})\,;d01 & \xrightarrow{\surd} \\
(3') & ((\underline{a00}_{11} \parallel \underline{b01}_{11}) \parallel \overline{c11}^{\,11})\,;d01 & \xrightarrow{\{c\}} \\
(4') & ((\underline{a00}_{11} \parallel \underline{b01}_{11}) \parallel \underline{c11}_{00})\,;d01 & \equiv \\
(4a') & ((a00 \parallel b01) \parallel c11)\,;\overline{d01}^{\,01} & \xrightarrow{\{d\}} \\
(5') & \underline{((a00 \parallel b01) \parallel c11)\,;d01}_{00} &
\end{array}
$$

scenario2

$$
\begin{array}{lll}
(1'') & \overline{((a00 \parallel b01) \parallel c11)\,;d01}^{\,00} & \xrightarrow{\{a\}} \\
(2'') & ((\underline{a00}_{00} \parallel \overline{b01}^{\,00}) \parallel \overline{c11}^{\,00})\,;d01 & \xrightarrow{\surd} \\
(3'') & ((\underline{a00}_{11} \parallel \overline{b01}^{\,11}) \parallel \overline{c11}^{\,11})\,;d01 & \xrightarrow{\{b,c\}} \\
(4'') & ((\underline{a00}_{11} \parallel \underline{b01}_{00}) \parallel \underline{c11}_{00})\,;d01 & \equiv \\
(4a'') & ((a00 \parallel b01) \parallel c11)\,;\overline{d01}^{\,01} & \xrightarrow{\{d\}} \\
(5'') & \underline{((a00 \parallel b01) \parallel c11)\,;d01}_{00} &
\end{array}
$$

Figure 5.2: Two execution sequences corresponding to scenario 1 and 2.

Basic Net: $\Sigma = \mathsf{N}_{\alpha el}$. Then $\mathsf{cl_e}(\Sigma) \stackrel{\text{df}}{=} \{^\circ\Sigma\}$ and $\mathsf{cl_i}(\Sigma) \stackrel{\text{df}}{=} \varnothing$.

Parallel composition: $\Sigma = \Sigma_1 \| \Sigma_2$. Then

$$\mathsf{cl_e}(\Sigma) \stackrel{\text{df}}{=} \bigcup_{k=1}^{2} \{e_\parallel^k \lhd v_\parallel^k \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl_e}(\Sigma_k)\}$$

$$\mathsf{cl_i}(\Sigma) \stackrel{\text{df}}{=} \bigcup_{k=1}^{2} \{v_\parallel^k \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl_i}(\Sigma_k)\}\,.$$

Sequential composition: $\Sigma = \Sigma_1\,;\Sigma_2$. Then

$$\mathsf{cl_e}(\Sigma) \stackrel{\text{df}}{=} \{e_i \lhd v_i^1 \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl_e}(\Sigma_1)\}$$

$$\mathsf{cl_i}(\Sigma) \stackrel{\text{df}}{=} \bigcup_{k=1}^{2} \{v_i^k \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl_i}(\Sigma_k)\} \cup \{i_i \lhd (v_i^1 \blacktriangleleft \Sigma_1^\circ, v_i^2 \blacktriangleleft \mathsf{cl}) \mid \mathsf{cl} \in \mathsf{cl_e}(\Sigma_2)\}\,.$$

Choice operator: $\Sigma = \Sigma_1 \Box \Sigma_2$. Then

$$\begin{aligned}\mathsf{cl}_e(\Sigma) &\stackrel{\text{df}}{=} \{e_\Box \lhd (v_\Box^1 \blacktriangleleft \mathsf{cl}, v_\Box^2 \blacktriangleleft {}^\circ\Sigma_2 \mid \mathsf{cl} \in \mathsf{cl}_e(\Sigma_1)\} \cup \\ &\quad \{e_\Box \lhd (v_\Box^1 \blacktriangleleft {}^\circ\Sigma_1, v_\Box^2 \blacktriangleleft \mathsf{cl}) \mid \mathsf{cl} \in \mathsf{cl}_e(\Sigma_2)\} \\ \mathsf{cl}_i(\Sigma) &\stackrel{\text{df}}{=} \bigcup_{k=1}^{2} \{v_\Box^k \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl}_i(\Sigma_k)\} .\end{aligned}$$

Iteration: $\Sigma = \langle\!\langle \Sigma_1 \circledast \Sigma_2 \circledast \Sigma_3 \rangle\!\rangle$. Then

$$\begin{aligned}\mathsf{cl}_e(\Sigma) &\stackrel{\text{df}}{=} \{e_\circledast \lhd v_\circledast^1 \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl}_e(\Sigma_1)\} \\ \mathsf{cl}_i(\Sigma) &\stackrel{\text{df}}{=} \bigcup_{k=1}^{3} \{v_\circledast^k \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl}_i(\Sigma_k)\} \cup \\ &\quad \{i_\circledast \lhd (v_\circledast^1 \blacktriangleleft \Sigma_1^\circ, v_\circledast^2 \blacktriangleleft \mathsf{cl}, v_\circledast^2 \blacktriangleleft \Sigma_2^\circ, v_\circledast^3 \blacktriangleleft {}^\circ\Sigma_3) \mid \mathsf{cl} \in \mathsf{cl}_e(\Sigma_2)\} \cup \\ &\quad \{i_\circledast \lhd (v_\circledast^1 \blacktriangleleft \Sigma_1^\circ, v_\circledast^2 \blacktriangleleft {}^\circ\Sigma_2, v_\circledast^2 \blacktriangleleft \Sigma_2^\circ, v_\circledast^3 \blacktriangleleft \mathsf{cl}) \mid \mathsf{cl} \in \mathsf{cl}_e(\Sigma_3)\} .\end{aligned}$$

Scoping: $\Sigma = \Sigma_1 \,\mathsf{sc}\, A$. Then

$$\begin{aligned}\mathsf{cl}_e(\Sigma) &\stackrel{\text{df}}{=} \{e_{\mathsf{sc}\,A} \lhd v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl}_e(\Sigma_1)\} \\ \mathsf{cl}_i(\Sigma) &\stackrel{\text{df}}{=} \{v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in \mathsf{cl}_i(\Sigma_1)\} .\end{aligned}$$

Visiting back the running example from figure 4.2 we will present every available cluster in this at-box. At first, we begin with clusters coming from the entry and exit places of at-box Σ, $\mathsf{cl}_1 = \{p_1, p_3\}$ and $\mathsf{cl}_2 = \{p_7\}$. The outmost operator is iteration and we continue with the cluster cl_e of the iteration which comes from the first part of the iteration $\Sigma_1 = (a\,;b) \parallel c$. Since in this part we have parallel composition, the cluster cl_e of Σ_1 will come from both cl_e of the parallel components. The first parallel component is a sequence of action a followed by action b. The cl_e of sequential composition comes from the cl_e of the first component which is basic action a. As a result $\mathsf{cl}_3 = \{p_3\}$. Moreover from the sequential composition, there is also a cl_i cluster, which is $\mathsf{cl}_4 = \{p_4\}$. The second parallel component is basic action c, so $\mathsf{cl}_5 = \{p_1\}$. Coming now to the cl_i clusters of the iteration, to begin with we have the cluster coming from the cl_e cluster of the second component of the iteration (basic action d). This is $\mathsf{cl}_6 = \{p_2, p_5\}$. Furthermore to the cl_i clusters of the iteration, we have the cl_e cluster of the third component of the iteration which is a choice composition between a basic action e and a sequential composition of f and g. In this case, this cluster is the same as cluster cl_6. Finally, we have a cl_i cluster from the sequential composition in the the third part of the iteration, $\mathsf{cl}_7 = \{p_6\}$.

Proposition 5.4. *Let Σ be a composite at-net. If $\mathsf{cl} \in \mathsf{cl}_e(\Sigma)$ then $\mathsf{cl} \subseteq {}^\circ\Sigma$, and if $\mathsf{cl} \in \mathsf{cl}_i(\Sigma)$ then $\mathsf{cl} \subseteq \ddot{\Sigma}$. Moreover, in the latter case, there is a unique interface region $Q \in \mathbb{IR}(\Sigma)$ such that $\mathsf{cl} \subseteq Q$.*

Proof. Follows from the definitions of net refinement and clusters, by a straightforward induction on the syntax of the expression from which Σ has been generated. The uniqueness property follows from proposition 4.3. □

Proposition 5.5. *Let Σ be an initial composite at-net, $\mathsf{cl} \in \mathsf{cl}_{\mathsf{i}}(\Sigma)$ be one of its internal clusters, and $M_1U_1M_2U_2 \ldots M_nU_nM_{n+1}$ be a sequence of markings and steps such that $M_1 = M_\Sigma = {}^\circ\Sigma$ and $M_i[U_i\rangle M_{i+1}$, for $i = 1, \ldots, n$. Moreover, let $M'_i = M_i \cap \mathsf{cl}$, for $i = 1, \ldots, n+1$.*
Then there are indices $k_1 < k_2 < \ldots < k_m$ such that $k_1 = 1$, $k_m = n+2$ and, for each $j < m$, one of the following holds:

$$\textit{Case 1:} \quad \varnothing = M'_{k_j} \subseteq M'_{k_j+1} \subseteq \cdots \subseteq M'_{k_{j+1}-1} .$$

$$\textit{Case 2:} \quad Q = M'_{k_j} \supseteq M'_{k_j+1} \supseteq \cdots \supseteq M'_{k_{j+1}-1} .$$

Moreover the two cases strictly alternate, beginning with Case 1.

Proof. Follows from propositions 4.4 and 5.4. □

5.2.3 Pre-clusters of a transition

For every composite at-net Σ and a transition $t \in T_\Sigma$, the *pre-clusters* of t are defined compositionally below.

Basic Net: $\Sigma = \mathsf{N}_{\alpha el}$. Then, for $t = v_{\alpha el}$, ${}^\diamond t \stackrel{\text{df}}{=} \{{}^\circ\Sigma\}$.

Parallel composition: $\Sigma = \Sigma_1 \| \Sigma_2$. Then, for $t = v^k_\| \lhd u$ $(k = 1, 2)$:

$${}^\diamond t \stackrel{\text{df}}{=} \{e^k_\| \lhd v^k_\| \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_k)\} \cup \{v^k_\| \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_k)\} .$$

Sequential composition: $\Sigma = \Sigma_1 ; \Sigma_2$. Then, for $t = v^1_; \lhd u$:

$${}^\diamond t \stackrel{\text{df}}{=} \{e_; \lhd v^1_; \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)\} \cup \{v^1_; \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)\}$$

and for $t = v^2_; \lhd u$:

$$\begin{aligned} {}^\diamond t \stackrel{\text{df}}{=} & \{i_; \lhd (v^1_; \blacktriangleleft \Sigma_1^\circ, v^2_; \blacktriangleleft \mathsf{cl}) \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_2)\} \cup \\ & \{v^2_; \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_2)\} . \end{aligned}$$

Choice: $\Sigma = \Sigma_1 \square \Sigma_2$. Then, for $t = v^1_\square \lhd u$:

$$\begin{aligned} {}^\diamond t \stackrel{\text{df}}{=} & \{e_\square \lhd (v^1_\square \blacktriangleleft \mathsf{cl}, v^2_\square \blacktriangleleft {}^\circ\Sigma_2) \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)\} \cup \\ & \{v^1_\square \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)\} \end{aligned}$$

and for $t = v^2_\square \lhd u$:

$$\begin{aligned} {}^\diamond t \stackrel{\text{df}}{=} & \{e_\square \lhd (v^1_\square \blacktriangleleft {}^\circ\Sigma_1, v^2_\square \blacktriangleleft \mathsf{cl}) \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_2)\} \cup \\ & \{v^2_\square \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_2)\} . \end{aligned}$$

Iteration: $\Sigma = \langle\!\langle \Sigma_1 \circledast \Sigma_2 \circledast \Sigma_3 \rangle\!\rangle$. Then, for $t = v^1_{\circledast} \lhd u$:

$$^{\diamond}t \stackrel{\text{df}}{=} \{e_{\circledast} \lhd v^1_{\circledast} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)\} \cup \{v^1_{\circledast} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)\}$$

for $t = v^2_{\circledast} \lhd u$:

$$\begin{aligned} ^{\diamond}t \stackrel{\text{df}}{=} & \{i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \Sigma_1^{\circ}, v^2_{\circledast} \blacktriangleleft \mathsf{cl}, v^2_{\circledast} \blacktriangleleft \Sigma_2^{\circ}, v^3_{\circledast} \blacktriangleleft {}^{\circ}\Sigma_3) \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_2)\} \cup \\ & \{v^2_{\circledast} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_2)\} \end{aligned}$$

and for $t = v^3_{\circledast} \lhd u$:

$$\begin{aligned} ^{\diamond}t \stackrel{\text{df}}{=} & \{i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \Sigma_1^{\circ}, v^2_{\circledast} \blacktriangleleft {}^{\circ}\Sigma_2, v^2_{\circledast} \blacktriangleleft \Sigma_2^{\circ}, v^3_{\circledast} \blacktriangleleft \mathsf{cl}) \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_3)\} \cup \\ & \{v^3_{\circledast} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_3)\} . \end{aligned}$$

Scoping: $\Sigma = \Sigma_1 \,\mathsf{sc}\, A$. Then, for $t = v_{\mathsf{sc}\,A} \lhd u$:

$$^{\diamond}t \stackrel{\text{df}}{=} \{e_{\mathsf{sc}\,A} \lhd v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)\} \cup \{v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)\}$$

and for $t = v_{\mathsf{sc}\,A} \lhd \{u, w\}$:

$$\begin{aligned} ^{\diamond}t \stackrel{\text{df}}{=} & \{e_{\mathsf{sc}\,A} \lhd v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in ({}^{\diamond}u \cup {}^{\diamond}w) \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)\} \cup \\ & \{v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl} \mid \mathsf{cl} \in ({}^{\diamond}u \cup {}^{\diamond}w) \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)\} . \end{aligned}$$

Coming back in the example in figure 4.2, we will present the pre-clusters of every transition. Once again due to compositional definition of pre-clusters we are starting from the outmost operator and we follow the rules until we end up with a basic net. We start with transition a. This transition belongs to the first component of the iteration operator. Inside the first component of iteration operator, it belongs to the first component of the parallel composition and finally is the first component of the sequential operation between action a and b. Consequently, the pre-cluster of this transition is ${}^{\diamond}a = \{p_1\}$. Following the same pattern for the rest of the pre-clusters, we have: ${}^{\diamond}b = \{p_4\}$, ${}^{\diamond}c = \{p_1\}$, ${}^{\diamond}d = \{p_2, p_5\}$, ${}^{\diamond}e = \{p_2, p_5\}$, ${}^{\diamond}f = \{p_2, p_5\}$, ${}^{\diamond}g = \{p_6\}$.

Proposition 5.6. *Let Σ be a composite at-net, $t \in T_\Sigma$ and $\mathsf{cl} \in {}^{\diamond}t$. Then $\mathsf{cl} \subseteq {}^{\bullet}t$ and $\lambda_\Sigma(p,t) = \lambda_\Sigma(q,t)$, for all $p, q \in \mathsf{cl}$.*

Proof. The proof proceeds by induction on the structure of the expression from which Σ has been derived. Below we assume that $t \in T_\Sigma$, $\mathsf{cl} \in {}^{\diamond}t$ and $p, q \in \mathsf{cl}$.

Base net: $\Sigma = \mathsf{N}_{\alpha el}$. Then $t = v_{\alpha el}$ and the property clearly holds.

Parallel composition: $\Sigma = \Sigma_1 \| \Sigma_2$.
Case 1: $t = v_{\|}^1 \lhd u$ where $u \in T_{\Sigma_1}$. Then, by the definition of ${}^{\diamond}t$, we have two possibilities:

- $\mathsf{cl} = e_{\|}^1 \lhd v_{\|}^1 \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)$.

  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = e_{\|} \lhd v_{\|}^1 \lhd p'$ and $q = e_{\|} \lhd v_{\|}^1 \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v_{\|}^1 \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)$.

  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v_{\|}^1 \lhd p'$ and $q = v_{\|}^1 \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

Case 2: $t = v_{\|}^2 \lhd u$ where $u \in T_{\Sigma_2}$. Then we proceed similarly as in Case 1.

Sequential composition: $\Sigma = \Sigma_1 ; \Sigma_2$.
Case 1: $t = v_{;}^1 \lhd u$ where $u \in T_{\Sigma_1}$. Then, by the definition of ${}^{\diamond}t$, we have two possibilities:

- $\mathsf{cl} = e_{;} \lhd v_{;}^1 \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = e_{;} \lhd v_{;}^1 \lhd p'$ and $q = e_{;} \lhd v_{;}^1 \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v_{;}^1 \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v_{;}^1 \lhd p'$ and $q = v_{;}^1 \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

Now, $\mathsf{cl} \subseteq {}^\bullet t$ follows from (i) and (ii). Moreover, $\lambda_\Sigma(p,u) = \lambda_\Sigma(q,u)$ follows from (iii) and (iv).

Case 2: $t = v_{\mathsf{i}}^2 \lhd u$ where $u \in T_{\Sigma_2}$. Then, by the definition of ${}^\diamond t$, we have two possibilities:

- $\mathsf{cl} = i_{\mathsf{i}} \lhd (v_{\mathsf{i}}^1 \blacktriangleleft \Sigma_1^\circ, v_{\mathsf{i}}^2 \blacktriangleleft \mathsf{cl}')$, where $\mathsf{cl}' \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_2)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^\bullet t \Leftrightarrow \mathsf{cl}' \subseteq {}^\bullet u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^\bullet u$; and (iii) $\lambda_{\Sigma_2}(p',u) = \lambda_{\Sigma_2}(q',u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = i_{\mathsf{i}} \lhd (v_{\mathsf{i}}^1 \lhd w, v_{\mathsf{i}}^2 \lhd p')$ and $q = i_{\mathsf{i}} \lhd (v_{\mathsf{i}}^1 \lhd w', v_{\mathsf{i}}^2 \lhd q')$ where $w, w' \in \Sigma_1^\circ$ and $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_\Sigma(p,t) = \lambda_{\Sigma_2}(p',u)$ and $\lambda_\Sigma(q,t) = \lambda_{\Sigma_2}(q',u)$.

  Now, $\mathsf{cl} \subseteq {}^\bullet t$ follows from (i) and (ii). Moreover, $\lambda_\Sigma(p,u) = \lambda_\Sigma(q,u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v_{\mathsf{i}}^2 \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_2)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^\bullet t \Leftrightarrow \mathsf{cl}' \subseteq {}^\bullet u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^\bullet u$; and (iii) $\lambda_{\Sigma_2}(p',u) = \lambda_{\Sigma_2}(q',u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v_{\mathsf{i}}^2 \lhd p'$ and $q = v_{\mathsf{i}}^2 \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_\Sigma(p,t) = \lambda_{\Sigma_2}(p',u)$ and $\lambda_\Sigma(q,t) = \lambda_{\Sigma_2}(q',u)$.

  Now, $\mathsf{cl} \subseteq {}^\bullet t$ follows from (i) and (ii). Moreover, $\lambda_\Sigma(p,u) = \lambda_\Sigma(q,u)$ follows from (iii) and (iv).

Choice: $\Sigma = \Sigma_1 \square \Sigma_2$.
Case 1: $t = v_\square^1 \lhd u$ where $u \in T_{\Sigma_1}$. Then, by the definition of ${}^\diamond t$, we have two possibilities:

- $\mathsf{cl} = e_\square \lhd (v_\square^1 \blacktriangleleft \mathsf{cl}', v_\square^2 \blacktriangleleft {}^\circ\Sigma_2)$, where $\mathsf{cl}' \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^\bullet t \Leftrightarrow \mathsf{cl}' \subseteq {}^\bullet u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^\bullet u$; and (iii) $\lambda_{\Sigma_1}(p',u) = \lambda_{\Sigma_1}(q',u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = e_\square \lhd (v_\square^1 \lhd p', v_\square^2 \lhd w)$ and $q = e_\square \lhd (v_\square^1 \lhd q', v_\square^2 \lhd w')$ where $w, w' \in {}^\circ\Sigma_2$ and $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_\Sigma(p,t) = \lambda_{\Sigma_1}(p',u)$ and $\lambda_\Sigma(q,t) = \lambda_{\Sigma_1}(q',u)$.

  Now, $\mathsf{cl} \subseteq {}^\bullet t$ follows from (i) and (ii). Moreover, $\lambda_\Sigma(p,u) = \lambda_\Sigma(q,u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v_\square^1 \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^\diamond u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^\bullet t \Leftrightarrow \mathsf{cl}' \subseteq {}^\bullet u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^\bullet u$; and (iii) $\lambda_{\Sigma_1}(p',u) = \lambda_{\Sigma_1}(q',u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v_\square^1 \lhd p'$ and $q = v_\square^1 \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_\Sigma(p,t) = \lambda_{\Sigma_1}(p',u)$ and $\lambda_\Sigma(q,t) = \lambda_{\Sigma_1}(q',u)$.

  Now, $\mathsf{cl} \subseteq {}^\bullet t$ follows from (i) and (ii). Moreover, $\lambda_\Sigma(p,u) = \lambda_\Sigma(q,u)$ follows from (iii) and (iv).

Case 2: $t = v^2_{\square} \lhd u$ where $u \in T_{\Sigma_2}$. Then we proceed similarly as in Case 1.

Iteration: $\Sigma = \langle\!\langle \Sigma_1 \circledast \Sigma_2 \circledast \Sigma_3 \rangle\!\rangle$.
Case 1: $t = v^1_{\circledast} \lhd u$ where $u \in T_{\Sigma_1}$. Then, by the definition of ${}^{\diamond}t$, we have two possibilities:

- $\mathsf{cl} = e_{\circledast} \lhd v^1_{\circledast} \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = e_{\circledast} \lhd v^1_{\circledast} \lhd p'$ and $q = e_{\circledast} \lhd v^1_{\circledast} \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v^1_{\circledast} \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v^1_{\circledast} \lhd p'$ and $q = v^1_{\circledast} \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

Case 2: $t = v^2_{\circledast} \lhd u$ where $u \in T_{\Sigma_2}$. Then, by the definition of ${}^{\diamond}t$, we have two possibilities:

- $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \Sigma_1^{\circ}, v^2_{\circledast} \blacktriangleleft \mathsf{cl}', v^2_{\circledast} \blacktriangleleft \Sigma_2^{\circ}, v^3_{\circledast} \blacktriangleleft {}^{\circ}\Sigma_3)$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_2)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_2}(p', u) = \lambda_{\Sigma_2}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = i_{\circledast} \lhd (v^1_{\circledast} \lhd w, v^2_{\circledast} \lhd p', v^2_{\circledast} \lhd y, v^3_{\circledast} \lhd z)$ and $q = i_{\circledast} \lhd (v^1_{\circledast} \lhd w', v^2_{\circledast} \lhd q', v^2_{\circledast} \lhd y', v^3_{\circledast} \lhd z')$ where $w, w' \in \Sigma_1^{\circ}$, $y, y' \in \Sigma_2^{\circ}$, $z, z' \in {}^{\circ}\Sigma_3$ and $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_2}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_2}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v^2_{\circledast} \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_2)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_2}(p', u) = \lambda_{\Sigma_2}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v^2_{\circledast} \lhd p'$ and $q = v^2_{\circledast} \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_2}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_2}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

Case 3: $t = v^3_{\circledast} \lhd u$ where $u \in T_{\Sigma_3}$. Then, by the definition of ${}^{\diamond}t$, we have two possibilities:

- $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \Sigma^{\circ}_1, v^2_{\circledast} \blacktriangleleft {}^{\circ}\Sigma_2, v^2_{\circledast} \blacktriangleleft \Sigma^{\circ}_2, v^3_{\circledast} \blacktriangleleft \mathsf{cl}')$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_3)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_3}(p', u) = \lambda_{\Sigma_3}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = i_{\circledast} \lhd (v^1_{\circledast} \lhd w, v^2_{\circledast} \lhd y, v^2_{\circledast} \lhd z, v^3_{\circledast} \lhd p')$ and $q = i_{\circledast} \lhd (v^1_{\circledast} \lhd w', v^2_{\circledast} \lhd y', v^2_{\circledast} \lhd z', v^3_{\circledast} \lhd q')$ where $w, w' \in \Sigma^{\circ}_1$, $y, y' \in {}^{\circ}\Sigma_2$, $z, z' \in \Sigma^{\circ}_2$ and $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_3}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_3}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v^3_{\circledast} \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_3)$
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_3}(p', u) = \lambda_{\Sigma_3}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v^3_{\circledast} \lhd p'$ and $q = v^3_{\circledast} \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_3}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_3}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

Scoping: $\Sigma = \Sigma_1 \,\mathsf{sc}\, A$.
Case 1: $t = v_{\mathsf{sc}\,A} \lhd u$ where $u \in T_{\Sigma_1}$. Then, by the definition of ${}^{\diamond}t$, we have two possibilities:

- $\mathsf{cl} = e_{\mathsf{sc}\,A} \lhd v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{e}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = e_{\mathsf{sc}\,A} \lhd v_{\mathsf{sc}\,A} \lhd p'$ and $q = e_{\mathsf{sc}\,A} \lhd v_{\mathsf{sc}\,A} \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

- $\mathsf{cl} = v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}'$, where $\mathsf{cl}' \in {}^{\diamond}u \cap \mathsf{cl}_{\mathsf{i}}(\Sigma_1)$.
  By the definition of net refinement, we have: (i) $\mathsf{cl} \subseteq {}^{\bullet}t \Leftrightarrow \mathsf{cl}' \subseteq {}^{\bullet}u$. And, by the induction hypothesis: (ii) $\mathsf{cl}' \subseteq {}^{\bullet}u$; and (iii) $\lambda_{\Sigma_1}(p', u) = \lambda_{\Sigma_1}(q', u)$, for all $p', q' \in \mathsf{cl}'$. Moreover, we have that $p = v_{\mathsf{sc}\,A} \lhd p'$ and $q = v_{\mathsf{sc}\,A} \lhd q'$ where $p', q' \in \mathsf{cl}'$, and by the definition of net refinement: (iv) $\lambda_{\Sigma}(p, t) = \lambda_{\Sigma_1}(p', u)$ and $\lambda_{\Sigma}(q, t) = \lambda_{\Sigma_1}(q', u)$.

  Now, $\mathsf{cl} \subseteq {}^{\bullet}t$ follows from (i) and (ii). Moreover, $\lambda_{\Sigma}(p, u) = \lambda_{\Sigma}(q, u)$ follows from (iii) and (iv).

Case 2: $t = v_{\mathsf{sc}\,A} \lhd \{u, w\}$. Then we proceed similarly as in Case 1. □

Proposition 5.7. *Let Σ be a composite at-net, $t \in T_\Sigma$ and $p \in {}^\bullet t$. Then there is $\mathsf{cl} \in {}^\diamond t$ such that $p \in \mathsf{cl}$.*

Proof. Follows by induction on the structure of the expression from which Σ has been derived, similarly as proposition 5.6. □

5.2.4 Intuition behind the cluster-based approach

We are now revisiting the example presented in the previous section in figure 5.1 having in mind the new cluster-based approach. By the definition of clusters, there are six clusters in this at-box: $\mathsf{cl}_1 \stackrel{\text{df}}{=} \{p_1, p_2, p_3\}$ $\mathsf{cl}_2 \stackrel{\text{df}}{=} \{p_1\}$, $\mathsf{cl}_3 \stackrel{\text{df}}{=} \{p_2\}$, $\mathsf{cl}_4 \stackrel{\text{df}}{=} \{p_3\}$, $\mathsf{cl}_5 \stackrel{\text{df}}{=} \{p_4, p_5, p_6\}$ and $\mathsf{cl}_6 \stackrel{\text{df}}{=} \{p_7\}$. Assuming this ordering of clusters, our two scenarios can be re-written as follows:

	scenario1		*scenario2*	
$(1''')$	$(00, 00, 00, 00, \bot, \bot)$	$[\{t_1, t_2\}\rangle$	$(00, 00, 00, 00, \bot, \bot)$	$[\{t_1\}\rangle$
$(2''')$	$(00, \bot, \bot, 00, 00, \bot)$	$[\surd\rangle$	$(00, \bot, 00, 00, 00, \bot)$	$[\surd\rangle$
$(3''')$	$(11, \bot, \bot, 11, 11, \bot)$	$[\{t_3\}\rangle$	$(11, \bot, 11, 11, 11, \bot)$	$[\{t_2, t_3\}\rangle$
$(4''')$	$(\bot, \bot, \bot, \bot, 01, \bot)$	$[\{t_4\}\rangle$	$(\bot, \bot, \bot, \bot, 01, \bot)$	$[\{t_4\}\rangle$
$(5''')$	$(\bot, \bot, \bot, \bot, \bot, 00)$		$(\bot, \bot, \bot, \bot, \bot, 00)$	

Note that the problem encountered before with line (4) in the execution scenarios is no longer present in line $(4''')$. Effectively, this means that we can suitably adopt the proof technique used in, e.g., [9], to justify the main results of this thesis.

We will now start the introduction of the auxiliary algebra of arc-timed boxes which will serve as a bridge between at-boxes and at-expressions.

5.3 Cluster-based timed-arc boxes

A *cluster at-box* (or cat-box) is a pair $\mathfrak{A} \stackrel{\text{df}}{=} (\Sigma, \mathcal{M})$ such that $\Sigma = \mathsf{box}(J)$, for some static or dynamic at-expression given by the syntax (3.1,3.2) and

$$\mathcal{M} : CL_\Sigma \to \mathbb{D}$$

is a *cluster filling* (state) such that the following *consistency conditions* hold:

- For every cl in CL_Σ, $\mathcal{M}(\mathsf{cl}) = \bot$ if and only if $M_\Sigma(p) = \{0\}, \forall p \in \mathsf{cl}$.
- For all cl and cl' in $\{^\circ\Sigma\} \cup \mathsf{cl}_\mathsf{e}$, if $\mathcal{M}(\mathsf{cl}) \neq \bot \neq \mathcal{M}(\mathsf{cl}')$ then $\mathcal{M}(\mathsf{cl}) = \mathcal{M}(\mathsf{cl}')$.

We say that $\mathfrak{A}$ is static/dynamic if so is J and denote $\mathfrak{A} \in \mathfrak{T}_J$. We then introduce some useful notations:

- $\lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor \stackrel{\text{df}}{=} \Sigma$ and $\lfloor \mathfrak{A} \rfloor \stackrel{\text{df}}{=} (\lfloor \Sigma \rfloor, \mathcal{N})$, where $\mathcal{N}$ always returns $\bot$.

- For every transition $t \in T_\Sigma$ and cluster $\mathsf{cl} \in {}^\diamond t$,

$$\lambda(\mathsf{cl}, t) \stackrel{\text{df}}{=} \lambda_\Sigma(p, t) ,$$

for any $p \in \mathsf{cl}$.

- The state $\mathcal{M}^\surd$ is defined so that, for every cluster cl in Σ,

$$\mathcal{M}^\surd(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} (\mathbb{E}+1)(\mathbb{L}+1) & \text{if } \mathcal{M}(\mathsf{cl}) = \mathbb{E}\mathbb{L} \\ \bot & \text{otherwise} \end{cases}$$

and the cat-box $\mathfrak{A}^\surd$ is then defined as $(\Sigma, \mathcal{M}^\surd)$.

The above notions are well-defined. This is immediate in all but one case, namely $\lambda(\mathsf{cl}, t)$ is well-defined by proposition 5.6.

Proposition 5.8. *Let $\mathfrak{A}$ be a cat-box in $\mathfrak{T}_J$.*

1. *$\lfloor \mathfrak{A} \rfloor$ is a static cat-box in $\mathfrak{T}_{\lfloor J \rfloor}$.*
2. *If $\mathfrak{A}$ is static, then $\lfloor \mathfrak{A} \rfloor = \mathfrak{A}$.*

Proof. Follows from the properties of the standard box algebra. □

A set of transitions $U \subseteq T_\Sigma$ is *enabled* by $\mathfrak{A}$ if it is enabled by Σ and, for every transition $t \in U$ and every cluster $\mathsf{cl} \in {}^\diamond t$, we have that $\mathcal{M}(\mathsf{cl})$ tsat $\lambda(\mathsf{cl}, t)$. We denote this by $U \in \mathsf{enabled}(\mathfrak{A})$. This enabling is *urgent*, denoted $U \in \mathsf{urgent}(\mathfrak{A})$, if U is not enabled by $\mathfrak{A}^\surd$.

An enabled step may be *executed* and yield a *follower* cat-box $\mathfrak{X} = (\Sigma', \mathcal{N})$ such that $\Sigma[U\rangle\Sigma'$ and, for every cluster cl in Σ,

$$\mathcal{N}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \bot & \text{if } M_{\Sigma'} \cap \mathsf{cl} = \varnothing \\ 0\mathbb{L} & \text{if } \mathsf{cl} \cap U^\bullet \neq 0 \text{ and } \mathcal{M}(\mathsf{cl}) = \mathbb{E}\mathbb{L} \\ 00 & \text{if } \mathsf{cl} \cap U^\bullet \neq 0 \text{ and } \mathcal{M}(\mathsf{cl}) = \bot \\ \mathcal{M}(\mathsf{cl}) & \text{otherwise .} \end{cases}$$

We denote this by $\mathfrak{A}[U\rangle\mathfrak{X}$.

A time move is enabled if there is no urgent enabled step; it then can be executed and yield a follower cat-box: $\mathfrak{A}[\surd\rangle\mathfrak{A}^\surd$.

Proposition 5.9. *Let $\mathfrak{A}$ be a cat-box and $\mathfrak{A}[U\rangle\mathfrak{X}$ or $\mathfrak{A}[\surd\rangle\mathfrak{X}$.*

1. *If $\mathfrak{A}$ is static, then $U = \varnothing$ and $\mathfrak{A} = \mathfrak{X}$.*
2. *If $\mathfrak{A}$ is dynamic then so is $\mathfrak{X}$.*

Proof. Follows from the properties of the standard box algebra and, additionally, we need to check that the two consistency conditions from the definition of at-boxes are satisfied. The latter is straightforward (ex-directedness of at-nets is again important here). □

Proposition 5.10. *Let $\mathfrak{A}[U\rangle\mathfrak{X}$, where $U = \{t_1, \ldots, t_k\}$. Then there are cat-boxes $\mathfrak{A}_0, \ldots, \mathfrak{A}_k$ such that $\mathfrak{A}_0 = \mathfrak{A}$, $\mathfrak{A}_k = \mathfrak{X}$ and $\mathfrak{A}_{i-1}[t_i\rangle\mathfrak{A}_i$, for $i = 1, \ldots, k$.*

Proof. Follows from the standard properties of safe Petri nets and proposition 5.5 which ensures that for each t_i no time token involved in the enabling of t_i is involved in the firing of the preceding transitions $t_1, \ldots, t_{i-1}$. □

5.3.1 Representing global behaviour of cat-boxes

As for at-boxes, we have four different ways of capturing the overall behaviour of cat-boxes, namely $\mathsf{RT}_{\mathfrak{A}}$, $\mathsf{fRT}_{\mathfrak{A}}$ $\mathsf{TS}_{\mathfrak{A}}$ and $\mathsf{fTS}_{\mathfrak{A}}$. Their definitions are a straightforward adaptation of those for at-boxes.

5.3.2 An algebra of cat-boxes

We define an algebra of cat-boxes following the syntax (3.1,3.2). To start with, the basic at-box $\mathsf{N}^{\mathsf{at}}_{\alpha el} = (\mathsf{N}_{\alpha el}, \mathcal{M})$, where for every cluster $\mathsf{cl} \in CL_{\mathsf{N}_{\alpha el}}$ we have:

$$\mathcal{M}(\mathsf{cl}) \stackrel{\text{df}}{=} \bot \tag{5.1}$$

is a basic building block of the algebra. In what now follows, we assume that $\mathfrak{A} = (\Sigma, \mathcal{M}) \in \mathfrak{T}_H$, $\mathfrak{X} = (\Psi, \mathcal{N}) \in \mathfrak{T}_J$ and $\mathfrak{V} = (\Phi, \mathcal{P}) \in \mathfrak{T}_K$ are cat-boxes.

Overbarring and underbarring: If H is a static at-expression and $\mathbb{EL} \in \mathbb{D}$, then $\overline{\mathfrak{A}}^{\mathbb{EL}} = (\overline{\Sigma}, \mathcal{R}) \in \mathfrak{T}_{\overline{H}^{\mathbb{EL}}}$ where, for every cluster $\mathsf{cl} \in CL_{\Sigma}$, we have:

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathbb{EL} & \text{if } \mathsf{cl} \in \mathsf{cl}_{\mathsf{e}}(\Sigma) \text{ or } \mathsf{cl} = {}^{\circ}\Sigma \\ \bot & \text{otherwise .} \end{cases} \tag{5.2}$$

Similarly, $\underline{\mathfrak{A}}_{\mathbb{EL}} = (\underline{\Sigma}, \mathcal{N}) \in \mathfrak{T}_{\underline{H}_{\mathbb{EL}}}$ where, for every cluster $\mathsf{cl} \in CL_{\Sigma}$, we have:

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathbb{EL} & \text{if } \mathsf{cl} = \Sigma^{\circ} \\ \bot & \text{otherwise .} \end{cases} \tag{5.3}$$

Choice: $\mathfrak{A}\Box\mathfrak{X}$ is defined if $H\Box J$ is generated by the syntax (3.1,3.2), and then $\mathfrak{A}\Box\mathfrak{X} \stackrel{\text{df}}{=} (\Sigma\Box\Psi, \mathcal{R}) \in \mathfrak{T}_{H\Box J}$ where, for every cluster $\mathsf{cl} \in CL_{\Sigma\Box\Psi}$, we have:

- when H is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathcal{M}({}^{\circ}\Sigma) & \text{if } \mathsf{cl} = {}^{\circ}(\Sigma\Box\Psi) \\ \mathcal{M}(\Sigma^{\circ}) & \text{if } \mathsf{cl} = (\Sigma\Box\Psi)^{\circ} \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = e_{\Box} \lhd (v^1_{\Box} \blacktriangleleft \mathsf{cl}', v^2_{\Box} \blacktriangleleft {}^{\circ}\Psi) \\ \mathcal{M}({}^{\circ}\Sigma) & \text{if } \mathsf{cl} = e_{\Box} \lhd (v^1_{\Box} \blacktriangleleft {}^{\circ}\Sigma, v^2_{\Box} \blacktriangleleft \mathsf{cl}') \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^1_{\Box} \blacktriangleleft \mathsf{cl}' \\ \bot & \text{if } \mathsf{cl} = v^2_{\Box} \blacktriangleleft \mathsf{cl}' \end{cases} \tag{5.4}$$

- when J is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathcal{N}(^\circ\Psi) & \text{if } \mathsf{cl} = {}^\circ(\Sigma \square \Psi) \\ \mathcal{N}(\Psi^\circ) & \text{if } \mathsf{cl} = (\Sigma \square \Psi)^\circ \\ \mathcal{N}(^\circ\Psi) & \text{if } \mathsf{cl} = e_\square \lhd (v^1_\square \blacktriangleleft \mathsf{cl}', v^2_\square \blacktriangleleft {}^\circ\Psi) \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = e_\square \lhd (v^1_\square \blacktriangleleft {}^\circ\Sigma, v^2_\square \blacktriangleleft \mathsf{cl}') \\ \bot & \text{if } \mathsf{cl} = v^1_\square \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^2_\square \blacktriangleleft \mathsf{cl}' \end{cases} \tag{5.5}$$

- when both H and J are static at-expressions,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \bot\,. \tag{5.6}$$

Sequence: $\mathfrak{A}\,;\mathfrak{X}$ is defined if $H\,;J$ is generated by the syntax (3.1,3.2), and then $\mathfrak{A}\,;\mathfrak{X} \stackrel{\text{df}}{=} (\Sigma\,;\Psi, \mathcal{R}) \in \mathfrak{T}_{H\,;J}$ where, for every cluster $\mathsf{cl} \in CL_{\Sigma\,;\Psi}$, we have:

- when H is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathcal{M}(^\circ\Sigma) & \text{if } \mathsf{cl} = {}^\circ(\Sigma\,;\Psi) \\ \bot & \text{if } \mathsf{cl} = (\Sigma\,;\Psi)^\circ \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = e_; \lhd (v^1_; \blacktriangleleft \mathsf{cl}') \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^1_; \blacktriangleleft \mathsf{cl}' \\ \bot & \text{if } \mathsf{cl} = v^2_; \blacktriangleleft \mathsf{cl}' \\ \mathcal{M}(\Sigma^\circ) & \text{if } \mathsf{cl} = i_; \lhd (v^1_; \blacktriangleleft \Sigma^\circ, v^2_; \blacktriangleleft \mathsf{cl}') \end{cases} \tag{5.7}$$

- when J is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \bot & \text{if } \mathsf{cl} = {}^\circ(\Sigma\,;\Psi) \\ \mathcal{N}(\Psi^\circ) & \text{if } \mathsf{cl} = (\Sigma\,;\Psi)^\circ \\ \bot & \text{if } \mathsf{cl} = e_; \lhd (v^1_; \blacktriangleleft \mathsf{cl}') \\ \bot & \text{if } \mathsf{cl} = v^1_; \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^2_; \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = i_; \lhd (v^1_; \blacktriangleleft \Sigma^\circ, v^2_; \blacktriangleleft \mathsf{cl}') \end{cases} \tag{5.8}$$

- when both H and J are static at-expressions,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \bot\,. \tag{5.9}$$

Parallel Composition: $\mathfrak{A}\|\mathfrak{X}$ is defined if $H\|J$ is generated by the syntax (3.1,3.2), and then $\mathfrak{A}\|\mathfrak{X} \stackrel{\text{df}}{=} (\Sigma\|\Psi, \mathcal{R}) \in \mathfrak{T}_{H\|J}$ where, for every cluster $\mathsf{cl} \in CL_{\Sigma\|\Psi}$,

we have:

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathcal{M}(^\circ\Sigma) \oplus \mathcal{N}(^\circ\Psi) & \text{if } \mathsf{cl} = {}^\circ(\Sigma \| \Psi) \\ \mathcal{M}(\Sigma^\circ) \oplus \mathcal{N}(\Psi^\circ) & \text{if } \mathsf{cl} = (\Sigma \| \Psi)^\circ \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = e^1_\| \lhd v^1_\| \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = e^2_\| \lhd v^2_\| \blacktriangleleft \mathsf{cl}' \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^1_\| \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^2_\| \blacktriangleleft \mathsf{cl}' \,. \end{cases} \tag{5.10}$$

Note that when both H and J are static at-expressions, then

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \bot \,. \tag{5.11}$$

for every cluster $\mathsf{cl} \in CL_{\Sigma\|\Psi}$.

Iteration: $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{Y} \rangle\!\rangle$ is defined if $\langle\!\langle H \circledast J \circledast K \rangle\!\rangle$ is generated by the syntax (3.1,3.2), and then

$$\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{Y} \rangle\!\rangle \stackrel{\text{df}}{=} (\langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle, \mathcal{R}) \in \mathfrak{T}_{\langle\!\langle H \circledast J \circledast K \rangle\!\rangle}$$

where, for every cluster $\mathsf{cl} \in CL_{\langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle}$, we have:

- when H is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathcal{M}(^\circ\Sigma) & \text{if } \mathsf{cl} = {}^\circ\langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle \\ \bot & \text{if } \mathsf{cl} = \langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle^\circ \\ \mathcal{M}(^\circ\Sigma) & \text{if } \mathsf{cl} = e_\circledast \lhd (v^1_\circledast \blacktriangleleft \mathsf{cl}') \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^1_\circledast \blacktriangleleft \mathsf{cl}' \\ \bot & \text{if } \mathsf{cl} = v^2_\circledast \blacktriangleleft \mathsf{cl}' \\ \bot & \text{if } \mathsf{cl} = v^3_\circledast \blacktriangleleft \mathsf{cl}' \\ \mathcal{M}(\Sigma^\circ) & \text{if } \mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft \Sigma^\circ, v^2_\circledast \blacktriangleleft \mathsf{cl}', \\ & \qquad v^2_\circledast \blacktriangleleft \Psi^\circ, v^3_\circledast \blacktriangleleft {}^\circ\Phi) \\ \mathcal{M}(\Sigma^\circ) & \text{if } \mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft \Sigma^\circ, v^2_\circledast \blacktriangleleft {}^\circ\Psi, \\ & \qquad v^2_\circledast \blacktriangleleft \Psi^\circ, v^3_\circledast \blacktriangleleft \mathsf{cl}') \end{cases} \tag{5.12}$$

- when J is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \bot & \text{if } \mathsf{cl} = {}^\circ\langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle \\ \bot & \text{if } \mathsf{cl} = \langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle)^\circ \\ \bot & \text{if } \mathsf{cl} = e_\circledast \lhd (v^1_\circledast \blacktriangleleft \mathsf{cl}') \\ \bot & \text{if } \mathsf{cl} = v^1_\circledast \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^2_\circledast \blacktriangleleft \mathsf{cl}' \\ \bot & \text{if } \mathsf{cl} = v^3_\circledast \blacktriangleleft \mathsf{cl}' \\ \mathcal{N}(\mathsf{cl}') \oplus \mathcal{N}(\Psi^\circ) & \text{if } \mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft \Sigma^\circ, v^2_\circledast \blacktriangleleft \mathsf{cl}', \\ & \qquad v^2_\circledast \blacktriangleleft \Psi^\circ, v^3_\circledast \blacktriangleleft {}^\circ\Phi) \\ \mathcal{N}(^\circ\Psi) \oplus \mathcal{N}(\Psi^\circ) & \text{if } \mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft \Sigma^\circ, v^2_\circledast \blacktriangleleft {}^\circ\Psi, \\ & \qquad v^2_\circledast \blacktriangleleft \Psi^\circ, v^3_\circledast \blacktriangleleft \mathsf{cl}') \end{cases} \tag{5.13}$$

- when K is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \bot & \text{if } \mathsf{cl} = {}^\circ\langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle \\ \mathcal{P}(\Phi^\circ) & \text{if } \mathsf{cl} = \langle\!\langle \Sigma \circledast \Psi \circledast \Phi \rangle\!\rangle^\circ \\ \bot & \text{if } \mathsf{cl} = e_\circledast \lhd (v^1_\circledast \blacktriangleleft \mathsf{cl}') \\ \bot & \text{if } \mathsf{cl} = v^1_\circledast \blacktriangleleft \mathsf{cl}' \\ \bot & \text{if } \mathsf{cl} = v^2_\circledast \blacktriangleleft \mathsf{cl}' \\ \mathcal{P}(\mathsf{cl}') & \text{if } \mathsf{cl} = v^3_\circledast \blacktriangleleft \mathsf{cl}' \\ \mathcal{P}(^\circ\Phi) & \text{if } \mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft \Sigma^\circ, v^2_\circledast \blacktriangleleft \mathsf{cl}', \\ & \qquad v^2_\circledast \blacktriangleleft \Psi^\circ, v^3_\circledast \blacktriangleleft {}^\circ\Phi) \\ \mathcal{P}(\mathsf{cl}') & \text{if } \mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft \Sigma^\circ, v^2_\circledast \blacktriangleleft {}^\circ\Psi, \\ & \qquad v^2_\circledast \blacktriangleleft \Psi^\circ, v^3_\circledast \blacktriangleleft \mathsf{cl}') \end{cases} \tag{5.14}$$

- when H, J and K are static at-expressions,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \bot\,. \tag{5.15}$$

Scoping: $\mathfrak{A}\,\mathsf{sc}\,A$ is defined if $H\,\mathsf{sc}\,A$ is generated by the syntax (3.1,3.2), and then $\mathfrak{A}\,\mathsf{sc}\,A \stackrel{\text{df}}{=} (\Sigma\,\mathsf{sc}\,A, \mathcal{R}) \in \mathfrak{T}_{H\,\mathsf{sc}\,A}$ where, for every cluster $\mathsf{cl} \in CL_{\Sigma\,\mathsf{sc}\,A}$, we have:

- when H is a dynamic at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \mathcal{M}(^\circ\Sigma) & \text{if } \mathsf{cl} = {}^\circ(\Sigma\,\mathsf{sc}\,A) \\ \mathcal{M}(\Sigma^\circ) & \text{if } \mathsf{cl} = (\Sigma\,\mathsf{sc}\,A)^\circ \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = e_{\mathsf{sc}\,A} \lhd (v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}') \\ \mathcal{M}(\mathsf{cl}') & \text{if } \mathsf{cl} = v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}' \end{cases} \tag{5.16}$$

- when H is a static at-expression,

$$\mathcal{R}(\mathsf{cl}) \stackrel{\text{df}}{=} \bot\,. \tag{5.17}$$

Note that for each of the above operations, one can easily check that the result is indeed a valid cat-box corresponding to the at-expression given in the definition.

5.3.3 Static properties of cat-boxes

An important result from the point of view of developing a correspondence between cat-boxes and at-expressions is given next (see also table 3.1).

Proposition 5.11. *Let $\mathfrak{A}$, $\mathfrak{X}$ and $\mathfrak{V}$ be static cat-boxes and $\mathbb{EL}, \mathbb{E'L'} \in \mathbb{D}$. Then the following hold.*

1. *For choice composition:*

$$\overline{\mathfrak{A}\Box\mathfrak{X}}^{\mathbf{EL}} = \overline{\mathfrak{A}}^{\mathbf{EL}}\Box\mathfrak{X} = \mathfrak{A}\Box\overline{\mathfrak{X}}^{\mathbf{EL}}$$

$$\underline{\mathfrak{A}\Box\mathfrak{X}}_{\mathbf{EL}} = \underline{\mathfrak{A}}_{\mathbf{EL}}\Box\mathfrak{X} = \mathfrak{A}\Box\underline{\mathfrak{X}}_{\mathbf{EL}} .$$

2. *For iteration:*

$$\overline{\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}^{\mathbf{EL}} = \langle\!\langle \overline{\mathfrak{A}}^{\mathbf{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle$$

$$\underline{\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}_{\mathbf{EL}} = \langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \underline{\mathfrak{V}}_{\mathbf{EL}} \rangle\!\rangle$$

$$\langle\!\langle \underline{\mathfrak{A}}_{\mathbf{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A} \circledast \overline{\mathfrak{X}}^{\mathbf{EL}} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A} \circledast \underline{\mathfrak{X}}_{\mathbf{EL}} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \overline{\mathfrak{V}}^{\mathbf{EL}} \rangle\!\rangle .$$

3. *For sequence composition:*

$$\overline{\mathfrak{A};\mathfrak{X}}^{\mathbf{EL}} = \overline{\mathfrak{A}}^{\mathbf{EL}};\mathfrak{X}$$

$$\underline{\mathfrak{A}}_{\mathbf{EL}};\mathfrak{X} = \mathfrak{A};\overline{\mathfrak{X}}^{\mathbf{EL}}$$

$$\mathfrak{A};\underline{\mathfrak{X}}_{\mathbf{EL}} = \underline{\mathfrak{A};\mathfrak{X}}_{\mathbf{EL}} .$$

4. *For parallel composition:*

$$\overline{\mathfrak{A}\|\mathfrak{X}}^{\mathbf{EL}} = \overline{\mathfrak{A}}^{\mathbf{EL}}\|\overline{\mathfrak{X}}^{\mathbf{EL}}$$

$$\underline{\mathfrak{A}}_{\mathbf{EL}}\|\underline{\mathfrak{X}}_{\mathbf{E'L'}} = \underline{\mathfrak{A}\|\mathfrak{X}}_{\min\{\mathbf{E},\mathbf{E'}\}\max\{\mathbf{L},\mathbf{L'}\}} .$$

5. *For scoping:*

$$\overline{\mathfrak{A}\,\mathsf{sc}\,A}^{\mathbf{EL}} = \overline{\mathfrak{A}}^{\mathbf{EL}}\,\mathsf{sc}\,A$$

$$\underline{\mathfrak{A}\,\mathsf{sc}\,A}_{\mathbf{EL}} = \underline{\mathfrak{A}}_{\mathbf{EL}}\,\mathsf{sc}\,A .$$

Proof. It follows from the standard box algebra results that the underlying at-nets are in each case equal. Therefore, all we need to do is check whether the cluster filling mapping are also identical.

Case 1: $\overline{\mathfrak{A}\Box\mathfrak{X}}^{\mathbf{EL}} = \overline{\mathfrak{A}}^{\mathbf{EL}}\Box\mathfrak{X} = \mathfrak{A}\Box\overline{\mathfrak{X}}^{\mathbf{EL}}$. After denoting $\overline{\mathfrak{A}\Box\mathfrak{X}}^{\mathbf{EL}} = A$, $\overline{\mathfrak{A}}^{\mathbf{EL}}\Box\mathfrak{X} = B$ and $\mathfrak{A}\Box\overline{\mathfrak{X}}^{\mathbf{EL}} = C$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^{\circ}\llbracket\mathfrak{A}\Box\mathfrak{X}\rrbracket$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbf{EL}}}({}^{\circ}\llbracket\mathfrak{A}\rrbracket) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbf{EL}}}({}^{\circ}\llbracket\mathfrak{X}\rrbracket) \overset{(5.2)}{=} \mathbb{EL}.$

- For $\mathsf{cl} = [\![\mathfrak{A} \Box \mathfrak{X}]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ) \overset{(5.2)}{=} \bot$.

- For $\mathsf{cl} = e_\Box \lhd (v_\Box^1 \blacktriangleleft \mathsf{cl}', v_\Box^2 \blacktriangleleft {}^\circ[\![\mathfrak{X}]\!])$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}({}^\circ[\![\mathfrak{X}]\!]) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = e_\Box \lhd (v_\Box^1 \blacktriangleleft {}^\circ[\![\mathfrak{A}]\!], v_\Box^2 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = v_\Box^1 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \bot$.

- For $\mathsf{cl} = v_\Box^2 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.

Case 2: $\underline{\mathfrak{A} \Box \mathfrak{X}}_{\mathbb{EL}} = \underline{\mathfrak{A}}_{\mathbb{EL}} \Box \mathfrak{X} = \mathfrak{A} \Box \underline{\mathfrak{X}}_{\mathbb{EL}}$. After denoting $\underline{\mathfrak{A} \Box \mathfrak{X}}_{\mathbb{EL}} = A$, $\underline{\mathfrak{A}}_{\mathbb{EL}} \Box \mathfrak{X} = B$ and $\mathfrak{A} \Box \underline{\mathfrak{X}}_{\mathbb{EL}} = C$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\mathfrak{A} \Box \mathfrak{X}]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\mathfrak{A}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\mathfrak{X}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{X}]\!]) \overset{(5.3)}{=} \bot$.

- For $\mathsf{cl} = [\![\mathfrak{A} \square \mathfrak{X}]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = e_\square \lhd (v_\square^1 \blacktriangleleft \mathsf{cl}', v_\square^2 \blacktriangleleft {}^\circ[\![\mathfrak{X}]\!])$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{X}]\!]) \overset{(5.3)}{=} \bot$.

- For $\mathsf{cl} = e_\square \lhd (v_\square^1 \blacktriangleleft {}^\circ[\![\mathfrak{A}]\!], v_\square^2 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$.

- For $\mathsf{cl} = v_\square^1 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \bot$.

- For $\mathsf{cl} = v_\square^2 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.4)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.5)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$.

Case 3: $\overline{\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}^{\mathbb{EL}} = \langle\!\langle \overline{\mathfrak{A}}^{\mathbb{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle$. After denoting

$$\overline{\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}^{\mathbb{EL}} = A \quad \text{and} \quad \langle\!\langle \overline{\mathfrak{A}}^{\mathbb{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = B\,,$$

we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = [\![\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \bot$.

- For $\mathsf{cl} = e^1_\circledast \lhd (v^1_\circledast \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(^\circ[\![\mathfrak{A}]\!]) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = v^1_\circledast \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.

- For $\mathsf{cl} = v^2_\circledast \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \bot$.

- For $\mathsf{cl} = v^3_\circledast \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \bot$.

- For $\mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v^2_\circledast \blacktriangleleft \mathsf{cl}', v^2_\circledast \blacktriangleleft [\![\mathfrak{X}]\!]^\circ, v^3_\circledast \blacktriangleleft {}^\circ[\![\mathfrak{V}]\!])$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.2)}{=} \bot$.

- For $\mathsf{cl} = i_\circledast \lhd (v^1_\circledast \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v^2_\circledast \blacktriangleleft {}^\circ[\![\mathfrak{X}]\!], v^2_\circledast \blacktriangleleft [\![\mathfrak{X}]\!]^\circ, v^3_\circledast \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.2)}{=} \bot$.

Case 4: $\underline{\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}_{\mathbb{EL}} = \langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \underline{\mathfrak{V}}_{\mathbb{EL}} \rangle\!\rangle$. After denoting $\underline{\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}_{\mathbb{EL}} = A$ and $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \underline{\mathfrak{V}}_{\mathbb{EL}} \rangle\!\rangle = B$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \bot$.

- For $\mathsf{cl} = [\![\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\underline{\mathfrak{V}}_{\mathbb{EL}}}([\![\mathfrak{V}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$.
- For $\mathsf{cl} = e^1_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \bot$.
- For $\mathsf{cl} = v^1_{\circledast} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \bot$.
- For $\mathsf{cl} = v^2_{\circledast} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \bot$.
- For $\mathsf{cl} = v^3_{\circledast} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\underline{\mathfrak{V}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v^2_{\circledast} \blacktriangleleft \mathsf{cl}', v^2_{\circledast} \blacktriangleleft [\![\mathfrak{X}]\!]^\circ, v^3_{\circledast} \blacktriangleleft {}^\circ[\![\mathfrak{V}]\!])$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\underline{\mathfrak{V}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{V}]\!]) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v^2_{\circledast} \blacktriangleleft {}^\circ[\![\mathfrak{X}]\!], v^2_{\circledast} \blacktriangleleft [\![\mathfrak{X}]\!]^\circ, v^3_{\circledast} \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\underline{\mathfrak{V}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$.

Case 5: $\langle\!\langle \underline{\mathfrak{A}}_{\mathbb{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A} \circledast \overline{\mathfrak{X}}^{\mathbb{EL}} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A} \circledast \underline{\mathfrak{X}}_{\mathbb{EL}} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \overline{\mathfrak{V}}^{\mathbb{EL}} \rangle\!\rangle$.
After denoting $\langle\!\langle \underline{\mathfrak{A}}_{\mathbb{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = A$, $\langle\!\langle \mathfrak{A} \circledast \overline{\mathfrak{X}}^{\mathbb{EL}} \circledast \mathfrak{V} \rangle\!\rangle = B$, $\langle\!\langle \mathfrak{A} \circledast \underline{\mathfrak{X}}_{\mathbb{EL}} \circledast \mathfrak{V} \rangle\!\rangle = C$ and $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \overline{\mathfrak{V}}^{\mathbb{EL}} \rangle\!\rangle = D$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.3)}{=} \bot$
  and

$\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \bot$
and
$\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \bot$
and
$\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \bot$.

- For $\mathsf{cl} = [\![\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \bot$
  and
  $\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\overline{\mathfrak{V}}^{\mathbb{EL}}}([\![\mathfrak{V}]\!]^\circ) \overset{(5.2)}{=} \bot$.
- For $\mathsf{cl} = e^1_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \bot$
  and
  $\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \bot$.
- For $\mathsf{cl} = v^1_{\circledast} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \bot$
  and
  $\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \bot$.
- For $\mathsf{cl} = v^2_{\circledast} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \bot$.
- For $\mathsf{cl} = v^3_{\circledast} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \bot$
  and

$\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \bot$
and
$\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \bot$
and
$\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\overline{\mathfrak{V}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.

- For $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v^2_{\circledast} \blacktriangleleft \mathsf{cl}', v^2_{\circledast} \blacktriangleleft [\![\mathfrak{X}]\!]^\circ, v^3_{\circledast} \blacktriangleleft {}^\circ[\![\mathfrak{V}]\!])$ we have the following:
$\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$
and
$\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \max\{\mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}'), \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ)\} \overset{(5.2)}{=} \mathbb{EL}$
and
$\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \max\{\mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}'), \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ)\} \overset{(5.3)}{=} \mathbb{EL}$
and
$\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\overline{\mathfrak{V}}^{\mathbb{EL}}}({}^\circ[\![\mathfrak{V}]\!]) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v^2_{\circledast} \blacktriangleleft {}^\circ[\![\mathfrak{X}]\!], v^2_{\circledast} \blacktriangleleft [\![\mathfrak{X}]\!]^\circ, v^3_{\circledast} \blacktriangleleft \mathsf{cl}')$ we have the following:
$\mathcal{M}_A(\mathsf{cl}) \overset{(5.12)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$
and
$\mathcal{M}_B(\mathsf{cl}) \overset{(5.13)}{=} \max\{\mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}({}^\circ[\![\mathfrak{X}]\!]), \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ)\} \overset{(5.2)}{=} \mathbb{EL}$
and
$\mathcal{M}_C(\mathsf{cl}) \overset{(5.13)}{=} \max\{\mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{X}]\!]), \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ)\} \overset{(5.3)}{=} \mathbb{EL}$
and
$\mathcal{M}_D(\mathsf{cl}) \overset{(5.14)}{=} \mathcal{M}_{\overline{\mathfrak{V}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$.

Case 6: $\overline{\mathfrak{A};\mathfrak{X}}^{\mathbb{EL}} = \overline{\mathfrak{A}}^{\mathbb{EL}};\mathfrak{X}$. After denoting $\overline{\mathfrak{A};\mathfrak{X}}^{\mathbb{EL}} = A$ and $\overline{\mathfrak{A};\mathfrak{X}}^{\mathbb{EL}} = B$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\mathfrak{A};\mathfrak{X}]\!]$ we have the following:
$\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
and
$\mathcal{M}_B(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = [\![\mathfrak{A};\mathfrak{X}]\!]^\circ$ we have the following:
$\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
and
$\mathcal{M}_B(\mathsf{cl}) \overset{(5.7)}{=} \bot$.

- For $\mathsf{cl} = e_i \lhd (v^1_i \blacktriangleleft \mathsf{cl}')$ we have the following:
$\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
and
$\mathcal{M}_B(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = v_i^1 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbf{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.
- For $\mathsf{cl} = v_i^2 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.7)}{=} \bot$.
- For $\mathsf{cl} = i_i \lhd (v_i^1 \blacktriangleleft \lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor^\circ, v_i^2 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbf{EL}}}(\lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor^\circ) \overset{(5.2)}{=} \bot$.

Case 7: $\underline{\mathfrak{A}}_{\mathbf{EL}} ; \mathfrak{X} = \mathfrak{A} ; \overline{\mathfrak{X}}^{\mathbf{EL}}$. After denoting $\underline{\mathfrak{A}}_{\mathbf{EL}} ; \mathfrak{X} = A$ and $\mathfrak{A} ; \overline{\mathfrak{X}}^{\mathbf{EL}} = B$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ\lfloor\!\lfloor \mathfrak{A} ; \mathfrak{X} \rfloor\!\rfloor$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbf{EL}}}({}^\circ\lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.8)}{=} \bot$.
- For $\mathsf{cl} = \lfloor\!\lfloor \mathfrak{A} ; \mathfrak{X} \rfloor\!\rfloor^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.7)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.8)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbf{EL}}}(\lfloor\!\lfloor \mathfrak{X} \rfloor\!\rfloor^\circ) \overset{(5.2)}{=} \bot$.
- For $\mathsf{cl} = e_i \lhd (v_i^1 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbf{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.8)}{=} \bot$.
- For $\mathsf{cl} = v_i^1 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbf{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.8)}{=} \bot$.
- For $\mathsf{cl} = v_i^2 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.7)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.8)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbf{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.

- For $\mathsf{cl} = i_i \lhd (v_i^1 \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v_i^2 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.7)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.8)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$.

Case 8: $\mathfrak{A}\,;\underline{\mathfrak{X}}_{\mathbb{EL}} = \underline{\mathfrak{A}\,;\mathfrak{X}}_{\mathbb{EL}}$. After denoting $\mathfrak{A}\,;\underline{\mathfrak{X}}_{\mathbb{EL}} = A$ and $\underline{\mathfrak{A}\,;\mathfrak{X}}_{\mathbb{EL}} = B$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\mathfrak{A}\,;\mathfrak{X}]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.8)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = [\![\mathfrak{A}\,;\mathfrak{X}]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.8)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}([\![\mathfrak{X}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \mathbb{EL}$.
- For $\mathsf{cl} = e_i \lhd (v_i^1 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.8)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = v_i^1 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.8)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = v_i^2 \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.8)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = i_i \lhd (v_i^1 \blacktriangleleft [\![\mathfrak{A}]\!]^\circ, v_i^2 \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.8)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.

Case 9: $\overline{\mathfrak{A}\|\mathfrak{X}}^{\mathbb{EL}} = \overline{\mathfrak{A}}^{\mathbb{EL}} \| \overline{\mathfrak{X}}^{\mathbb{EL}}$. After denoting $\overline{\mathfrak{A}\|\mathfrak{X}}^{\mathbb{EL}} = A$ and $\overline{\mathfrak{A}}^{\mathbb{EL}} \| \overline{\mathfrak{X}}^{\mathbb{EL}} = B$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\mathfrak{A}\|\mathfrak{X}]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.10)}{=} min\{\mathbb{E},\mathbb{E}\} max\{\mathbb{L},\mathbb{L}\} = \mathbb{EL}$.

- For $\mathsf{cl} = [\![\mathfrak{A}\|\mathfrak{X}]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.10)}{=} \bot$.
- For $\mathsf{cl} = e^1_{\|} \lhd v^1_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$.
- For $\mathsf{cl} = e^2_{\|} \lhd v^2_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$.
- For $\mathsf{cl} = v^1_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.
- For $\mathsf{cl} = v^2_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\overline{\mathfrak{X}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$.

Case 10: $\mathfrak{A}_{\mathbb{EL}}\|\mathfrak{X}_{\mathbb{E}'\mathbb{L}'} = \underline{\mathfrak{A}\|\mathfrak{X}}_{\min\{\mathbb{E},\mathbb{E}'\}\max\{\mathbb{L},\mathbb{L}'\}}$. After denoting

$$\mathfrak{A}_{\mathbb{EL}}\|\mathfrak{X}_{\mathbb{E}'\mathbb{L}'} = A \text{ and } \underline{\mathfrak{A}\|\mathfrak{X}}_{\min\{\mathbb{E},\mathbb{E}'\}\max\{\mathbb{L},\mathbb{L}'\}} = B\,,$$

we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\mathfrak{A}\|\mathfrak{X}]\!]$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.10)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = [\![\mathfrak{A}\|\mathfrak{X}]\!]^\circ$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.10)}{=} \min\{\mathbb{E},\mathbb{E}'\}\max\{\mathbb{L},\mathbb{L}'\}$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \min\{\mathbb{E},\mathbb{E}'\}\max\{\mathbb{L},\mathbb{L}'\}$.
- For $\mathsf{cl} = e^1_{\|} \lhd v^1_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\mathfrak{A}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.

- For $\mathsf{cl} = e^2_{\|} \lhd v^2_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.

- For $\mathsf{cl} = v^1_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.

- For $\mathsf{cl} = v^2_{\|} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_A(\mathsf{cl}) \overset{(5.10)}{=} \mathcal{M}_{\underline{\mathfrak{X}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.3)}{=} \bot$.

Case 11: $\overline{\mathfrak{A}\,\mathsf{sc}\,A}^{\mathbb{EL}} = \overline{\mathfrak{A}}^{\mathbb{EL}}\,\mathsf{sc}\,A$. After denoting $\overline{\mathfrak{A}\,\mathsf{sc}\,A}^{\mathbb{EL}} = B$ and $\overline{\mathfrak{A}}^{\mathbb{EL}}\,\mathsf{sc}\,A = C$, we have a number of sub-cases:

- For $\mathsf{cl} = {}^{\circ}\llbracket \mathfrak{A}\,\mathsf{sc}\,A \rrbracket$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}({}^{\circ}\llbracket \mathfrak{A} \rrbracket) \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = \llbracket \mathfrak{A}\,\mathsf{sc}\,A \rrbracket^{\circ}$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\llbracket \mathfrak{A} \rrbracket^{\circ}) \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.2)}{=} \bot$.

- For $\mathsf{cl} = e_{\mathsf{sc}\,A} \lhd (v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.2)}{=} \mathbb{EL}$.

- For $\mathsf{cl} = v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\overline{\mathfrak{A}}^{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.2)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.2)}{=} \bot$.

Case 12: $\underline{\mathfrak{A}\,\mathsf{sc}\,A}_{\mathbb{EL}} = \underline{\mathfrak{A}}_{\mathbb{EL}}\,\mathsf{sc}\,A$. After denoting

$$\underline{\mathfrak{A}\,\mathsf{sc}\,A}_{\mathbb{EL}} = B \quad \text{and} \quad \underline{\mathfrak{A}}_{\mathbb{EL}}\,\mathsf{sc}\,A = C\,,$$

we have a number of sub-cases:

- For $\mathsf{cl} = {}^\circ[\![\mathfrak{A}\,\mathsf{sc}\,A]\!]$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}({}^\circ[\![\mathfrak{A}]\!]) \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = [\![\mathfrak{A}\,\mathsf{sc}\,A]\!]^\circ$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}([\![\mathfrak{A}]\!]^\circ) \overset{(5.3)}{=} \mathbb{EL}$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.3)}{=} \mathbb{EL}$.
- For $\mathsf{cl} = e_{\mathsf{sc}\,A} \triangleleft (v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}')$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.3)}{=} \bot$.
- For $\mathsf{cl} = v_{\mathsf{sc}\,A} \blacktriangleleft \mathsf{cl}'$ we have the following:
  $\mathcal{M}_B(\mathsf{cl}) \overset{(5.16)}{=} \mathcal{M}_{\underline{\mathfrak{A}}_{\mathbb{EL}}}(\mathsf{cl}') \overset{(5.3)}{=} \bot$
  and
  $\mathcal{M}_C(\mathsf{cl}) \overset{(5.3)}{=} \bot$.

□

5.3.4 Structural equivalence

We now want to capture situations where different applications of a same operator box lead to the same cat-box. We start by defining three auxiliary relations which are the smallest equivalence relations on pairs of cat-boxes satisfying the following (below $\mathfrak{A}$, $\mathfrak{X}$ and $\mathfrak{V}$ are static cat-boxes and $\mathbb{EL} \in \mathbb{D}$):

- $(\overline{\mathfrak{A}}^{\mathbb{EL}}, \mathfrak{X}) \equiv_\Box (\mathfrak{A}, \overline{\mathfrak{X}}^{\mathbb{EL}})$ and $(\underline{\mathfrak{A}}_{\mathbb{EL}}, \mathfrak{X}) \equiv_\Box (\mathfrak{A}, \underline{\mathfrak{X}}_{\mathbb{EL}})$.
- $(\underline{\mathfrak{A}}_{\mathbb{EL}}, \mathfrak{X}) \equiv_; (\mathfrak{A}, \overline{\mathfrak{X}}^{\mathbb{EL}})$.
- $(\underline{\mathfrak{A}}_{\mathbb{EL}}, \mathfrak{X}, \mathfrak{V}) \equiv_\circledast (\mathfrak{A}, \overline{\mathfrak{X}}^{\mathbb{EL}}, \mathfrak{V}) \equiv_\circledast (\mathfrak{A}, \underline{\mathfrak{X}}_{\mathbb{EL}}, \mathfrak{V}) \equiv_\circledast (\mathfrak{A}, \mathfrak{X}, \overline{\mathfrak{V}}^{\mathbb{EL}})$.

Moreover, $\equiv_\|$ is the identity on the pairs of cat-boxes.

Proposition 5.12. *Let $\mathfrak{A}, \mathfrak{A}', \mathfrak{X}$ and $\mathfrak{X}'$ be cat-boxes.*

1. $\mathfrak{A}\Box\mathfrak{X} = \mathfrak{A}'\Box\mathfrak{X}'$ iff $(\mathfrak{A}, \mathfrak{X}) \equiv_\Box (\mathfrak{A}', \mathfrak{X}')$.
2. $\mathfrak{A}\,;\mathfrak{X} = \mathfrak{A}'\,;\mathfrak{X}'$ iff $(\mathfrak{A}, \mathfrak{X}) \equiv_; (\mathfrak{A}', \mathfrak{X}')$.
3. $\mathfrak{A}\|\mathfrak{X} = \mathfrak{A}'\|\mathfrak{X}'$ iff $(\mathfrak{A}, \mathfrak{X}) \equiv_\| (\mathfrak{A}', \mathfrak{X}')$.
4. $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A}' \circledast \mathfrak{X}' \circledast \mathfrak{V}' \rangle\!\rangle$ iff $(\mathfrak{A}, \mathfrak{X}, \mathfrak{V}) \equiv_\circledast (\mathfrak{A}', \mathfrak{X}', \mathfrak{V}')$.

Proof. If $(\mathfrak{A}, \mathfrak{X}) = (\mathfrak{A}', \mathfrak{X}')$ then the proof is trivial. We therefore assume that $(\mathfrak{A}, \mathfrak{X}) \neq (\mathfrak{A}', \mathfrak{X}')$ and then consider four cases.

Case 1: $\mathfrak{A}\Box\mathfrak{X} = \mathfrak{A}'\Box\mathfrak{X}'$ iff $(\mathfrak{A},\mathfrak{X}) \equiv_{\Box} (\mathfrak{A}',\mathfrak{X}')$.
($\Longleftarrow$) Without loss of generality

$$\mathfrak{A} = \overline{\mathfrak{A}'}^{\mathbb{EL}} \text{ and } \mathfrak{X}' = \overline{\mathfrak{X}}^{\mathbb{EL}} .$$

Then $\overline{\mathfrak{A}'}^{\mathbb{EL}}\Box\mathfrak{X} = \mathfrak{A}'\Box\overline{\mathfrak{X}}^{\mathbb{EL}}$ follows from proposition 5.11(1)
($\Longrightarrow$) We first observe that $\mathfrak{A}\Box\mathfrak{X} = \mathfrak{A}'\Box\mathfrak{X}'$ implies

$$\lfloor\mathfrak{A}\rfloor\Box\lfloor\mathfrak{X}\rfloor = \lfloor\mathfrak{A}'\rfloor\Box\lfloor\mathfrak{X}'\rfloor .$$

Hence, from the results of the standard box algebra it follows that, without loss of generality, $\lfloor\mathfrak{A}\rfloor = \overline{\lfloor\mathfrak{A}'\rfloor}$ and $\lfloor\mathfrak{X}'\rfloor = \overline{\lfloor\mathfrak{X}\rfloor}$. Consequently, $\mathfrak{A}'$ and $\mathfrak{X}$ must be of the form:

$$\mathfrak{A} = \overline{\mathfrak{A}'}^{\mathbb{EL}} \text{ and } \mathfrak{X}' = \overline{\mathfrak{X}}^{\mathbb{E}'\mathbb{L}'} ,$$

for some $\mathbb{EL}, \mathbb{E}'\mathbb{L}' \in \mathbb{D}$. All we need to show now is that $\mathbb{EL} = \mathbb{E}'\mathbb{L}'$. From our hypothesis and the proof of proposition 5.11, we know that:

$$\mathcal{M}_{\overline{\mathfrak{A}'}^{\mathbb{EL}}\Box\mathfrak{X}}(\mathsf{cl}) = \mathcal{M}_{\mathfrak{A}'\Box\overline{\mathfrak{X}}^{\mathbb{E}'\mathbb{L}'}}(\mathsf{cl}) \Longleftrightarrow \mathbb{EL} = \mathbb{E}'\mathbb{L}' ,$$

for the cluster $\mathsf{cl} = e_{\Box}\lhd(v^1_{\Box} \blacktriangleleft {}^{\circ}\lfloor\mathfrak{A}\rfloor, v^2_{\Box} \blacktriangleleft {}^{\circ}\lfloor\mathfrak{X}\rfloor)$. Hence $\mathbb{EL} = \mathbb{E}'\mathbb{L}'$.

Case 2: $\mathfrak{A};\mathfrak{X} = \mathfrak{A}';\mathfrak{X}'$ iff $(\mathfrak{A},\mathfrak{X}) \equiv_{;} (\mathfrak{A}',\mathfrak{X}')$.
($\Longleftarrow$) Without loss of generality

$$\mathfrak{A} = \underline{\mathfrak{A}'}_{\mathbb{EL}} \text{ and } \mathfrak{X}' = \overline{\mathfrak{X}}^{\mathbb{EL}} .$$

Then $\underline{\mathfrak{A}'}_{\mathbb{EL}};\mathfrak{X} = \mathfrak{A}';\overline{\mathfrak{X}}^{\mathbb{EL}}$ follows from proposition 5.11(3).
($\Longrightarrow$) We first observe that $\mathfrak{A};\mathfrak{X} = \mathfrak{A}';\mathfrak{X}'$ implies

$$\lfloor\mathfrak{A}\rfloor;\lfloor\mathfrak{X}\rfloor = \lfloor\mathfrak{A}'\rfloor;\lfloor\mathfrak{X}'\rfloor .$$

Hence, from the results of the standard box algebra it follows that, without loss of generality, $\lfloor\mathfrak{A}\rfloor = \underline{\lfloor\mathfrak{A}'\rfloor}$ and $\lfloor\mathfrak{X}'\rfloor = \overline{\lfloor\mathfrak{X}\rfloor}$. Consequently, $\mathfrak{A}$ and $\mathfrak{X}'$ must be of the form:

$$\mathfrak{A} = \underline{\mathfrak{A}'}_{\mathbb{EL}} \text{ and } \mathfrak{X}' = \overline{\mathfrak{X}}^{\mathbb{E}'\mathbb{L}'} ,$$

for some $\mathbb{EL}, \mathbb{E}'\mathbb{L}' \in \mathbb{D}$. All we need to show now is that $\mathbb{EL} = \mathbb{E}'\mathbb{L}'$. From our hypothesis and the proof of proposition 5.11 we know that:

$$\mathcal{M}_{\underline{\mathfrak{A}'}_{\mathbb{EL}};\mathfrak{X}}(\mathsf{cl}) = \mathcal{M}_{\mathfrak{A}';\overline{\mathfrak{X}}^{\mathbb{E}'\mathbb{L}'}}(\mathsf{cl}) \Longleftrightarrow \mathbb{EL} = \mathbb{E}'\mathbb{L}'$$

for any cluster $\mathsf{cl} = i_{;}\lhd(v^1_{;} \blacktriangleleft \lfloor\mathfrak{A}\rfloor^{\circ}, v^2_{;} \blacktriangleleft \mathsf{cl}')$. Hence $\mathbb{EL} = \mathbb{E}'\mathbb{L}'$.

Case 3: $\mathfrak{A}\|\mathfrak{X} = \mathfrak{A}'\|\mathfrak{X}'$ iff $(\mathfrak{A},\mathfrak{X}) \equiv_{\|} (\mathfrak{A}',\mathfrak{X}')$.
($\Longleftarrow$) Then $\mathfrak{A} = \mathfrak{A}'$ and $\mathfrak{X} = \mathfrak{X}'$, and so $\mathfrak{A}\|\mathfrak{X} = \mathfrak{A}'\|\mathfrak{X}'$.
($\Longrightarrow$) We first observe that $\mathfrak{A}\|\mathfrak{X} = \mathfrak{A}'\|\mathfrak{X}'$ implies

$$\lfloor\mathfrak{A}\rfloor\|\lfloor\mathfrak{X}\rfloor = \lfloor\mathfrak{A}'\rfloor\|\lfloor\mathfrak{X}'\rfloor .$$

Hence, from the results of the standard box algebra it follows that $\lfloor\mathfrak{A}\rfloor = \lfloor\mathfrak{A}'\rfloor$ and $\lfloor\mathfrak{X}\rfloor = \lfloor\mathfrak{X}'\rfloor$. It is then easy to see that $\mathfrak{A} = \mathfrak{A}'$ and $\mathfrak{X} = \mathfrak{X}'$.

Case 4: $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A}' \circledast \mathfrak{X}' \circledast \mathfrak{V}' \rangle\!\rangle$ iff $(\mathfrak{A}, \mathfrak{X}, \mathfrak{V}) \equiv_{\circledast} (\mathfrak{A}', \mathfrak{X}', \mathfrak{V}')$. Without loss of generality

$$\mathfrak{A} = \underline{\mathfrak{A}'}_{\mathbb{EL}} \text{ and } \mathfrak{X}' = \overline{\mathfrak{X}}^{\mathbb{EL}} \text{ and } \mathfrak{V} = \mathfrak{V}' .$$

Then $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A}' \circledast \mathfrak{X}' \circledast \mathfrak{V}' \rangle\!\rangle$ follows from proposition 5.11(2).
($\Longrightarrow$) We first observe that $\langle\!\langle \mathfrak{A} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle = \langle\!\langle \mathfrak{A}' \circledast \mathfrak{X}' \circledast \mathfrak{V}' \rangle\!\rangle$ implies

$$\langle\!\langle \lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor \circledast \lfloor\!\lfloor \mathfrak{X} \rfloor\!\rfloor \circledast \lfloor\!\lfloor \mathfrak{V} \rfloor\!\rfloor \rangle\!\rangle = \langle\!\langle \lfloor\!\lfloor \mathfrak{A}' \rfloor\!\rfloor \circledast \lfloor\!\lfloor \mathfrak{X}' \rfloor\!\rfloor \circledast \lfloor\!\lfloor \mathfrak{V}' \rfloor\!\rfloor \rangle\!\rangle .$$

Hence, from the results of the standard box algebra it follows that, without loss of generality, $\lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor = \lfloor\!\lfloor \underline{\mathfrak{A}'} \rfloor\!\rfloor$, $\lfloor\!\lfloor \mathfrak{X}' \rfloor\!\rfloor = \lfloor\!\lfloor \overline{\mathfrak{X}} \rfloor\!\rfloor$ and, moreover, $\lfloor\!\lfloor \mathfrak{V} \rfloor\!\rfloor = \lfloor\!\lfloor \mathfrak{V}' \rfloor\!\rfloor$ is a static at-net. Consequently, $\mathfrak{A}$ and $\mathfrak{X}'$ must be of the form:

$$\mathfrak{A} = \underline{\mathfrak{A}'}_{\mathbb{EL}} \text{ and } \mathfrak{X}' = \overline{\mathfrak{X}}^{\mathbb{E}'\mathbb{L}'} ,$$

for some $\mathbb{EL}, \mathbb{E}'\mathbb{L}' \in \mathbb{D}$, and $\mathfrak{V} = \mathfrak{V}'$. All we need to show now is that $\mathbb{EL} = \mathbb{E}'\mathbb{L}'$. From our hypothesis and the proof of proposition 5.11 we know that:

$$\mathcal{M}_{\langle\!\langle \underline{\mathfrak{A}'}_{\mathbb{EL}} \circledast \mathfrak{X} \circledast \mathfrak{V} \rangle\!\rangle}(\mathsf{cl}) = \mathcal{M}_{\langle\!\langle \mathfrak{A}' \circledast \overline{\mathfrak{X}}^{\mathbb{E}'\mathbb{L}'} \circledast \mathfrak{V} \rangle\!\rangle}(\mathsf{cl}) \Longleftrightarrow \mathbb{EL} = \mathbb{E}'\mathbb{L}'$$

for any cluster $\mathsf{cl} = i_{\circledast} \lhd (v^1_{\circledast} \blacktriangleleft \lfloor\!\lfloor \mathfrak{A} \rfloor\!\rfloor^{\circ}, v^2_{\circledast} \blacktriangleleft \mathsf{cl}', v^2_{\circledast} \blacktriangleleft \lfloor\!\lfloor \mathfrak{X} \rfloor\!\rfloor^{\circ}, v^3_{\circledast} \blacktriangleleft {}^{\circ}\lfloor\!\lfloor \mathfrak{V} \rfloor\!\rfloor)$. Hence $\mathbb{EL} = \mathbb{E}'\mathbb{L}'$. □

5.3.5 Structural execution of transition steps

We now provide a characterisation of steps executed by cat-boxes which reflects the compositional way in which they have been defined, providing a direct link to the execution rules of the corresponding at-expressions.

Proposition 5.13. *Let $\Omega_{op} \in \{\Omega_{\square}, \Omega_{\circledast}, \Omega_{;}, \Omega_{\|}\}$ be any n-unary ($n \geq 2$) operator box and $\widetilde{\mathfrak{A}} = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$ be a tuple of static and dynamic cat-boxes in its domain of application.*

1. *If $\mathfrak{A}_i[U_i\rangle\mathfrak{X}_i$ (for $i \leq n$), then $\widetilde{\mathfrak{X}} = (\mathfrak{X}_1, \ldots, \mathfrak{X}_n)$ is in the domain of application of Ω_{op} and $\Omega_{op}(\widetilde{\mathfrak{A}})[U\rangle\Omega_{op}(\widetilde{\mathfrak{X}})$, where*

$$U = (v^1_{op} \blacktriangleleft U_1) \cup \ldots \cup (v^n_{op} \blacktriangleleft U_n) . \tag{5.18}$$

2. *If $\Omega_{op}(\widetilde{\mathfrak{A}})[U\rangle\mathfrak{K}$, then there are tuples $\widetilde{\mathfrak{X}}, \widetilde{\mathfrak{V}}$ of cat-boxes in the application domain of Ω_{op} as well as steps $U_1, \ldots, U_n$ (some of them possibly empty) such that (5.18) holds, $\widetilde{\mathfrak{A}} \equiv_{\Omega_{op}} \widetilde{\mathfrak{X}}$, $\mathfrak{X}_i[U_i\rangle\mathfrak{V}_i$ (for $i \leq n$) and $\mathfrak{K} = \Omega_{op}(\widetilde{\mathfrak{V}})$.*

Note: As a consequence, $\mathsf{enabled}(\Omega_{op}(\widetilde{\mathfrak{A}}))$ *comprises exactly all sets*

$$(v^1_{op} \blacktriangleleft U_1) \cup \ldots \cup (v^n_{op} \blacktriangleleft U_n)$$

of transitions such that there is $\widetilde{\mathfrak{X}} = (\mathfrak{X}_1, \ldots, \mathfrak{X}_n)$ satisfying $\widetilde{\mathfrak{X}} \equiv_{\Omega_{op}} \widetilde{\mathfrak{A}}$ and $U_i \in \mathsf{enabled}(\mathfrak{X}_i)$ (for $i \leq n$).

Proof. Follows from similar results holding in the standard box algebra, proposition 5.12, and the fact that the age of tokens and the time annotations are consistently inherited through the composition operation specified by Ω_{op}. □

Proposition 5.14. *Let $\mathfrak{A}$ be a dynamic cat-box and $A \subseteq \mathcal{A}$.*

1. *If $\mathfrak{A}[\{t_1, u_1, \ldots, t_k, u_k, w_1, \ldots, w_m\}\rangle\mathfrak{X}$ where $\lambda_{\lfloor\mathfrak{A}\rfloor}(t_i) = \widehat{\lambda_{\lfloor\mathfrak{A}\rfloor}(u_i)} \in A$ for all $i \leq k$ and $\lambda_{\lfloor\mathfrak{A}\rfloor}(w_j) \notin A$ for all $j \leq m$, then $\Omega_{\mathsf{sc}\,A}(\mathfrak{A})[U\rangle\Omega_{\mathsf{sc}\,A}(\mathfrak{X})$, where*

$$U = \{v_{\mathsf{sc}\,A} \lhd \{t_1, u_1\}, \ldots, v_{\mathsf{sc}\,A} \lhd \{t_k, u_k\}, v_{\mathsf{sc}\,A} \lhd w_1, \ldots, v_{\mathsf{sc}\,A} \lhd w_m\} \ . \quad (5.19)$$

2. *If $\Omega_{\mathsf{sc}\,A}(\mathfrak{A})[U\rangle\mathfrak{K}$ then there are transitions $t_1, u_1, \ldots, t_k, u_k, w_1, \ldots, w_m$ and a cat-box $\mathfrak{X}$ as in part (1) which satisfy $\mathfrak{K} = \Omega_{\mathsf{sc}\,A}(\mathfrak{X})$ and (5.19).*

Note: As a consequence, $\mathsf{enabled}(\Omega_{\mathsf{sc}\,A}(\mathfrak{A}))$ *comprises exactly all*

$$U = \{v_{\mathsf{sc}\,A} \lhd \{t_1, u_1\}, \ldots, v_{\mathsf{sc}\,A} \lhd \{t_k, u_k\}, v_{\mathsf{sc}\,A} \lhd w_1, \ldots, v_{\mathsf{sc}\,A} \lhd w_m\}$$

such that $\lambda_{\lfloor\mathfrak{A}\rfloor}(t_i) = \widehat{\lambda_{\lfloor\mathfrak{A}\rfloor}(u_i)} \in A$ for all $i \leq k$ and $\lambda_{\lfloor\mathfrak{A}\rfloor}(w_j) \notin A$ for all $j \leq m$.

Proof. Follows from a similar result holding in the standard box algebra, and the fact that the age of tokens and the time annotations are consistently inherited through the composition operation specified by $\Omega_{\mathsf{sc}\,A}$. □

5.3.6 Structural characterisation of urgent transitions

We now provide a compositional characterisation of urgent transitions of cat-boxes.

Proposition 5.15. *Let $\Omega_{op} \in \{\Omega_\square, \Omega_\circledast, \Omega_;, \Omega_\|\}$ be any n-unary ($n \geq 2$) operator box and $\widetilde{\mathfrak{A}} = (\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$ be a tuple of static and dynamic cat-boxes in its domain of application.*

1. *If $t \in \mathsf{urgent}(\mathfrak{A}_i)$, for some $i \leq n$, then $v^i_{op} \lhd t \in \mathsf{urgent}(\Omega_{op}(\widetilde{\mathfrak{A}}))$.*

2. *If $v^i_{op} \lhd t \in \mathsf{urgent}(\Omega_{op}(\widetilde{\mathfrak{A}}))$, for some $i \leq n$, then there is a tuple $\widetilde{\mathfrak{X}} = (\mathfrak{X}_1, \ldots, \mathfrak{X}_n)$ of cat-boxes in the application domain of Ω_{op} such that $\widetilde{\mathfrak{A}} \equiv_{\Omega_{op}} \widetilde{\mathfrak{X}}$ and $t \in \mathsf{urgent}(\mathfrak{X}_i)$.*

Proof. Follows from the note in the formulation of proposition 5.13, and the fact that the age of tokens and the time annotations are consistently inherited through the composition operation specified by Ω_{op}. □

Proposition 5.16. *Let $\mathfrak{A}$ be a dynamic cat-box, $A \subseteq \mathcal{A}$ and $v_{\mathsf{sc}\,A} \lhd U \in T_{\Omega_{\mathsf{sc}\,A}(\mathfrak{A})}$. Then*

$$v_{\mathsf{sc}\,A} \lhd U \in \mathsf{urgent}(\Omega_{\mathsf{sc}\,A}(\llbracket\mathfrak{A}\rrbracket)) \iff U \cap \mathsf{urgent}(\mathfrak{A}) \neq \varnothing \ .$$

Proof. Follows from the note in the formulation of proposition 5.14, and the fact that the age of tokens and the time annotations are consistently inherited through the composition operation specified by $\Omega_{\mathsf{sc}\,A}$. □

5.3.7 From at-expressions to cat-boxes

We now provide a compositional translation from at-expressions to cat-boxes. The mapping cBox from at-expressions to cat-boxes is defined so that:

$$
\begin{aligned}
\mathsf{cBox}(\alpha el) &\stackrel{\mathrm{df}}{=} \mathsf{N}^{\mathsf{cat}}_{\alpha el} \\
\mathsf{cBox}(\overline{H}^{\mathbf{EL}}) &\stackrel{\mathrm{df}}{=} \overline{\mathsf{cBox}(H)}^{\mathbf{EL}} \\
\mathsf{cBox}(\underline{H}_{\mathbf{EL}}) &\stackrel{\mathrm{df}}{=} \underline{\mathsf{cBox}(H)}_{\mathbf{EL}} \\
\mathsf{cBox}(H\,\mathsf{sc}\,A) &\stackrel{\mathrm{df}}{=} \mathsf{cBox}(H)\,\mathsf{sc}\,A \\
\mathsf{cBox}(H \Box J) &\stackrel{\mathrm{df}}{=} \mathsf{cBox}(H) \Box \mathsf{cBox}(J) \\
\mathsf{cBox}(H \| J) &\stackrel{\mathrm{df}}{=} \mathsf{cBox}(H) \| \mathsf{cBox}(J) \\
\mathsf{cBox}(H\,;J) &\stackrel{\mathrm{df}}{=} \mathsf{cBox}(H)\,;\mathsf{cBox}(J) \\
\mathsf{cBox}(\langle\!\langle H \circledast J \circledast I \rangle\!\rangle) &\stackrel{\mathrm{df}}{=} \langle\!\langle \mathsf{cBox}(H) \circledast \mathsf{cBox}(J) \circledast \mathsf{cBox}(I) \rangle\!\rangle\,,
\end{aligned}
$$

where $\mathsf{N}^{\mathsf{cat}}_{\alpha el}$ is $\mathsf{N}_{\alpha el}$ with the token filling mapping returning only $\perp$. The semantical mapping always returns a cat-box, and the property of corresponding to a static or dynamic box has been captured by the syntax (3.1,3.2).

Proposition 5.17. *Let H be an at-expression.*

1. $\mathsf{cBox}(H)$ *is a static or dynamic cat-box.*
2. $\mathsf{cBox}(H)$ *is a static cat-box* iff H *is a static at-expression.*

Proof. Follows by induction on the structure of the at-expressions, using similar results holding in the standard box algebra. □

Chapter 6

Behavioural Relationships

6.1 Relationship between at-expressions and cat-boxes

The consistency between the denotational and the operational semantics of at-expressions will be formulated in terms of the full transition systems they generate. We now have a fundamental result which demonstrates that the operational and denotational semantics of an at-expression capture the same behaviour.

Theorem 6.1. *For every at-expression H,*

$$\mathsf{iso}_H \stackrel{\text{df}}{=} \{([J]_{\equiv}, \mathsf{cBox}(J)) \mid [J]_{\equiv} \textit{ is a node of } \mathsf{fTS}_H\}$$

is an isomorphism between the transition systems fTS_H and $\mathsf{fTS}_{\mathsf{cBox}(H)}$.

Proof. We proceed by induction on the structure of H. The result clearly holds when $\lfloor H \rfloor = \alpha el$. In the inductive step we do not need to consider H which is completely overbarred or underbarred (since then a rewriting, based on the rules in table 3.1, can be applied to push the bar inside the expression). After that we consider various cases for executing (transition or time) steps from H as well as $\mathsf{cBox}(H)$, and derive the appropriate steps in the counterpart node using the operational semantics rules, propositions 5.13, 5.14, 5.15, 5.16 and 5.17 as well as $\mathsf{cBox}(H^{\surd}) = \mathsf{cBox}(H)^{\surd}$. □

From the above result, a number of immediate corollaries can be derived, as stated next.

Theorem 6.2. *For every at-expression H and the corresponding cat-box $\mathsf{Box}(H)$, we have that:*

1. TS_H *and* $\mathsf{TS}_{\mathsf{Box}(H)}$ *are isomorphic.*
2. fRT_H *and* $\mathsf{fRT}_{\mathsf{Box}(H)}$ *are isomorphic.*
3. RT_H *and* $\mathsf{RT}_{\mathsf{Box}(H)}$ *are isomorphic.*

Proof. Follows from theorem 6.1 and the fact that, for both expressions and boxes, moving from a transition-based graph representing global behaviour to a label-based graph amounts to replacing in the original arcs all the U's by their multisets of communication labels (duplicate arcs are then deleted). □

6.2 Relationship between at-boxes and cat-boxes

We are now going to relate the global behaviour of at-boxes and cat-boxes. This time, however, the main correspondence result will be expressed in terms of reachability trees rather than transition systems.

Let $\Theta = (\Sigma, \mu)$ be an input-reachable at-box. Then $¥(\Theta) \stackrel{\text{df}}{=} (\Sigma, \mathcal{M})$ where

$$¥(\mu) : CL_\Sigma \to \mathbb{D}^\perp$$

is a cluster filling mapping such that, for every $\mathsf{cl} \in CL_\Sigma$:

$$¥(\mu)(\mathsf{cl}) \stackrel{\text{df}}{=} \begin{cases} \perp & \text{if } M_\Sigma(\mathsf{cl}) = \{\perp\} \\ \mathbb{E}\mathbb{L} & \text{otherwise,} \end{cases}$$

with $\mathbb{E} = \min(\mu(\mathsf{cl} \cap M_\Sigma))$ and $\mathbb{L} = \max(\mu(\mathsf{cl} \cap M_\Sigma))$. It is easy to see that $¥(\Theta)$ is a cat-box since the two conditions from the definition of a cat-box are satisfied due to the two corresponding conditions in the definition of an at-box.

Proposition 6.1. *Let $\Theta = (\Sigma, \mu)$ be an input-reachable at-box, $\mathfrak{A} = ¥(\Theta) = (\Sigma, \mathcal{M})$, and $t \in T_\Sigma$.*

1. $t \in \mathsf{enabled}(\Theta)$ iff $t \in \mathsf{enabled}(\mathfrak{A})$.
2. $t \in \mathsf{urgent}(\Theta)$ iff $t \in \mathsf{urgent}(\mathfrak{A})$.
3. *$\surd$ is enabled in Θ* iff *$\surd$ is enabled in $\mathfrak{A}$.*
4. *If $\Theta[\{t\}\rangle\Xi$ then $\mathfrak{A}[\{t\}\rangle¥(\Xi)$.*
5. *If $\Theta[\surd\rangle\Xi$ then $\mathfrak{A}[\surd\rangle¥(\Xi)$.*

Proof. (1,2) ($\Longrightarrow$) Suppose that $t \in \mathsf{enabled}(\Theta)$ and $\mathsf{cl} \in {}^\diamond t$. Then, by proposition 5.6, we have that $\mathsf{cl} \subseteq {}^\bullet t$ and $\lambda_{\mathfrak{A}}(\mathsf{cl}, t) = \lambda_\Sigma(p, t)$, for all $p \in \mathsf{cl}$. Thus, since $\mu(p)$ tsat $\lambda_\Sigma(p, t)$, for all $p \in \mathsf{cl}$, we have $\mathcal{M}(\mathsf{cl})$ tsat $\lambda_{\mathfrak{A}}(\mathsf{cl}, t)$. Moreover, if $t \in \mathsf{urgent}(\Theta)$ then $t \in \mathsf{urgent}(\mathfrak{A})$ since, for any set of integers $K = \{k_1, \ldots, k_l\}$, we have

$$\begin{aligned} \min\{1 + k_1, \ldots, 1 + k_l\} &= 1 + \min K \\ \max\{1 + k_1, \ldots, 1 + k_l\} &= 1 + \max K\,. \end{aligned} \tag{6.1}$$

($\Longleftarrow$) Suppose that $t \in \mathsf{enabled}(\mathfrak{A})$ and $p \in {}^\bullet t$. Then, by proposition 5.7, there is $\mathsf{cl} \subseteq {}^\bullet t$ such that $p \in \mathsf{cl}$. After that we proceed by essentially reversing the argument for the ($\Longrightarrow$) implication.

(3) Follows from part (2).

(4) Let $\Sigma[\{t\}\rangle\Psi$ and $\Xi = (\Psi, \nu)$. By part (1), there is a cat-box $\mathfrak{X} = (\Psi, \mathcal{N})$ such that $\mathfrak{A}[\{t\}\rangle\mathfrak{X}$. All we need to show is that $\mathcal{N} = ¥(\nu)$. To this end we take $\mathsf{cl} \in CL_\Sigma$. If $({}^\bullet t \cup t^\bullet) \cap \mathsf{cl} = \varnothing$ then

$$\mathcal{N}(\mathsf{cl}) = \mathcal{M}(\mathsf{cl}) = ¥(\mu(\mathsf{cl})) = ¥(\nu)(\mathsf{cl})$$

clearly holds. So, we assume that $({}^\bullet t \cup t^\bullet) \cap \mathsf{cl} \neq \varnothing$ and then consider three cases.

Case 1: $\mathsf{cl} \subseteq {}^\circ\Sigma$. Due to the ex-directedness of Σ, we have that ${}^\bullet t \cap \mathsf{cl} \neq \varnothing$ and $t^\bullet \cap \mathsf{cl} = \varnothing$. Hence we have the following:

- If $M_\Psi = \varnothing$ then $\mathcal{N}(\mathsf{cl}) = \bot = ¥(\nu)(\mathsf{cl})$.
- If $M_\Psi \neq \varnothing$ then $\mathcal{N}(\mathsf{cl}) = \mathcal{M}(\mathsf{cl})$, by definition of a step in cat-boxes. On the other hand, the second condition in the definition of an at-box guarantees that $¥(\nu)(\mathsf{cl}) = ¥(\mu)(\mathsf{cl})$. Hence $\mathcal{N}(\mathsf{cl}) = ¥(\nu)(\mathsf{cl})$.

Case 2: $\mathsf{cl} \subseteq \Sigma^\circ$. Due to the ex-directedness of Σ, we have that $t^\bullet \cap \mathsf{cl} \neq \varnothing$ and ${}^\bullet t \cap \mathsf{cl} = \varnothing$. Hence we have the following:

- If $\mathcal{M}(\mathsf{cl}) = \mathbb{EL}$ then $\mathcal{N}(\mathsf{cl}) = 0\mathbb{L}$. On the other hand, $\nu(\mathsf{cl}) = \mu(\mathsf{cl}) \cup \{0\}$ and so $¥(\nu)(\mathsf{cl}) = 0\mathbb{L}$.
- If $\mathcal{M}(\mathsf{cl}) = \bot$ then $\mathcal{N}(\mathsf{cl}) = 00$. On the other hand, $\nu(\mathsf{cl}) = \{0\}$ and so $¥(\nu)(\mathsf{cl}) = 00$.

Case 3: $\mathsf{cl} \subseteq \ddot{\Sigma}$. By proceeding similarly as above, we may verify the property when $M_\Sigma \cap \mathsf{cl} = \varnothing$ or $M_\Psi \cap \mathsf{cl} = \varnothing$ or $t^\bullet \cap \mathsf{cl} \neq \varnothing$ (which, by proposition 4.4 means that ${}^\bullet t \cap \mathsf{cl} \neq \varnothing$). The only situation which needs consideration is when:

$$M_\Sigma \cap \mathsf{cl} \neq \varnothing \neq M_\Psi \cap \mathsf{cl} \quad \text{and} \quad {}^\bullet t \cap \mathsf{cl} \neq \varnothing = t^\bullet \cap \mathsf{cl}\,.$$

We then have $\mathcal{N}(\mathsf{cl}) = \mathcal{M}(\mathsf{cl})$, and so it suffices to show that $\nu(\mathsf{cl} \cap M_\Psi) = \mu(\mathsf{cl} \cap M_\Sigma)$.

From proposition 5.5 it follows that we had Case 2 situation when t was executed. Moreover, if we look at the tokens residing in the places of cl we observe that they age uniformly and, crucially, if two tokens were produced by firing of the same transition filling the cluster, and they are still present in Θ then their age given by μ is exactly the same.

From proposition 5.5 it follows that we must have had Case 2 situation when executing t. Therefore, we have that $\nu(\mathsf{cl} \cap M_\Psi) \subseteq \mu(\mathsf{cl} \cap M_\Sigma)$. Suppose now that $p \in (\mathsf{cl} \cap M_\Sigma) \setminus (\mathsf{cl} \cap M_\Psi)$ and that u was the transition which for the last time filled p with a timed token. Furthermore, without loss of generality, assume that cl (or, more precisely, its predecessor) has been formed by an application of the sequence operator on nets, $\Phi\,;\Phi'$. We therefore had a number of transitions $t_1, \ldots, t_m$ which were predecessors of the transitions emptying the cluster cl since the last time it has been filled. We note that there was at least one place q in

${}^\circ\Phi' \setminus {}^\bullet\{t_1, \ldots, t_m\}$ because $M_\Psi \cap \mathsf{cl} \neq \varnothing$. Let $r \in \Phi^\circ$ be any output place of a transition which was a predecessor of u. In the interface region $\Phi ; \Phi'$ there existed then a place resulting from a combination of r and q. Its successor is then present in $\mathsf{cl} \cap M_\Psi$ and it has been filled for the last time by transition u at the same time as p. It therefore follows that $\mu(p) \in \nu(\mathsf{cl} \cap M_\Psi)$, and so $\mu(\mathsf{cl} \cap M_\Sigma) \subseteq \nu(\mathsf{cl} \cap M_\Psi)$

(5) By part (3), $\surd$ is enabled in $¥(\Theta)$. Moreover, we have $¥(\Theta)[\surd)¥(\Xi)$ by property (6.1). □

Theorem 6.3. *Let Θ be an input-reachable at-box. Then the following hold.*

1. fTS_Θ *is strongly bisimilar (see [46]) to* $\mathsf{fTS}_{¥(\Theta)}$.
2. TS_Θ *is strongly bisimilar to* $\mathsf{TS}_{¥(\Theta)}$.
3. fRT_Θ *is isomorphic to* $\mathsf{fRT}_{¥(\Theta)}$.
4. RT_Θ *is isomorphic to* $\mathsf{RT}_{¥(\Theta)}$.

Proof. (1) Follows from propositions 5.3, 5.10 and 6.1, using the mapping ¥ to relate the nodes of the two transition systems.

(2) This is an immediate consequence of part (1).

(3) Follows from part (1) and the fact that both transition systems are deterministic (no annotation can label two different arrows outgoing from a node of the trees; this follows from the properties of transition systems of Petri nets, and the properties of the evolutions in the box algebra[1]).

(4) This is an immediate consequence of part (3). □

6.3 Relationship between at-expressions and at-boxes

After showing the relationships between of at-expressions with cat-boxes and of at-boxes with cat-boxes, is now possible to formulate the main result of this thesis showing the strong relation of at-expressions with at-boxes.

Theorem 6.4. *Let $G = \overline{E}^{00}$ be an initial dynamic at-expression and $\Theta = \overline{\mathsf{Box}(E)}^{00}$ be the corresponding at-box. Then the following hold.*

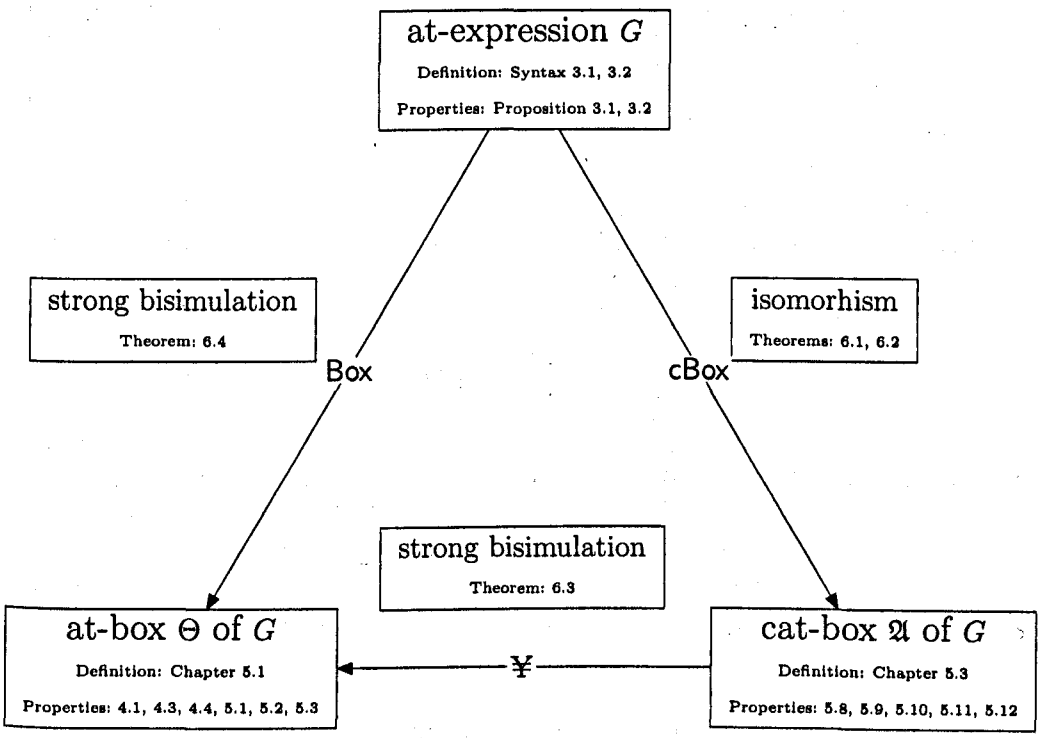
1. fTS_G *is strongly bisimilar to* fTS_Θ.
2. TS_G *is strongly bisimilar to* TS_Θ.
3. fRT_G *is isomorphic to* fRT_Θ.

[1] In particular, that if $G \xrightarrow{\Gamma} H$ and $G \xrightarrow{\Gamma} J$ then $H \equiv J$, which is easily re-stated in the at-expressions framework as well.

4. RT_G *isomorphic to* RT_Θ.

Proof. Follows from theorems 6.1 and 6.3. □

The relationships between the corresponding transition systems presented in this chapter together with links to the supporting definitions and theorems are depicted in the following graph



Chapter 7

Applications and Extensions

After obtaining the main consistency result, in this chapter, we will present several possible extensions of the newly proposed framework. Based on the existing proofs, these extensions will be able to cover different scenarios and applications and increase the modeling power of the timed-arc Petri Box Calculus.

7.1 Overview of possible extensions

In the original model, several assumptions have been made. These assumptions were imposed to somehow restrict the modeling power and make it easier to obtain the necessary equivalence results and proofs. On the other hand, these assumptions are not that restrictive since this type of timed-arc Petri nets was used in the past to model some interesting complex systems.

Starting the discussion about these assumptions, the time restrictions imposed on transition incoming arcs are supposed to be hard. Tokens that reached their maximum waiting time must either be used to fire the corresponding transition (if their corresponding transition is enabled) or they will become *dead* in the next time move (if their corresponding transition is not enabled). When a token is 'dead' for a transition means that it cannot be used to fire this transition anymore but it is possible to be consumed by another transition. Moreover, there was no consideration for some *clock reset* type of move. As a result, it is not possible to actually *revive* a dead token for a specific transition. The time is passing uniformly and the age of tokens can either increase with the help of a time move or be set to 0 for newly created tokens. Finally, time moves were meant to be global, meaning that by the execution of a time move, the age of all existing tokens will be increased by one time unit.

The possible extensions will be derived directly from the existing assumptions. To begin with, instead of hard time restrictions and urgent transitions, deadlines can also be soft. In this case, a time move will still be possible, even if the maximum waiting time for a token has been reached (and the corresponding transition is enabled) and it will lead to a disabled transition. Another interesting extension might be to introduce local clocks. Instead of having one global clock

and forcing the age of every available token to be increased when this clock ticks, there can be a number of different *locally-based clocks*. These locally-based clocks correspond and affect a specific place or most likely groups of places. In case a locally-based clock ticks, then the age of tokens belonging to the corresponding groups of places will be increased and the age of the remaining available tokens will remain unaffected. Yet another extension consists of the clock reset moves' introduction. Several types of such moves can be added. One may consider resetting the clock only when no other moves are possible. In conjunction with the soft deadlines rule, this resetting may give some new behaviours from previously disabled transitions due to the age of tokens. A different reseting move will reset clocks at any time without any special reason. Such a move is always possible but one has to treat this type of move with care since it is possible to disable several currently enabled transitions at once. The final reseting move is a very interesting one, especially when used in conjunction with local clocks. In this case, time resets happen after a specific number of ticks of the clock (either global or local). When local clocks are used, time resets in specific intervals for every locality.

In the following sections, we will present the operational semantics for some interesting combination of extensions.

7.2 Introduction of local clocks

This type of extension with locally-based clocks seems really interesting since this way it is possible to represent *Globally Asynchronous Locally Synchronous* systems, see e.g., [43]. Areas (places and transitions) that are affected by a specific locally-based clock are grouped together and they form a *synchronous block* SB. Each synchronous block can be considered to be an at-expression and several synchronous blocks are composed in parallel to describe the complete system. Moreover, in order to ensure communication between different synchronous blocks, some communication (synchronisation) actions are present in each block and it is possible to execute them via the scoping mechanism. Finally, a global clock that will affect the age of every token in the complete system may or may not be present. These new at-expressions will be called *lat-expressions*. For an example, we want to consider a GALS system with three synchronous blocks will look like the one in figure 7.1 and the (simplified) corresponding lat-expression will be

$$\begin{aligned} G = \ & \overline{(SB_1 \parallel SB_2 \parallel SB_3)\,\mathsf{sc}\,A}^{\mathbb{EL}} && \equiv \\ & \overline{SB_1}^{\mathbb{EL}} \parallel \overline{SB_2}^{\mathbb{EL}} \parallel \overline{SB_3}^{\mathbb{EL}}\,\mathsf{sc}\,A && \equiv \\ & \overline{SB_1}^{\mathbb{EL}_1} \parallel \overline{SB_2}^{\mathbb{EL}_2} \parallel \overline{SB_3}^{\mathbb{EL}_3}\,\mathsf{sc}\,A \end{aligned}$$

where A is the set of communication actions between synchronous blocks and $\mathbb{EL}_i \in \mathbb{D}$ is the age of the youngest and oldest token of synchronous block i that is being affected by the corresponding locally-based clock.

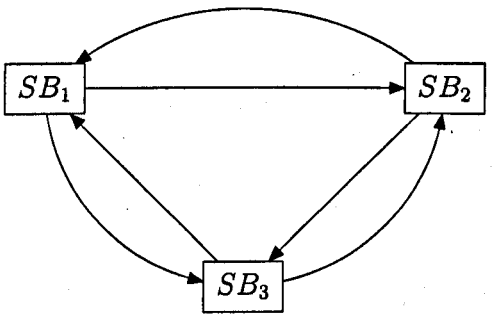


Figure 7.1: A simple representation of GALS system with three synchronous blocks.

Once again, we are using the same way to define operational semantics as in standard PBC but with the necessary modifications to address the timing restrictions. Since the lat-expressions are essentially a parallel composition of standard at-expressions with some scoping mechanism in place, $\equiv$ is the least equivalence relation on dynamic lat-expressions such that the rules in table 3.1 are satisfied.

SOS rules

In the case of lat-expressions three kinds of operational semantics moves are possible, namely action moves, *global-time* moves and *local-time* moves. A global-time move has the form

$$G \xrightarrow{\surd} H$$

a local time move has the form

$$SB_i \xrightarrow{\surd_i} SB_i'$$

where SB_i is the *ith* synchronous block and an action move has the form

$$G \xrightarrow{\Gamma} H$$

where Γ is a finite multiset of communication actions. We now define various types of moves of the structural operational semantics of dynamic at-expressions.

Empty moves

The following rules deal with the empty action moves.

$$\frac{G \equiv H}{G \xrightarrow{\varnothing} H} \qquad \frac{G \xrightarrow{\varnothing} J \xrightarrow{\Gamma} H}{G \xrightarrow{\Gamma} H} \qquad \frac{G \xrightarrow{\Gamma} J \xrightarrow{\varnothing} H}{G \xrightarrow{\Gamma} H}$$

Basic action

A basic action can occur if the timing restrictions of the synchronous block SB_i are satisfied by the age range of its overbar:

$$\frac{clock_i \text{ tsat } el_i}{\overline{\alpha el_i}^{\mathbb{EL}} \xrightarrow{\{\alpha\}} \underline{\alpha el_i}_{00}}$$

Note that the age range of a newly created underbar is always set to (00).

Scoping

There is a single rule for scoping:

$$\frac{G \xrightarrow{\{a_1,\widehat{a}_1\}+\cdots+\{a_k,\widehat{a}_k\}+\Gamma} H \ , \ (A \cup \widehat{A}) \cap \Gamma = \varnothing \ , \ a_1,\ldots,a_k \in A}{G \,\mathsf{sc}\, A \xrightarrow{k\cdot\{i\}+\Gamma} H \,\mathsf{sc}\, A}$$

Other operators

There is no real difference in the rules for the remaining operators when compared with the standard atPBC.

$$\frac{G \xrightarrow{\Gamma} G' \ , \ H \xrightarrow{\Gamma'} H'}{G\|H \xrightarrow{\Gamma+\Gamma'} G'\|H'} \qquad \frac{G \xrightarrow{\Gamma} H}{\begin{array}{l}\langle\langle G \circledast E \circledast F\rangle\rangle \xrightarrow{\Gamma} \langle\langle H \circledast E \circledast F\rangle\rangle \\ \langle\langle E \circledast G \circledast F\rangle\rangle \xrightarrow{\Gamma} \langle\langle E \circledast H \circledast F\rangle\rangle \\ \langle\langle E \circledast F \circledast G\rangle\rangle \xrightarrow{\Gamma} \langle\langle E \circledast F \circledast H\rangle\rangle\end{array}}$$

$$\frac{G \xrightarrow{\Gamma} H}{\begin{array}{l}E \square G \xrightarrow{\Gamma} E \square H \\ G \square E \xrightarrow{\Gamma} H \square E\end{array}} \qquad \frac{G \xrightarrow{\Gamma} H}{\begin{array}{l}G\,;E \xrightarrow{\Gamma} H\,;E \\ E\,;G \xrightarrow{\Gamma} E\,;H\end{array}}$$

At this point we need to decide whether soft or hard deadlines will be present in this model. This decision will affect the SOS rules for time moves together with the need to define urgent labels of lat-expressions. Both approaches will be presented in the following sections.

7.2.1 Time moves with soft deadlines

When soft deadlines are used in this model, there is no need to define urgency in the execution of actions. The age of tokens can increase without consideration of

the timing restrictions. The only consequence is that the transition corresponding to the exceeded maximum waiting time will become disabled. But even then a time move will still be possible.

Global-time moves

We have a time rule for global-time moves:

$$\frac{G \xrightarrow{\surd} G^{\surd}}{G \xrightarrow{\surd} H}$$

where $G^{\surd}$ is G with each time annotation $\mathbb{EL}$ at an over- or underbar changed to $(\mathbb{E}+1)(\mathbb{L}+1)$.

Local-time moves

There is a time rule for local-time moves:

$$\frac{SB_i \xrightarrow{\surd_i} SB_i^{\surd}}{G \xrightarrow{\surd_i} H}$$

where $SB_i^{\surd}$ is the synchronous block i with all of its time annotations $\mathbb{EL}$ at an over- or underbar changed to $(\mathbb{E}+1)(\mathbb{L}+1)$.

7.2.2 Time moves with hard deadlines

When hard deadlines are used in the model, if a token belonging to synchronous block i has reached its maximum waiting time for a specific enabled action then we have the following options. Time cannot increase anymore in this synchronous block by the execution of either a global or a local-time move to this SB. The corresponding token must either be used to execute this action, or it can be consumed by another action thus disabling the previous urgent action. Furthermore, it is obvious that local-time moves in synchronous blocks different from i are still possible since, these type of moves do not affect the age of tokens in SB_i and consequently there is no violation of the hard deadlines rules.

Urgent labels of lat-expressions

To identify cases when time moves can be applied, we need the notion of *urgent* labels which can be executed by a lat-expression and especially by its synchronous blocks. Urgent labels of dynamic lat-expressions are defined by

$$\mathsf{urgent}_{lab}(G) \stackrel{\text{df}}{=} \{\alpha \mid \alpha^0 \in \mathsf{enabled}_{aux}(G)\},$$

where $\mathsf{enabled}_{aux}(G)$ is a set defined by induction on the structure of G. This denotes that no further global-time moves are possible, but there may be some local-time moves still possible. Urgent labels of specific synchronous blocks of lat-expressions are defined by

$$\mathsf{urgent}_{lab}(SB_i) \stackrel{\mathrm{df}}{=} \{\alpha \mid \alpha^0 \in \mathsf{enabled}_{aux}(SB_i)\},$$

where $\mathsf{enabled}_{aux}(SB_i)$ is a set defined by induction on the structure of SB_i and this denotes that neither local-time moves nor of course global-time moves for the SB_i are possible. There are two kinds of objects which $\mathsf{enabled}_{aux}(G)$ can contain, namely α^δ and a, where $\alpha \in \mathcal{A} \cup \{\imath\}$, $a \in \mathcal{A}$ and $\delta \in \{0,1\}$. Intuitively, α^0 means that the label α is enabled and urgent in expression G, α^1 means that the label α is enabled but non-urgent, and a means that there is a pair of conjugate labels $(a, \widehat{a})$ enabled simultaneously and at least one of these labels is urgent. The same two kinds of objects are also contained into $\mathsf{enabled}_{aux}(SB_i)$. Similarly to the global-time case, α^0 means that the label α is enabled and urgent in synchronous block SB_i, α^1 means that the label α is enabled but non-urgent, and a means that there is a pair of conjugate labels $(a, \widehat{a})$ enabled simultaneously and at least one of these labels is urgent. This pair of conjugate labels $(a, \widehat{a})$ can be either in different synchronous block or in the same. Since the global-time case is contained in the local-time one, for the base case, we have:

$$\mathsf{enabled}_{aux}(\overline{\alpha el}^{\mathbb{EL}_i}) \stackrel{\mathrm{df}}{=} \begin{cases} \{\alpha^0\} & \text{if } \mathbb{EL}_i \text{ tsat } el \text{ and } l = \mathbb{L} \\ \{\alpha^1\} & \text{if } \mathbb{EL}_i \text{ tsat } el \text{ and } l > \mathbb{L} \\ \varnothing & \text{otherwise .} \end{cases}$$

$$\mathsf{enabled}_{aux}(\underline{\alpha el}_{\mathbb{EL}}) \stackrel{\mathrm{df}}{=} \varnothing$$

For more complicated expressions H, we define $\mathsf{enabled}_{aux}(H)$ as the smallest set such that, whenever $H \equiv G$ then

$$\mathsf{enabled}_{aux}(G) = \mathsf{enabled}_{aux}(H)$$

and then the following hold for individual cases of composition operators. For scoping, if $a \in \mathsf{enabled}_{aux}(G)$ and $a \in (A \cup \widehat{A})$ then:

$$\imath^0 \in \mathsf{enabled}_{aux}(G \,\mathsf{sc}\, A)\ ,$$

as well as

$$\{\alpha^\delta \in \mathsf{enabled}_{aux}(G) \mid \alpha \notin (A \cup \widehat{A})\} \subseteq \mathsf{enabled}_{aux}(G \,\mathsf{sc}\, A)$$
$$\{a \in \mathsf{enabled}_{aux}(G) \mid a \notin (A \cup \widehat{A})\} \subseteq \mathsf{enabled}_{aux}(G \,\mathsf{sc}\, A)\ .$$

For concurrent composition,

$$\mathsf{enabled}_{aux}(G) \cup \mathsf{enabled}_{aux}(J) \subseteq \mathsf{enabled}_{aux}(G \| J)$$
$$\{a \mid a^\delta \in \mathsf{enabled}_{aux}(G) \wedge \widehat{a}^{\delta'} \in \mathsf{enabled}_{aux}(J) \wedge \delta \cdot \delta' = 0\} \subseteq \mathsf{enabled}_{aux}(G \| J)\ .$$

For concurrent composition between different synchronous blocks,

$$\mathsf{enabled}_{aux}(SB_i) \cup \mathsf{enabled}_{aux}(SB_j) \subseteq \mathsf{enabled}_{aux}(SB_i \| SB_j)$$
$$\{a \mid a^{\delta} \in \mathsf{enabled}_{aux}(SB_i) \wedge \widehat{a}^{\delta'} \in \mathsf{enabled}_{aux}(SB_j) \wedge \delta \cdot \delta' = 0\} \subseteq$$
$$\mathsf{enabled}_{aux}(SB_i \| SB_j) .$$

For the remaining operators, we have that:

$$\begin{aligned}
\mathsf{enabled}_{aux}(G) \;\subseteq\; & \mathsf{enabled}_{aux}(\langle\!\langle G \circledast E \circledast F \rangle\!\rangle) \cap \mathsf{enabled}_{aux}(\langle\!\langle E \circledast G \circledast F \rangle\!\rangle) \\
& \cap \mathsf{enabled}_{aux}(\langle\!\langle E \circledast F \circledast G \rangle\!\rangle) \\
\mathsf{enabled}_{aux}(G) \;\subseteq\; & \mathsf{enabled}_{aux}(G \Box E) \cap \mathsf{enabled}_{aux}(E \Box G) \\
\mathsf{enabled}_{aux}(G) \;\subseteq\; & \mathsf{enabled}_{aux}(G ; E) \cap \mathsf{enabled}_{aux}(E ; G) .
\end{aligned}$$

Global-time moves

There exists a global-time rule:

$$\frac{\mathsf{urgent}_{lab}(G) = \varnothing}{G \xrightarrow{\surd} G^{\surd}}$$

where $G^{\surd}$ is G with each time annotation $\mathbb{EL}$ at an over- or underbar changed to $(\mathbb{E}+1)(\mathbb{L}+1)$.

Local-time moves

We have the following local-time move:

$$\frac{\mathsf{urgent}_{lab}(SB_i) = \varnothing}{SB_i \xrightarrow{\surd_i} SB_i^{\surd}}$$

where $SB_i^{\surd}$ is the synchronous block i with all of its time annotations $\mathbb{EL}$ at an over- or underbar changed to $(\mathbb{E}+1)(\mathbb{L}+1)$.

In both cases (soft or hard deadlines), both global and local time moves can only be applied at the topmost level of an expression as it cannot be 'propagated' through the expression using action rules. This ensures that time progresses uniformly. Finally it can be seen that the rules of operational semantics do not lead outside the set of dynamic lat-expressions.

7.3 Introduction of reset moves

The addition of reset moves is another possible extension to the existing model. Like mentioned before, in the overview section of this chapter, several types of reset moves are possible according to the modeler's needs.

7.3.1 Unconditional case

To begin with the most general case, it is possible to reset time (essentially resetting the age of available tokens to zero) at any given moment. No special conditions must be present for such move to occur and there are no restrictions to the number of possible repetitions of such move. Based on the original model, action moves do not increase the age of tokens (firing of transitions are supposed to be instant) and the only way to increase their age is by the occurrence of a time move either global or local. As a result, in order to affect the state of the system it only makes sense for an additional reset move to take place only after the execution of either a global or a local time move. Furthermore, when local clocks are present in the model, a reset move can either affect every available token (*global reset move*) or affect a specific locality of tokens that corresponds to a specific locally-based clock (*local reset move*). These new timed-arc expressions with reset moves will be called *rat-expressions*. Since the structure of the rat-expressions remains essentially the same as in the at-expressions model, $\equiv$ is again the least equivalence relation on dynamic rat-expressions such that the rules of structural equivalence in table 3.1 are satisfied. In the following sections, in order to avoid as much as possible repetitions of the SOS rules from previous sections, we only present the new different rules together with rules necessary for the readability of chapter.

SOS rules

In the case of rat-expressions where global and local clocks are present, five different operational semantics moves are possible, namely action moves, global-time moves, local-time moves, global-reset and local-reset moves. It is obvious that when no local clocks are present in the expressions, we cannot have local-time and local-reset moves. Like before, a global-time move has the form

$$G \xrightarrow{\surd} H$$

a local time move has the form

$$SB_i \xrightarrow{\surd_i} SB_i'$$

where SB_i is the *ith* synchronous block and an action move has the form

$$G \xrightarrow{\Gamma} H$$

where Γ is a finite multiset of communication actions. The additions to these three moves are the two reset moves. A global-reset move has the form

$$\frac{G \xrightarrow{\surd} G^{\surd}}{G \xrightarrow{\surd} H}$$

where $G^{\surd}$ is G with every available tokens' age reset to 0 and as a consequence each time annotation $\mathbb{EL}$ at an over- or underbar changed to 00. Finally, a local-reset move has the form

$$\frac{SB_i \xrightarrow{\surd_i} SB_i^{\surd}}{G \xrightarrow{\surd_i} H}$$

where $SB_i^{\surd}$ is the synchronous block i with the age of every corresponding token reset to 0 and as a consequence the time annotation $\mathbb{EL}_i$ at an over- or underbar changed to 00.

The SOS rules for both local and global time moves are essentially the same as in lat-expressions and thus we avoid repeating them. Some discussion is necessary about the 'critical' case where a transition has reached its maximum waiting time. In both soft and hard deadline cases, the transition can fire before the occurrence of a time move. When soft deadlines are used in this model, the age of tokens can increase without consideration of the timing restrictions and as a result a time move is always possible. Once again the consequence is that the transition corresponding to the exceeded maximum waiting time will become disabled, but even then a time move will still be possible. This disabling might be only temporary in this case since a reset move is also always possible. If the necessary tokens are still available in the preset of the disabled transition, a reset move and a number of time moves will make the transition active once more. When hard deadlines are used, a time move (global or local to the corresponding synchronous block) is possible until an enabled transition becomes urgent. At this point we have the following options. Either the urgent transition must fire consuming every corresponding token or the urgent transition must become disabled by the firing of another transition or we can have a reset move. This reset move may disable the transition if the minimum waiting time is greater than zero. Furthermore, a reset move at this stage may not allow further time moves if the maximum waiting time is zero. To depict these possibilities, let us consider the following two evolution scenarios of a rat-expression:

$$\overline{a22}^{00} \xrightarrow{\surd\surd} \overline{a22}^{22} \xrightarrow{a} \underline{a22}_{00}$$

or

$$\overline{a22}^{00} \xrightarrow{\surd\surd} \overline{a22}^{22} \xrightarrow{\surd} \overline{a22}^{00} \xrightarrow{\surd\surd} \overline{a22}^{22} \xrightarrow{a} \underline{a22}_{00}$$

The only available move at the initial state is a time move. Actually, two time moves are necessary in order to reach the minimum waiting time. At this point, no further time move is possible and we have the following options. Either action a can fire and the expression will reach its final state or a reset move can occur. The age of token(s) will reset to zero and now action a cannot fire since the minimum waiting time has not been reached. After two time moves, action is enabled and urgent once again. Now, let us consider a slightly different

expression $\overline{a00}^{00}$. In this case, a reset move is without meaning since it cannot allow any additional time moves.

7.3.2 Controlled reset moves

In this section, we will investigate reset moves that are somehow used in a controlled manner. By controlled manner we mean that this moves cannot occur at any time but only when certain conditions are satisfied. For example, we may include a reset move that becomes activated only when no other action moves are possible. This type of reset moves are interesting when soft deadlines are used. In this situation, dead tokens can be present in the system and a system that is deadlocked because of time restrictions will be able to make some additional moves, since some of its transitions may still be enabled in the usual Petri nets way. This type of reset move not only occurs in situations where a system is deadlocked because of time restrictions. There are cases where a system is not deadlocked but only time moves are possible, for example $\overline{a22}^{00}$. Before action a becomes ready to fire, two time moves must be executed. A reset move in this case will cause a delay in the firing of a.

In another case, we may want to model resetting the age of tokens in a system after a specific number of time moves.

SOS Rules

Similar to the unconditional case, when global and local clocks are present, there are five possible operational semantics moves, namely action moves, global-time moves, local-time moves, global-reset and local-reset moves. It is obvious that when no local clocks are present in the expressions, we cannot have local-time and local-reset moves. Action, global-time and local-time moves are the same as in the unconditional case and we present only the two different reset moves. A global-reset move has the following form

$$\frac{\neg\exists\ G \xrightarrow{\Gamma} H}{G \xrightarrow{\Upsilon} G^{\Upsilon}}$$

where G^{Υ} is G each time annotation $\mathbb{EL}$ at an over- or underbar changed to 00. Finally, when local clocks are used a local-reset move is possible when no action moves are possible in the corresponding synchronous block. A local-reset move has the form

$$\frac{\neg\exists\ SB_i \xrightarrow{\Gamma} SB_i'}{SB_i \xrightarrow{\Upsilon_i} SB_i^{\Upsilon}}$$

where SB_i^{Υ} is the synchronous block i the time annotation $\mathbb{EL}_i$ at an over- or underbar changed to 00.

Example

In the example in figure 7.2, we have one evolution of the same at-expression as the one in figure 3.2. The difference is that we now allow reset moves when no other moves are possible. In line (5), action a is urgent, but its counterpart $\widehat{a}$ is not enabled yet due to the time restrictions. As a result, the synchronisation action of the scoping operator is not possible and there are no other possible action moves after that. A global-reset move is now possible in line (6) and after the passage of one time unit in line (7), the synchronisation action is now possible and the expression reaches its terminal state.

$$
\begin{array}{lll}
(1) & \overline{(a11 \parallel (b11\,;\widehat{a}11))\,\mathsf{sc}\{a\}}^{00} & \equiv \\
(2) & (\overline{a11}^{00} \parallel (\overline{b11}^{00}\,;\widehat{a}11))\,\mathsf{sc}\{a\} & \xrightarrow{\surd} \\
(3) & (\overline{a11}^{11} \parallel (\overline{b11}^{11}\,;\widehat{a}11))\,\mathsf{sc}\{a\} & \xrightarrow{\{b\}} \\
(4) & (\overline{a11}^{11} \parallel (\underline{b11}_{00}\,;\widehat{a}11))\,\mathsf{sc}\{a\} & \equiv \\
(5) & (\overline{a11}^{11} \parallel (b11\,;\overline{\widehat{a}11}^{00}))\,\mathsf{sc}\{a\} & \xrightarrow{\Upsilon} \\
(6) & (\overline{a11}^{00} \parallel (b11\,;\overline{\widehat{a}11}^{00}))\,\mathsf{sc}\{a\} & \xrightarrow{\surd} \\
(7) & (\overline{a11}^{11} \parallel (b11\,;\overline{\widehat{a}11}^{11}))\,\mathsf{sc}\{a\} & \xrightarrow{\{\imath\}} \\
(8) & (\underline{a11}_{00} \parallel (b11\,;\underline{\widehat{a}11}_{00}))\,\mathsf{sc}\{a\} & \equiv \\
(9) & (\underline{a11}_{00} \parallel \underline{(b11\,;\widehat{a}11}_{00}))\,\mathsf{sc}\{a\} & \equiv \\
(10) & \underline{(a11 \parallel (b11\,;\widehat{a}11))\,\mathsf{sc}\{a\}}_{00} &
\end{array}
$$

Figure 7.2: An evolution of the expression $\overline{(a00 \parallel (b11\,;\widehat{a}01))\,\mathsf{sc}\{a\}}^{00}$.

Conclusions

In this thesis, we proposed a new framework to model concurrent computations. To be more precise, we managed to provide an extension of framework presented in [9] to support timing restrictions on the resources of a concurrent system. This framework introduced a new compositional model of timed-arc Petri nets, and a corresponding process algebra of time expressions. This type of time restricted Petri nets and the process algebra co-exist and we managed to establish the existence of a strong relationship between them.

In chapter 3 we presented the syntax for the new algebra of process expressions which is based on the syntax of standard PBC. Furthermore, we defined the label based operational semantics of these process expressions. In chapter 4, we extended the algebra of expressions to an algebra of nets by the compositional definition of mapping from at-expressions to at-boxes. Additionally, we defined transition based operational semantics for the at-expressions since this type of semantics were necessary for the proof of the main results. Finally, we identified interface regions in at-boxes. The monotonic behaviour of these sets of places cleared our perception about the evolution of composite nets.

In chapter 5, we have explained the nature of the correspondence between the two newly created algebras, in terms of their respective reachability trees. We highlighted arguments showing that, in general, there can be no direct translation from dynamic at-expressions to at-boxes since, informally, there are fewer of the former than of the latter. Consequently, our main result showing behavioural relations between at-expressions and at-boxes could not be obtained by a simple adaptation of that used in standard PBC since dynamic at-expressions cannot be unambiguously mapped to at-boxes. In order to prove the correspondence between at-expressions and at-boxes and take advantage of existing strong results from standard PBC, an intermediate cluster-based representation was introduced. Following the same pattern as before, we extend the algebra of at-expressions to cat-boxes. The main result of this thesis is presented in chapter 6 and shows the necessary strong behavioural relationship between at-expressions and at-boxes. Since we used the intermediate cluster-based representation, a couple of secondary results were required, in order to obtain the necessary proof for the main result. These results revealed a strong behavioural relationship between at-boxes and cat-boxes, together with isomorphism between at-expressions and cat-boxes. Finally, in chapter 7 several possible extensions were presented in order to increase the modeling power of the framework and address further

modeling needs.

Future work

In this thesis it was possible to obtain all necessary results to support the newly presented framework, but we also managed to reveal several new directions for further research. The extensions presented in chapter 7 although they come directly from the presented theory are not thoroughly explored. For the support of these extensions, a comprehensive investigation is necessary together with the development of complete proofs. Furthermore, several new extensions can be considered, i.e., nested time cases. On a different front, the obtained results made it possible to combine the verification techniques developed independently for process algebra and Petri nets with timing, and to give a syntax oriented semantics of real-time specification languages. At this point, it must be clear that at-expressions are more *abstract* than the corresponding at-boxes. This, as we expect, can be used to improve model-checking of behaviours specified by at-expressions, by providing an equivalence relation between reachable token timings of at-boxes which could be used to improve the efficiency of the unfolding of at-boxes (with the resulting unfolding being smaller). The development of model checking algorithms and the corresponding tool support tailored to the presented model is currently under investigation. The basis for this investigation is the general scheme for generating net unfoldings presented in [35, 36].

Furthermore as part of the future research ideas, someone may want to consider different translations from the algebra of at-expression to some different formalisms that can handle time restrictions. This way it will be possible to take advantage of existing model-checking tools for these formalisms. One such example is timed automata [2, 3]. This is a well known formalism used for the analysis of systems with timing information and has been extensively studied in the past. A timed automaton is an ω-automaton coupled with a finite set of clocks recording the passage of time. Moreover, any transition of the automaton can reset these clocks (which is similar to the model presented here) and the timing constraints in this model are expressed by comparing clock values with time constants found in transition enabling conditions. Essentially actions are composed in parallel with different time annotations. The reasons behind our choice of timed-arc Petri nets instead of a translation to timed automata are that the structure of Petri nets is richer, the timing constraints are easier to handle in Petri nets, and we are also losing the ability to represent concurrency explicitly.

Bibliography

[1] P. A. Abdulla and A. Nylén: Timed Petri Nets and BQOs. Proc. of *ICATPN'01*, J.-M. Colom, M. Koutny (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2075 (2001) 53-70.

[2] R. Alur and D. L. Dill: A theory of timed automata. *Theoretical Computer Science* 126 (1994) 183-235.

[3] R. Alur and D. L. Dill: Automata For Modeling Real-Time Systems. Proc. of *ICALP '90: Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, M. S. Paterson, G. Goos, J. Hartmanis (Eds.). Springer-Verlag, Lecture Notes in Computer Science 443 (1990) 322–335.

[4] J. Baeten and W. P. Weijland: *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press (1990).

[5] B. Berthomieu and M. Diaz: Modelling and verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. on Soft. Eng.* 17 (1991) 259-273.

[6] E. Best and R. Devillers: Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* 55 (1988) 87-136.

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

[8] E. Best, R. Devillers and M. Koutny: A Unified Model for Nets and Process Algebras. In: *Handbook of Process Algebra*, J. A. Bergstra, A. Ponse, S. A. Smolka, (Eds.). Elsevier (2001) 873–944.

[9] E. Best, R. Devillers and M. Koutny: *Petri Net Algebra.* EATCS Monographs on TCS, Springer (2001).

[10] E. Best, R. Devillers and M. Koutny: The box algebra = Petri nets + process expressions. *Information and Computation* 178 (2002) 44–100.

[11] B. Bieber and H. Fleischhack: Model Checking of Time Petri Nets Based on Partial Order Semantics. Proc. of *CONCUR'99*, J. Baeten and S. Mauw (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1664 (1999) 210-225.

[12] E. Best, H. Fleischhack, W. Fraczak, R. Hopkins, H. Klaudel, and E. Pelz: A Class of Composable High Level Petri Nets. Proc. of *Application and Theory of Petri Nets (ATPN'1995)*, G. DeMichelis and M. Diaz (Eds.). Springer-Verlag, Lecture Notes in Computer Science 935 (1995) 103–120.

[13] E. Best, H. Fleischhack, W. Fraczak, R. Hopkins, H. Klaudel, and E. Pelz: An M-net Semantics of $B(PN)^2$. Proc. of *International Workshop on Structures in Concurrency Theory (STRICT'1995)*, J. Desel (Ed.). Berlin (1995) 85–100.

[14] T. Bolognesi and P. Cremonese: The weakness of some timed models for concurrent systems. Technical Report, CNUCE C89-29, CNUCEC.N.R. (1989).

[15] T. Bolognesi, F. Lucidi and S. Trigila: From Timed Petri Nets to timed LOTOS. Proc. of *PSTV'90*, L. Logrippo et al. (Eds.). North-Holland (1990) 395-408.

[16] F. D. J. Bowden: Modelling Time in Petri Nets. Proc. of *the second Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management, Gold Coast, Australia*, (1996) .

[17] G. W. Brams (nom collectif de Ch. André, G. Berthelot, C. Girault, G. Memmi, G. Roucairol, J. Sifakis, R. Valette, G. Vidal-Naquet): *Résaux de Petri: Théorie et Pratique.* Two volumes. Editions Masson (1985).

[18] I. Castellani: Process Algebra with Locations. Chapter 5.3 in this issue.

[19] E. M. Clarke, O. Grumberg, and D. Peled: *Model Checking.* MIT Press (1999).

[20] J. E. Coolahan Jr. and N. Roussopoulos: Timing Requirements for Time-Driven Systems Using Augmented Petri Nets.. *IEEE Trans. Software Eng.* 9.5 (1983) 603-616.

[21] H. Fleischhack, B. Grahlmann: A Petri Net Semantics for $B(PN)^2$ with Procedures which Allows Verification. Technical Report, 21, Universität Hildesheim (1996).

[22] H. Fleischhack, B. Grahlmann: A Petri Net Semantics for $B(PN)^2$ with Procedures. Proc. of *2nd International Workshop on Software Engineering for Parallel and Distributed Systems (PDSE'1997)*, IEEE Computer Society Press (1997) 15–27.

[23] D. de Frutos-Escrig and V. Valero Ruiz and O. Marroquín Alonso: Decidability of Properties of Timed-Arc Petri Nets.. Proc. of *ICATPN'00*, Mogens Nielsen and Dan Simpson (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1825 (2000) 187-206.

[24] H. J. Genrich, K. Lautenbach and P. S. Thiagarajan: Elements of General Net Theory. In: *Net Theory and Applications, Proc. of the Advanced Course on General Net Theory of Processes and Systems*, W. Brauer (Ed.). Springer-Verlag, Lecture Notes in Computer Science 84 (1980) 21-163.

[25] U. Goltz: On Representing CCS Programs by Finite Petri Nets. Proc. of *MFCS'88*, M. P. Chytil, L. Janiga and V. Koubek (Eds.). Springer-Verlag, Lecture Notes in Computer Science 324 (1988) 339-350.

[26] U. Goltz and R. Loogen: A Non-interleaving Semantic Model for Nondeterministic Concurrent Processes. *Fundamentae Informaticae* 14 (1991) 39-73.

[27] U. Goltz and A. Mycroft: On the Relationship of CCS and Petri Nets. Proc. of *ICALP 84, International Conference on Automata, Languages and Programming*, J. Paredaens (Ed.). Springer-Verlag, Lecture Notes in Computer Science 172 (1984) 196-208.

[28] U. Goltz and W. Reisig: The Non-sequential Behaviour of Petri Nets. *Information and Control* 57 (1983) 125-147.

[29] J. Goubault-Larrecq and I. MacKie: *Proof Theory and Automated Deduction.* Kluwer Academic Publishers (1997).

[30] J. Grabowski: On Partial Languages. *Fundamentae Informaticae* 4 (1981) 427-498.

[31] H.-M. Hanisch: Analysis of Place/Transition Nets with Timed Arcs and its Application to Batch Process Control. Proc. of *the 14th International Conference on Application and Theory of Petri Nets*, Springer-Verlag (1993) 282–299.

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

[33] ISO/IEC 13818-2 Draft International Standard Generic Coding of Moving Pictures and Associated Audio. Recommendation H.262.

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

[35] V. Khomenko: *Model Checking Based on Prefixes of Petri Net Unfoldings.* School of Computing Science, University of Newcastle upon Tyne (2003).

[36] V. Khomenko, M. Koutny and W. Vogler: Canonical Prefixes of Petri Net Unfoldings. *Acta Informatica* 40 (2003) 95-118.

[37] M. Koutny: A Compositional Model of Time Petri Nets. Proc. of *ICATPN'00*, M. Nielsen and D. Simpson (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1825 (2000) 303-322.

[38] M. Koutny and E. Best: Fundamental Study: Operational and Denotational Semantics for the Box Algebra. *Theoretical Computer Science* 211 (1999) 1–83.

[39] O. Marroquin Alonso and D. de Frutos-Escrig: Extending the Petri Box Calculus with Time. Proc. of *ICATPN'01*, J.-M. Colom and M. Koutny (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2075 (2001) 303-322.

[40] O. Marroquin Alonso and D. de Frutos-Escrig: Time-outs and Delays in TPBC. Technical Report, Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid (2003).

[41] O. Marroquin Alonso and D. de Frutos-Escrig: Urgency in TPBC. Technical Report, Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid (2002).

[42] A. Mazurkiewicz: Trace Theory. In: *Advances in Petri Nets1986, Petri Nets: Applications and Relationships to Other Models of Concurrency, Part II*, W. Brauer, W. Reisig and G. Rozenberg (Eds.). Springer-Verlag, Lecture Notes in Computer Science 255 (1987) 279-324.

[43] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Öberg, D. Lindqvist, H. Tenhunen and A. Postula: Evaluating benefits of Globally Asynchronous Locally Synchronous VLSI architecture. Proc. of *16th Norchip*, (1998) 50-57.

[44] P. Merlin and D. Farber: Recoverability of Communication Protocols - Implication of a Theoretical Study. *IEEE Trans. on Soft. Comm.* 24 (1976) 1036-1043.

[45] R. Milner: *A Calculus of Communicating Systems.* Springer-Verlag, Lecture Notes in Computer Science 92 (1980).

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

[47] T. Murata: Petri Nets: Properties, Analysis and Applications. *Proc. of IEEE* 77 (1989) 541-580.

[48] A. Niaouris: An Algebra of Petri Nets with Arc-Based Time Restrictions. Proc. of *ICTAC'04*, Z. Liu and K. Araki (Eds.). Springer-Verlag, Lecture Notes in Computer Science 3407 (2005) 447-462.

[49] A. Niaouris, M. Koutny: An Algebra of Timed-Arc Petri Nets. Technical Report, School of Computing Science, University of Newcastle (Mar 2005).

[50] M. Nielsen, V. Sassone and J. Srba: Towards a Notion of Distributed Time for Petri Nets. Proc. of *the 22nd international Conference on Application and theory of Petri Nets*, J. M. Colom and M. Koutny (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2075 (2001) 23-31.

[51] M. Nielsen, V. Sassone and J. Srba: Properties of Distributed Timed-Arc Petri Nets. Proc. of *FSTTCS'01*, R. Hariharan, M. Mukund, V. Vinay (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2245 (2001) 280-285.

[52] J. Jose Pardo, V. Valero, F. Cuartero, D. Cazorla: Automatic Translation of a Timed Process Algebra into Dynamic State Graphs. Proc. of *Eighth Asia-Pacific Software Engineering Conference*, IEEE Computer Society Press (2001) p. 63.

[53] F. L. Pelayo, F. Cuartero, V. Valero, H. Macia, M. L. Pelayo: Applying Timed-Arc Petri Nets to improve the performance of the MPEG-2 Encoding Algorithm. Proc. of *10th International Multimedia Modelling Conference*, IEEE Computer Society Press (2004) 49-56.

[54] G. D. Plotkin: A Structural Approach to Operational Semantics. Technical Report, FN-19, Computer Science Department, University of Aarhus (1981).

[55] L. Popova: Time Petri Nets. *Journal of Information Processing and Cybernetics EIK* 27 (1991) 227-244.

[56] C. Ramchandani: *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets.* Cambridge, Mass.: MIT, Dept. Electrical Engineering, PhD Thesis (1974).

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

[58] A. W. Roscoe: *The Theory and practice of Concurrency.* Prentice-Hall (1998).

[59] V. V. Ruiz and D. Escrig and F. Cuartero: On Non-Decidability of Reachability for Timed-Arc Petri Nets. Proc. of *8th. International Workshop on Petri Nets and Performance Models*, IEEE Computer Society Press (1999) 188-196.

[60] V. V. Ruiz, J. J. Pardo and F. Cuartero: Translating TPAL Specifications into Timed-Arc Petri Nets. Proc. of *of the 23rd international Conference*

on Applications and theory of Petri Nets, J. Esparza and C. Lakos (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2360 (2002) 414-433.

[61] V. V. Ruiz, F. L. Pelayo, F. Cuartero and D. Cazorla: On the Improvements of the MPEG-2 Encoding Algorithm by Timed-Arc Petri Nets. Proc. of *18th Annual UK Performance Engineering Workshop (UKPEW'02)*, University of Glasgow (2002) 211-221.

[62] V. V. Ruiz, F. L. Pelayo, F. Cuartero and D. Cazorla: Specification and Analysis of the MPEG–2 Video Encoder with Timed-Arc Petri Nets. *Electr. Notes Theor. Comput. Sci.* 6, no.2 (2002) .

[63] J. Sifakis: Use of Petri Nets for Performance Evaluation. Proc. of *the Third International Symposium IFIP W.G. 7.3., on Measuring, Modelling and Evaluating Computer Systems*, Elsevier Science Publ. (1977) 75–93.

[64] P. H. Starke: Processes in Petri Nets. *Elektronische Informationsverarbeitung und Kybernetik* 17 (1981) 389-416.

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

[66] W. Vogler: Partial Words versus Processes: a Short Comparison. In: *Advances in Petri Nets1992*, G. Rozenberg (Ed.). Springer-Verlag, Lecture Notes in Computer Science 609 (1992) 292-303.

[67] B. Walter: Timed Petri Nets for Modelling and Analyzing Protocols with Real-time Characteristics. *Protocol Specification, Testing and Verification III* (1983) Elsevier Science Publ. B.V. (North Holland) 149-159

[68] J. Wang: *Timed Petri Nets: Theory and Application.* Kluwer Academic Publishers (1998).

Index