Compositional Verification of Quantitative Properties of Statecharts

Francesca Levi
Dipartimento di Informatica, Universitá di Pisa
levifran@di.unipi.it

September 20, 2000

Abstract

In this paper we propose a process language $\mathcal{TSP}$ which abstractly models timed statecharts with minimal and maximal delays associated to transitions. Statecharts processes are equipped with a labelled transition system semantics that combines the basic principles of the semantics of Pnueli and Shalev with discrete time. Furthermore, we propose a compositional proof system to check quantitative temporal properties of statecharts processes. Properties are expressed in a discrete extension of μ-calculus with reset over clocks and clock constraints. The proof system is sound in general and it is complete for the class of regular processes (including processes corresponding to statecharts).

Key words: Timed Statecharts, Compositional Verification, Propositional μ-calculus, Local Model Checking.

1 Introduction

Reactive systems [41] maintain a continuous interaction with the environment in contrast to classical transformational systems which compute a result given an input. Typical examples of reactive systems are operating systems, flight reservation systems and embedded systems. Among the variety of formalisms to specify and program reactive systems *Statecharts* [18] is a visual formalism obtained by extending classical state-transition diagrams with notions of *parallelism*, *hierarchy* and *broadcast communication*. These features allow to develop high-level specifications, even of complex reactive systems, by parallel composition and refinement of simple statecharts. Since these applications are usually very complex formal verification of their correctness is an essential task. Temporal logics [15] are undoubtedly the most common formalism to express correctness properties of reactive systems. A formal and clear semantics is mandatory both for automatic (model checking) and proof theoretic verification of temporal properties.

Statecharts, similarly to other languages [11, 12, 37], is based on the *synchrony hypothesis* [11]: the program reacts to input events instantaneously by producing output events. This hypothesis relies on two reasonable assumptions: (1) the environment can be modelled as a discrete process, (2) the system is infinitely faster than the environment. Thanks to synchrony the behaviour of statecharts can be abstractly described by the sequence of instantaneous reactions (steps) produced by external events. Unfortunately, the definition of step is quite controversial, since the synchrony hypothesis produces some paradoxal and unexpected situations. Several very different notions of step have been proposed in the last few years [20, 42, 35, 36] (see [48, 25] for a survey on statecharts variants and semantics). Most of the semantics agree on a basic causality principle that allows to rule out grave situations such as self-causing transitions. Other problems arise whenever transitions may depend on the presence of events as well as on the absence of signals. Negated events may produce well-known causal paradoxes, such as transitions disallowing their own causes, which lead to different approaches.

In this paper we address a difficult problem: compositional verification of real-time properties for a timed variant of statecharts. The idea of compositional verification is that of reducing a verification task for a compound system to verification tasks for its components. Compositionality is essential to suitably support stepwise development of complex systems. A compositional approach allows, whenever a system component is eventually refined or modified, to check only the parts involved instead of the whole system, as well as allows to verify high-level descriptions with undefined parts. Compositional verification may also be used in the model checking approach to reduce the well-known state explosion problem, since the size of the systems, which have to be automatically verified, can be reduced.

We consider a timed version of the basic statecharts paradigm [18], where analogously to [27], a (discrete) interval of time is assigned to transitions. The model differs from the classical one since the enabling of transitions depends both on input events and on the time elapsed. A transition is enabled whenever the source box has been continously entered for a time between the minimal and maximal delay specified by the interval. As far as concern the notion of step we consider a semantics in the style of [42], where synchrony hypothesis, causality principle and global consistency are combined together. This semantics has, in our opinion, a suitable treatment of causal paradoxes due to negated events. A causal paradox produces an undefined reaction, that is a run-time error.

The work consists of some main steps:

1. the definition of a statecharts process algebra $\mathcal{TSP}$;

2. the definition of a LTS semantics of $\mathcal{TSP}$ equivalent to the timed statecharts one;

3. the definition of a compositional proof system to check real-time properties of timed statecharts processes.

The idea of a statecharts process algebra is not new [47, 46, 30, 33]; a process algebra gives actually a more abstract representation of statecharts for which a LTS semantics can be more simply defined and more standard verification methods can be applied. In [30, 33] a process algebra and a LTS semantics equivalent to the Pnueli and Shalev one for the original statecharts model have been proposed. The language is characterized by classical operators of prefixing, choice, recursion, (synchronous) parallel composition and by a new operator of refinement. Refinement is essential for representing the statecharts hierarchy. The LTS semantics is quite complex compared to other process algebras, since as discussed [25] the statecharts semantics satisfying the causality principle and the synchrony hypothesis cannot modular. As a consequence a step of a complex statechart cannot be derived by observing only the input-output behaviour of its components. More information concerning the internal behaviour of components is needed; in particular the causality relation between inputs and outputs and the kind of the step, either idle or non-idle. The complexity of the problem is demonstrated by the proposal of [46] which is not equivalent to the Pnueli and Shalev semantics since some necessary information was not properly captured.

In this paper, we show that this approach can be extended to timed statecharts by introducing a proper timed process algebra $\mathcal{TSP}$. The main novelties of $\mathcal{TSP}$ are timed actions and explicit locations in processes. Timed actions reflect the statecharts transitions depending on time, while locations correspond to statechart boxes. Explicit locations are very useful to simply evaluate the enabling of transitions with respect to time. By considering discrete clocks corresponding to locations the enabling of timed actions does not introduce additional complexity in the LTS construction.

The compositional LTS is the basis to develop a compositional proof system for checking real-time properties of $\mathcal{TSP}$ processes. In order to express real-time requirements we adopt a discrete real-time extension of propositional μ-calculus [43, 28] (RTL_μ) with reset over clocks [22] and clock constraints. The logic is powerful enough to express the typical real-time properties [5, 4] of bounded response and bounded invariance involving the quantitative as well the qualitative aspect of time.

The proof system is defined along the lines of [9] which combines the tagged fix-point method [49] with compositionality to establish μ-calculus properties of a simple process algebra. The approach relies on the combination of two types of rules:

1. logical rules that work on the structure of the formula;

2. compositional rules for the next modalities that work on the structure of the process and reduce a satisfaction problem for a compound process to satisfaction problems for its components. Special formulas (*extended assertions*) are used to eventually record the residual process arising by a decomposition.

A similar proof system for the un-timed version of $\mathcal{TSP}$ has been proposed in [33] by adapting the simple compositional rules to handle the complex process constructors typical of statecharts processes. In particular, the refinement operator and the synchronous parallel composition require special care to be handled. Logical rules have not to be modified for $\mathcal{TSP}$, while both the tagged fix-point method and the compositional rules have to be substantially modified. The underlying transition system is infinite due to time so that the straightforward extension of these techniques would be not effective. We show that by considering proofs statements parametrized to clock constraints both the tagged fix-points rules and the compositional rules can be properly adapted. Since clock constraints allow to implicitly represent set of timed states (also infinite), the proof system turns out to be correct and complete for a class of processes (including the ones corresponding to statecharts).

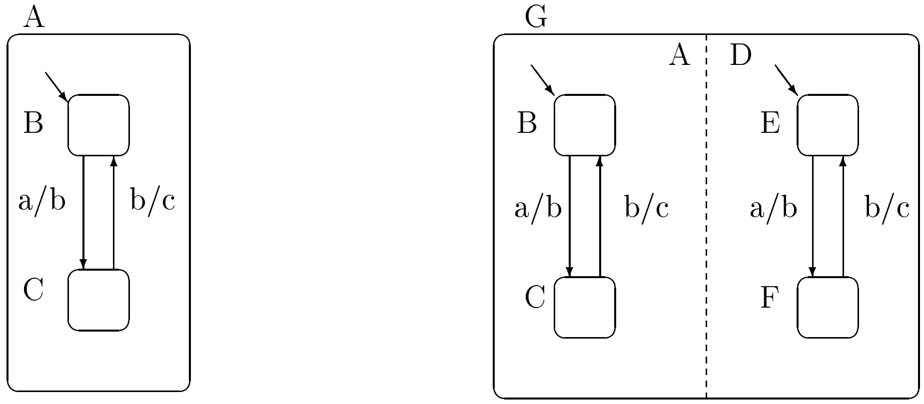

Figure 1: Statecharts

The paper is organized as follows. Section 2 presents timed statecharts and section 3 presents their semantics. Section 4 presents $\mathcal{TSP}$ and section 5 presents the labelled transition system semantics. The logic RTL_μ is introduced in section 6. The main ideas of the proof system are shown in section 7, while the complete proof rules set is shown in section 8. Correctness and completeness are given in section 9. A verification example is shown in section 10.

2 Timed Statecharts Syntax

Statecharts [18] are based on the idea of extending classical state-transition diagrams with notions of parallelism, hierarchy and broadcast communication. Basic statecharts are essentially flat state-transition diagrams with transitions depending on input events and producing output events. Statecharts can be composed in parallel and refined so that a hierarchical structure with concurrency is obtained. Figure 1 shows a basic statechart A with boxes B and C. Transitions are labelled by a pair *trigger/action*. The *trigger* contains events and negated events, which specify the enabling condition of transitions. In the trigger a condition e requires the presence of event e, while a negative condition $\bar{e}$ requires e to be absent. On the contrary, the *action* contains the events which are produced by transition execution. The box B pointed by the arrow without source is the unique default (initial) state. Statechart G shows the parallel composition of statecharts A and D which allows parallel execution of transitions and communication of events between components A and D. The hierarchical structure is shown by statechart H of figure 2. The execution of a high-level transition such as the one from A to D causes all the internal activities of statechart A to be interrupted.

Several variants of the original model of statecharts have been proposed [48]. In this paper we consider a version of statecharts where transitions depend in addition on time. As in [27, 39] timing constraints are specified by minimal and maximal delays on transitions. A transition is enabled only whenever the source state has been continuously entered for a time between the minimal and the maximal delay. In order to more properly define a compositional semantics more complex features such

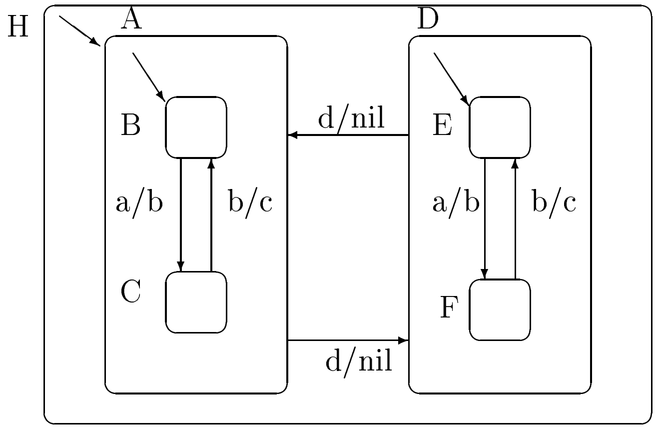

Figure 2: Hierarchy

as inter-level transitions and history mechanisms are ruled out.

A statechart is formally defined over a finite set Π of primitive events. Events are used both for internal and external communication. We assume the null event $nil \in \Pi$. We denote by $\bar{\Pi} = \{\bar{e} \mid e \in \Pi\}$, where $\bar{e}$ stands for the negation of event e and $\bar{\bar{e}}$ is event e for $\bar{e} \in \bar{\Pi}$. For $I \subseteq \Pi \cup \bar{\Pi}$, we say that I is consistent if it is not the case that $e, \bar{e} \in I$ for some $e \in \Pi$. The set of transition labels is given by $Lab(\Pi) = \{I/O \mid I \in 2^{\Pi \cup \bar{\Pi}}, O \in 2^{\Pi}$ and such that I is consistent$\}$.

In the following, we use the notation $i \in \{1, k\}$ for $i \in \{1, \ldots, k\}$. Moreover, we write singleton sets without brackets.

Definition 1 Let Π be a set of primitive events and Loc be a set of boxes.

- $S : (\{S_1, \ldots, S_k\}, T, in, out, \chi, \tau, S_i)$ is a *basic timed statechart* S where
  - $\{S_1, \ldots, S_k,\} \cup \{S\} \subseteq Loc$, $S \neq S_i$ and $S_i \neq S_j$ for each $i, j \in \{1, k\}$ with $i \neq j$;
  - T is a set of transitions;
  - in, $out : T \to \{S_1, \ldots S_k\}$ give the target and source states of transitions, respectively;
  - $\chi : T \to Lab(\Pi)$ is the labelling function of transitions;
  - $\tau : T \to (N \times (N \cup \{\infty\}))$ is the timing labelling function of transitions;
  - S_i such that $i \in \{1, k\}$ is the default state.

  Let $boxes(S) = \{S\} \cup \{S_1, \ldots, S_k\}$.

- $\mathrm{OR}(S, \{S_1, \ldots, S_k\})$ is a *refined timed statechart*, where
  - S is a basic statechart over Π and Loc such that $boxes(S) = \{S, S_1, \ldots, S_k\}$ and $k \neq 0$;
  - $\{S_1, \ldots, S_k\}$ is a set of statecharts S_i over Π and Loc, such that $\{S\} \cap boxes(S_i) = \emptyset$ and $boxes(S_i) \cap boxes(S_j) = \emptyset$ for all $i, j \in \{1, k\}$ with $i \neq j$.

Let $boxes(\mathrm{OR}(S, \{S_1, \ldots, S_k\})) = \{S\} \cup \bigcup_{i \in \{1,k\}} boxes(S_i)$.

- $S : \mathrm{AND}(S_1, S_2)$ is an AND *timed statechart*, where S_1, S_2 are statecharts over Π and Loc, $S \in Loc$, $\{S\} \cap boxes(S_i) = \emptyset$ and $boxes(S_i) \cap boxes(S_j) = \emptyset$, for all $i, j \in \{1, 2\}$ with $i \neq j$. Let $boxes(S) = \{S\} \cup \bigcup_{i \in \{1,2\}} boxes(S_i)$.

A *timed statechart* is either a basic, AND or refined timed statechart.

Example 2 The statechart of figure 2 is the textual representation of $\mathrm{OR}(H, \{A, D\})$ where H, A and D are the following basic statecharts:

$$H : (\{A, D\}, \{t_1, t_2\}, in_1, out_1, \chi_1, \tau_1, A)$$

$$A : (\{B, C\}, \{t_3, t_4\}, in_2, out_2, \chi_2, \tau_2, B)$$

$$D : (\{E, F\}, \{t_5, t_6\}, in_3, out_3, \chi_3, \tau_3, E)$$

where

$in_1(t_1) = A$	$in_1(t_2) = D$	$out_1(t_1) = D$	$out_1(t_2) = A$
$\chi_1(t_1) = d/nil$		$\chi_1(t_2) = d/nil$	
$\tau_1(t_1) = (0, \infty)$		$\tau_1(t_2) = (0, \infty)$	
$in_2(t_3) = B$	$in_2(t_4) = C$	$out_2(t_3) = C$	$out_2(t_4) = B$
$\chi_2(t_3) = b/c$		$\chi_2(t_4) = a/b$	
$\tau_2(t_3) = (0, \infty)$		$\tau_2(t_4) = (0, \infty)$	
$in_3(t_5) = E$	$in_3(t_6) = F$	$out_3(t_5) = F$	$out_3(t_6) = E$
$\chi_3(t_5) = b/c$		$\chi_3(t_6) = a/b$	
$\tau_3(t_5) = (0, \infty)$		$\tau_3(t_5) = (0, \infty)$	

In the following we consider a restricted set of statecharts satisfying the following restrictions. For a statechart S, we denote by $T(S)$ the set of transitions of S, namely the union of sets T such that $S' : (\{S_1, \ldots, S_k\}, T, in, out, \chi, \tau, S_i)$ and $S' \in boxes(S)$. For a transition $t \in T(S)$, let $trigger(t) = I$ and $action(t) = O$, where $\chi(t) = I/O$, and let $min(t)$ and $max(t)$ denoting the first and the second component of $\tau(t)$ respectively.

We require that for each transition $t \in T(S)$:

1. $min(t) \leq max(t)$;
2. if $trigger(t) \neq nil$, then $min(t) = 0$ and $max(t) = \infty$.

The first restriction ensures the enabling interval of transitions to be non-negative. The second one guarantees transitions depending on input events to be always enabled with respect to time. This choice allows us to have a more simple and clear semantics where timing constraints depend on the time spent in boxes only. Transitions depending on input events have to be executed immediately whenever the conditions specified by the trigger are satisfied (*immediate transitions*). On the contrary, *timed transitions* can be arbitrarily delayed until the maximal delay.

In the following we denote by $T_1(S) \subseteq T(S)$ the subset of immediate transitions and by $T_2(S) = T(S) \setminus T_1(S)$ the subset of timed transitions.

Example 3 Consider the timed statechart depicted in figure 2 and replace label d/nil with label $[2,2]nil/d$. The transitions from A to D and from D to A are timed and can be executed whenever states A and D have been continuously entered for two units of time. The other transitions are immediate. The timing constraint $[0,\infty]$ for immediate transitions is omitted in the graphical specification.

3 The Semantics of Timed Statecharts

Statecharts is based on the *synchrony hypothesis* [11]: "the reaction of the system to environment inputs is instantaneous". This approach relies on the following assumptions, which are reasonable in practice for a wide class of reactive systems:

1. the environment can be described as a discrete process, namely as an infinite sequence of inputs I_1, $I_2 \ldots$ occurring at successive instants of time;

2. the system is infinitely faster than the environment so that the reaction of the system to inputs I_i is completed before inputs I_{i+1} are produced.

Given these assumptions the behaviour of a reactive system can be abstractly described by the sequence of successive reactions (*steps*) $S_1, S_2 \ldots$ caused by the input events $I_1, I_2 \ldots$. Unfortunately the definition of step given originally revealed some basic problems so that there is not a widely accepted semantics of statecharts. We will briefly explain the main approaches and we refer the reader to [48, 25] for a detailed explanation. The basic idea of statecharts semantics, when one assumes the synchrony hypothesis, is that the execution of transitions generates at the same time new events which may enable other parallel transitions. Therefore, an instantaneous chain reaction of transitions typically arise respecting the principle that enabled transitions have to be immediately executed (*maximal progress*).

Example 4 Let us consider the statechart G of figure 1 where transitions from B to C and from E to F are labelled by a/b and b/c respectively. Suppose the control to be in states B and E and the environment to produce event a. Both transitions are executed, since the first instantaneously produces event b which enables the second.

However, suppose the transition from E to F to be labelled by b/a. What is an admissible step whenever no input is given by the environment? It seems reasonable to consider the idle step (no transition) as a possible step, since no transition is enabled with respect to the environments input. On the other hand, the performance of both transitions generates at the same time both event a and b. Since both transitions are enabled with respect to the current set of generated events $\{a, b\}$, also the firing of both transitions could be consistently considered a step.

Following [20] a widely accepted solution to solve these problems connected to synchrony hypothesis is that of requiring a causal ordering between transitions of a step (*causality principle*). With this interpretation, in the second case of example 4, only the idle step is allowed. The other one is not admissible since each transition causes the other. In [20] a step is formally defined as a maximal sequence of *micro-steps* (sets of parallel transitions), where each micro-step has to be enabled with respect to the set of events produced by the previous ones starting from the external events.

However, the micro-step approach presents serious problems in presence of negated events. Negated events are essential to specify relevant notions like priorities [34].

Example 5 Let us consider the statechart G of figure 1 where transitions from B to C and from E to F are labelled by $\neg a/b$ and b/a respectively. If the control is in states B and E and if no event is produced by the environment it is not obvious which step should be considered.

The first transition is obviously enabled with respect to external inputs and its execution produces event b which enables the second. In the original proposal of [20] both transitions have to be executed. However, the performance of the transition with label b/a generates instantaneously event a, which disables the one with label $\neg a/b$. This is a well-known *causal paradox* where a transition causes a reaction disallowing its own causes.

The problem of the micro-step semantics is that the absence of negated events is checked with respect to the outputs of the previous micro-steps only. A convincing solution to deal with negated events consists of requiring transitions to be enabled with respect to the whole set of generated events (*global consistency*). In the approach of Pnueli and Shalev [42] a step is defined as a maximal set of parallel transitions satisfying the following requirements:

1. the transitions are enabled with respect to the whole set of current events (*global consistency*);
2. all enabled transitions are performed (*maximal progress*);
3. they are causally related (*causality principle*).

In this framework classical causal paradoxes are ruled out in a very reasonable way. For instance in example 5 no admissible step is possible. Therefore the reaction is undefined, namely a *run-time error* occurs. Even with global consistency other semantics have been studied to have always a defined reaction [35].

An alternative approach for statecharts semantics is the one of [19] where the synchrony hypothesis is slightly modified. Events produced by transitions are actually not instantaneously produced.

For timed statecharts we extend the Pnueli and Shalev semantics by combining synchrony hypothesis, global consistency and causality principle with discrete time.

For a set C of clocks, $\pi : C \to N$ is a *clock assignment* over C. For $\delta \in N$, we denote by $\pi + \delta$ the clock assignment, such that for each $x \in C$, $\pi + \delta(x) = \pi(x) + \delta$ (and analogously for $\pi - \delta$). Moreover for $C_1 \subseteq C$, $\pi[C_1 := 0]$ denotes the assignment such that, $\pi(y) = \pi[C_1 := 0](y)$, for $y \notin C_1$, and $\pi[C_1 := 0](x) = 0$ for $x \in C_1$. Let

$C_1 \subseteq C$. We define $pro_{C_1}(\pi) : C_1 \to N$, as $\pi(x) = pro_{C_1}(\pi)(x)$, for each $x \in C_1$. For C_1 and C_2 with $C_1 \cap C_2 = \emptyset$, and for $\pi_1 : C_1 \to N$ and $\pi_2 : C_2 \to N$, we define $\pi_1 \cup \pi_2 : C_1 \cup C_2 \to N$, such that $\pi_1 \cup \pi_2(x) = \pi_i(x)$, for $x \in C_i$ and $i \in \{1, 2\}$. Moreover, for $C_1 \subseteq C$ and C_2, such that $C_1 \cap C_2 = \emptyset$, and for $\pi : C \to N$, we denote by $reset(\pi, C_1, C_2) : C_1 \cup C_2 \to N$ the clock assignment $pro_{C_1}(\pi) \cup \{C_2 := 0\}$.

Let $S_1, S_2 \in boxes(S)$. We denote by $lca(S_1, S_2) = S_3$ such that $S_3 \in boxes(S)$, $S_1 \in boxes(S_3)$ and $S_2 \in boxes(S_3)$ and, for each $S_4 \in boxes(S)$ such that $S_1 \in boxes(S_4)$ and $S_2 \in boxes(S_4)$, we have $S_4 \in boxes(S_3)$.

Definition 6 Let S be a timed statechart over Π and Loc. A configuration $c \subseteq Loc$ is a maximal set such that $S \notin c$ and for each $S', S'' \in c$, either $lca(S', S'')$ is an AND statechart or $lca(S', S'') = S''$ or $lca(S', S'') = S'$. A *timed configuration* is a pair (c, π), where c is a configuration and π is a clock assignment over c.

We denote by $C(S)$ and by $TC(S)$ the set of configurations and of timed configurations of S respectively. The *initial timed configuration* of S is $(\Delta(S), \pi_0)$, where $\pi_0(S') = 0$ for $S' \in \Delta(S)$, and $\Delta(S)$ is the configuration c such that for each $S' \in boxes(S)$ if $S' : (\{S_1, \ldots, S_k)\}, T, in, out, \chi, \tau, S_i)$ is a basic statechart, then $S_i \in c$.

A configuration is a maximal set of boxes which are in parallel components. Note that $\Delta(S)$ is the default configuration where all default states are entered. In a timed configuration the assignment π gives the time spent in each box.

Definition 7 Let S be a timed statechart over Π and Loc. For a set of events $I \subseteq \Pi$, for a timed configuration $(c, \pi) \in TC(S)$ and for $T \subseteq T(S)$ we define

- $trig(I) = \{t \in T_1(S) \mid$ for all $e \in trigger(t)$, $e \in I$ and for all $\bar{e} \in trigger(t)$, $e \notin I\}$;
- $rel(c) = \{t \in T(S) \mid out(t) \in c\}$;
- $con(T) = \{t \in T(S) \mid lca(out(t), out(t'))$ is an AND statechart for all $t' \in T\}$;
- $enabled(\pi) = \{t \in T_2(S) \mid min(t) \leq \pi(out(t)) \leq max(t)\}$.
- $En_{(c,\pi),I}(T) = rel(c) \cap con(T) \cap (trig(I \cup gen(T)) \cup enabled(\pi))$, where $gen(T) = \bigcup_{t \in T} action(t)$.

Given a timed configuration (c, π), a set of inputs I and a set of transitions T, $En_{(c,\pi),I}(T)$ gives the set of transitions which are *enabled* in the configuration with respect to the input by assuming the transitions in T to be taken. An immediate transition is enabled iff the whole set of generated signals satisfies the trigger $(trig(I \cup gen(T)))$. By contrast, timed transitions are enabled whenever the source state has been entered for a time included between the minimal and the maximal delay $(enabled(\pi))$. Condition $con(T)$ guarantees transitions to be in parallel components.

A step is formally defined as follows.

Definition 8 Let S be a timed statechart over Π and Loc. Let $(c, \pi) \in TC(S)$ be a timed configuration and $I \subseteq \Pi$ be a set of events. The set of transitions $T \subseteq T(S)$ is an *admissible step* iff

1. $T \subseteq En_{(c,\pi),I}(T)$;

2. $En_{(c,\pi),I}(T) \cap T_1(S) \subseteq T$;

3. for each $t \in T_2(S) \setminus T$ if $t \in rel(c)$ and $max(t) = \pi(out(t))$, then $t \notin rel(c')$;

4. T is *inseparable*, i.e. for each $T' \subset T$, $En_{(c,\pi),I}(T') \cap (T - T') \neq \emptyset$,

where $c' \in C(S)$ is the configuration $c' = c - (\bigcup_{t \in T} boxes(out(t))) \cup (\bigcup_{t \in T} \Delta(in(t)))$. The reached timed configuration is (c', π'), where $\pi' = (pro_{C_2}(\pi) + 1)[C_1 := 0]$ for $C_1 = c' \setminus c$ and $C_2 = c' \cap c$.

The configuration reached by a step T is obtained by entering all the target states of transitions and by resetting the corresponding clocks. For the other states a step causes time to increase of one unit. We use the notation $(c, \pi) \overset{I,O}{\Longrightarrow} (c', \pi')$ to denote a step T from timed configuration (c, π) to (c', π') with respect to input I such that $gen(T) = O$.

Let us explain the main features of this definition. Condition $T \subseteq En_{(c,\pi),I}(T)$ ensures all the transitions to be enabled with respect to the global set of events (global consistency). On the other hand, condition $En_{(c,\pi),I}(T) \cap T_1(S) \subseteq T$ ensures that all the immediate transitions, that are enabled with respect to the set of current events, are performed (maximal progress).

Example 9 Consider the statechart G of figure 1 where the transitions from B to C and from E to F are labelled by a/b and b/c respectively. In a timed configuration containing B and E, if the environment communicates a, $trig(\{a\} \cup \{b\}) \not\subseteq \{t_1\}$ so that $\{t_1\}$ is not an admissible step. Since t_2 is enabled by the performance of t_1 also t_2 has to be executed.

On the other hand, a timed transition may be performed non-deterministically within the minimal and the maximal delay. To force transitions to be executed at the maximal delay the following condition is checked: for each $t \in T_2(S) \setminus T$ and $t \in rel(c)$ and $max(t) = \pi(out(t))$, then $t \notin rel(c')$.

Example 10 Consider the statechart of example 3 and a timed configuration (c, π) such that $B \in c$ and $\pi(A) = 2$. With respect to an input containing a the only possible step is the one taking the statechart in state D even if the transition from B to C is enabled. The high-level box A has to be left because of the maximal delay. In the same situation if the maximal delay of the transition from A to D was $[2, 5]$ the choice between the two transitions would be non-deterministic.

The condition of inseparability ensures the existence of a causality relation between transitions.

Example 11 Consider the causal problem shown in example 4. With respect to the empty input both transitions from B to C and from E to F would be enabled since the whole set of generated events is $\{a, b\}$. The step given by both transitions is ruled out by the inseparability condition.

Note that the choice of restricting timing constraints to states duration only allows us to have a more simple semantics. Allowing timed transitions depending in addition on input events would be difficult and could lead to unexpected situations such as transitions which are arbitrary delayed even if the trigger is satisfied. Example 10 shows that this interpretation of timed transitions allows to suitably model basic real-time features such as priorities, interrupts and time-out by means of high-level transitions.

Remark 12 An important result on the statecharts semantics which has to be mentioned is the lack of *modularity*. A semantics of a reactive system is modular whenever the semantics of the parallel composition of two systems can be obtained by observing the input-output behaviour of the two components. A well-known result [25] states that synchrony hypothesis, causality and modularity cannot be combined together. The problem of finding a good compromise leading to a modular and reasonable semantics is still under investigation. See for instance [36].

4 The Language $\mathcal{TSP}$

In this section we introduce a process algebra for modelling timed statecharts: the process language $\mathcal{TSP}$. A statecharts process algebra provides a more abstract and structured representation of statecharts, which has several advantages:

1. a labelled transition system semantics can be more simply defined;
2. classical compositional verification methods (see section 8) can properly adapted to the statecharts framework.

The idea of a statecharts process algebra is not new [30, 32, 45, 47]. We consider a timed extension of the language presented in [30] whose main features, with respect for instance to CCS [38], are locations corresponding to statecharts boxes and a new operator of refinement. Refinement is necessary to model the statecharts hierarchical structure. The explicit use of locations is particularly useful in the timed framework to easily evaluate the enabling conditions on time.

Definition 13 Let Π be a set of primitive events and Loc be a set of locations. $\mathcal{TSP}$ processes are defined as follows:

$$\begin{array}{lr} b ::= nil \mid b_1 + b_2 \mid [m, M]a.t & \\ t ::= \ell :: b \mid rec\ x.t \mid t \bigtriangledown p \mid x & \text{Terms} \\ p ::= \ell :: t \mid \ell :: p_1 \times p_2 & \text{Processes} \end{array}$$

where $a \in Lab(\Pi)$, $x \in Var$, $\ell \in Loc$, $m \in N$, $M \in N \cup \infty$ and $m \leq M$.

The process grammar is quite complex since the tree-like structure of statecharts has to be simulated. The constructors $+$, rec, $\times$ stand respectively for *choice*, *recursion* and *parallel composition*. Timed prefixing is denoted by $[m, M]a.$, where a is a statecharts transitions label and $[m, M]$ is the timed enabling interval. The new constructor of process refinement is denoted by $\bigtriangledown$.

A process is a term enclosed into a location corresponding to the outermost box of statecharts. We denote by $\mathcal{T}$ the set of terms. We say that a process p is closed if it does not have free variables according to the usual notion of free recursion variables $free(p)$.

The construction of $\mathcal{TSP}$ processes corresponding to statecharts is quite straightforward. Each statechart is represented by a process whose main location is the outermost box. The key step consists of the representation of basic statecharts; both refined and AND statecharts can be obtained by composing the processes of components with $\bigtriangledown$ and $\times$.

Example 14 Let us give an idea of how the process corresponding to the statechart of figure 3 is obtained. Let us assume for simplicity that a_i is the timed action corresponding to the label and the timed interval of transition t_i for $i \in \{1, 6\}$.

Basic statechart B is modelled by a process $B :: t_B$ where t_B is a term representing the internal behaviour, i.e. transitions t_1 and t_5. The idea to model a transition is that of building a term where the timed action corresponding to the label is executed in the source location and evolves into the process corresponding to the target location.

Assume for instance that transition t_5 does not exist. The term representing location E would be $t_E = E :: nil$ since no transition exits from E. Therefore, the term representing location D would be

$$t_D = rec\ d.D :: a_1.t_E.$$

The statechart could be best represented by $t_D = D :: a_1.t_E$ without any recursive process. However, recursion is needed whenever also transition t_5 is present in order to have a finite representation. In this case the term representing location E would be $t_E = rec\ e.E :: a_5.d$ with free variable d. Thus the whole term representing D would be

$$t_D = rec\ d.D :: a_1.rec\ e.E :: a_5.d.$$

Finally the process obtained for the basic statechart B is $B :: t_B$ where $t_B = tD$.x Since D is the default box of B the initial configuration is the one described by term of location D, i.e. t_D.

Analogously basic statecharts C and I are represented by processes $C :: t_C$ and $I :: t_I$ where

$$t_C = rec\ f.F :: a_2.rec\ g.G :: a_6.f$$

$$t_I = rec\ a.A :: a_3.rec\ h.H :: a_4.a.$$

Statechart A : AND(B, C) is modelled by $A :: t_A$, where $t_A = B :: t_B \times C :: t_C$ is obtained by parallel composition. On the other hand, the process corresponding to the refined statechart OR$(I, \{A, H\})$ is obtained by refining locations A and H in $I :: t_I$ with $A :: t_A$ and $H :: nil$ respectively. Therefore we have

$$I :: rec\ a.\Big(A :: a_3.rec\ h.(H :: a_4.a) \bigtriangledown H :: nil) \bigtriangledown A :: t_A\Big).$$

Before introducing the formal definition of the translation $proc(S)$ we need some preliminary definitions. Let $p, p_1, \ldots, p_n$ be $\mathcal{TSP}$ processes over Π and Loc and let $\{l_i\}_{i\in\{1,n\}} \subseteq Loc$. We define $ref(p, \{(l_1, p_1), \ldots, (l_n, p_n)\})$ as the process obtained by substituting in p each occurrence of a term $l_i :: t'$ with $l_i :: t' \bigtriangledown p_i$.

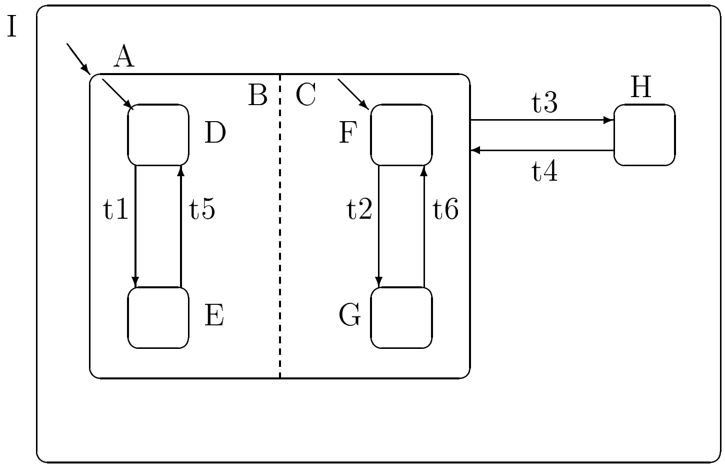

Figure 3: A statechart

Definition 15 • Let $S : (\{S_1, \ldots, S_n\}, T, in, out, \chi, \tau, S_j)$ be a basic timed statechart. Let $t_{S_i} = rec\ s_i.S_i :: \sum_{(t \in T | out(t) = S_i)} [min(t), max(t)]\chi(t).in(t)$, if there exists at least one $t \in T$ with $out(t) = S_i$, and $t_{S_i} = S_i :: nil$ otherwise.

We define $proc(S)$ as the process obtained from the following steps starting with $i = 0$:

- $proc^0(S) = S :: t_{S_j}$;
- $proc(S) = proc^i(S)$ if $proc^i(S)$ is closed;
- $proc^{i+1}(S) = proc^i(S)[t_{S_j}/s_j]$ for some $s_j \in free(proc^i(S))$ if $proc^i(S)$ is open.

• Let S be a refined statechart. We define

$$proc(\mathrm{OR}(S, \{S_1, \ldots, S_n\})) = ref(proc(S), \{(S_1, proc(S_1)), \ldots, (S_n, proc(S_n))\}).$$

• Let S be an AND statechart. We define

$$proc(S : \mathrm{AND}(S_1, S_2)) = S :: proc(S_1) \times proc(S_2).$$

The processes obtained from definition 15 are not minimal. Some recursions are not necessary, but they are due to the generality of the translation.

Processes corresponding to statecharts have some useful features: locations are all distinct and timed actions $[m, M]a$ satisfy the following restriction: if the associated timed interval is different from $[0, \infty]$ then $a = I/O$ such that $I = nil$. Therefore, in analogy with statecharts transitions actions can be divided into timed and immediate actions.

Moreover, recursion satisfies a regularity condition which guarantees processes to be finite-states. A closed process p is *regular* iff it has guarded recursion and for each variable x, x does not occur free in a subprocess $\ell :: p_1 \times p_2$ and in process p_1, such that $\ell :: t \bigtriangledown p_1$ is a subprocess of p. In the following we will consider the class of $\mathcal{TSP}$ processes satisfying the previous restrictions only.

5 The Semantics of $\mathcal{TSP}$ Processes

We define a labelled transition system semantics of closed $\mathcal{TSP}$ processes. Such a semantics is equivalent to the one of timed statecharts (section 3). The main problem for defining a LTS semantics is that the semantics of timed statecharts, as well as the semantics of Pnueli and Shalev and most of the semantics, is not modular [25] (see remark 12). Therefore, the behaviour of a compound statechart cannot be obtained by composing the input-output behaviour of components. In [30, 33] it has been shown that a compositional LTS semantics for un-timed statecharts processes can be obtained by considering complex labels containg the following information:

1. the causal relationship between input and output;
2. the type: idle or non-idle.

Intuitively, the causal information is due to the causality principle, while the representation of idle steps is connected to x maximal progress and consistency of steps. Both these observations are needed to achieve a semantics equivalent to the one of Pnueli and Shalev. For instance without idle steps it is not possible to compute an equivalent semantics as in [47]. The problem is that idle steps are confused with undefined reactions.

We extend the approach of [33, 30] to the timed framework by considering timed states and timed actions in analogy with timed statecharts semantics. Since locations are explicitly represented, the time spent in a location can suitably be counted by corresponding clocks. Therefore, states of the LTS correspond to a process together with a clock assignment for its locations. Transitions correspond to steps and labels represent the required information concerning steps as in [30].

5.1 The LTS Labels

In this section we present the syntax of labels representing steps and the operations over labels. For a more detailed explanation we refer the reader to [33, 30].

For sets $I, I_1, I_2 \subset \Pi$, where $I_1 \subseteq I$, we use the notation $I[I_2/I_1]$ to denote the replacement of subset I_1 by I_2.

Definition 16 Let Π be a finite set of primitive events. We define the set of *causal actions* $\Sigma(\Pi)$ as the set of pairs $(\ell, \prec)$ where

1. $\ell \in 2^{\Pi \cup \bar{\Pi}}$ such that ℓ is *consistent*;
2. $\prec$ is an ordering over $2^\ell \times (2^\ell \cap \Pi)$ such that:
   (a) $\prec$ is an *irreflexive* ordering relation;
   (b) for each $C, C' \in 2^\ell$ such that $C' \subseteq C$, if $C'' \prec C'$ then $C[C''/C'] \prec C$ (*transitivity*).

A causal action $(\ell, \prec)$ represents the conditions (events and negated events) involved in a step (ℓ) and the causal relation between input and output $(\prec)$. The consistency of ℓ (see section 2) ensures global consistency, e and $\bar{e}$ cannot be satisfied at the same time. The relation $\prec$ between subset of events and negated events gives a causal ordering. For instance $\{a,b\} \prec \{c\}$ means that the presence of both a and b causes event c to be produced. On the other hand, $\{a\} \prec \{c\}$ and $\{b\} \prec \{c\}$ means that both a and b are enough for event c to be produced. The relation is transitive and irreflexive. The notion of transitivity is motivated by the following example: if we have $\{b\} \prec \{c\}$ and $\{a,c\} \prec \{d\}$ we want $\{a,b\} \prec \{d\}$.

A causal action represents a step which is triggered by the least conditions in the ordering and which produces the greatest events in the ordering. For a causal action $\sigma = (\ell, \prec)$, we define

1. $trigger(\sigma) = \{e \in \Pi \cup \bar{\Pi} \mid \not\exists C, C' \in 2^{\Pi \cup \bar{\Pi}}$ such that $C \prec C'$ and $e \in C'\}$;
2. $action(\sigma) = \ell \setminus trigger(\sigma)$.

Let $\sigma_0 = (\{nil\}, \emptyset)$ be the causal action that is always enabled.

For basic steps corresponding to a single transition the definition of a corresponding causal action consistent with the input-output behaviour is quite straightforward.

Definition 17 Let $I/O \in Lab(\Pi)$ be a transition label. We define

$$\mathcal{C}(I/O) = (I \cup O, \{I \prec \{e'\} \mid e' \in O - \{nil, I\}\}).$$

The main problem is that of defining a suitable composition of causal actions to deal with parallel composition. For this purpose, we introduce an operation $\otimes$ of *parallel composition* over causal actions. A causal action σ resulting from $\sigma_1 \otimes \sigma_2$ has to properly represent the steps obtained by the parallel execution of the steps corresponding to σ_1 and σ_2. The definition is quite involved because of the internal communication of events which has to be combined with causality and consistency.

Given two orderings $\prec_1, \prec_2$, we denote by $(\prec_1 \cup \prec_2)^+$, the minimal ordering $\prec$, such that $\prec_1 \cup \prec_2 \subseteq \prec$ and such that, if $C \prec C'$ and $C' \prec C''$, then $C \prec C''$ and such that, for each C, C', if $C' \subseteq C$ and $C'' \prec C'$, then $C[C''/C'] \prec C$.

Definition 18 Let Π be a set of primitive events. Given $\sigma_1 = (\ell_1, \prec_1) \in \Sigma(\Pi)$ and $\sigma_2 = (\ell_2, \prec_2) \in \Sigma(\Pi)$, we define $\sigma_1 \otimes \sigma_2 = \{(\ell, \prec) \in \Sigma(\Pi)$ such that

- $\ell = \ell_1 \cup \ell_2$ if $\ell_1 \cup \ell_2$ is consistent;
- $\prec \subseteq (\prec_1 \cup \prec_2)^+$ such that $\prec$ is the maximal ordering with $(\ell, \prec) \in \Sigma(\Pi)\}$.

The parallel composition of actions $(\ell_1, \prec_1)$ and $(\ell_2, \prec_2)$ is obtained by the union of events $\ell_1 \cup \ell_2$ (if they are consistent) and by the ordering, which is the maximal irreflexive subset of $(\prec_1 \cup \prec_2)^+$.

Let us explain the main features of $\otimes$ through simple examples.

Example 19 Consider $\sigma_1 = (\{a,b\}, \{a\} \prec_1 \{b\})$ and $\sigma_2 = (\{b,c\}, \{b\} \prec_2 \{c\})$ corresponding for instance to two parallel transitions with labels a/b and b/c. We have $\sigma_1 \otimes \sigma_2 = \{\sigma\}$ such that $\sigma = (\{a,b,c\}, \{a\} \prec \{b\}, \{b\} \prec \{c\}, \{a\} \prec \{c\})$. The internal communication of events is suitably realized by the transitive closure of the two orderings. The compound step represented by σ is triggered by event a and produces both c and b (event b is actually caused by a).

Note that $(\prec_1 \cup \prec_2)^+$ may be reflexive.

Example 20 Consider $\sigma_1 = (\{a,b\},\{a\} \prec_1 \{b\})$ and $\sigma_2 = (\{a,b\},\{b\} \prec_2 \{a\})$ corresponding to two parallel transitions with labels a/b and b/a. The parallel composition of σ_1 and σ_2 is $\sigma_1 \otimes \sigma_2 = \{\sigma_1, \sigma_2\}$. Therefore, either event a or event b has to be given by the environment in order to trigger the compound step. Note that $\sigma = (\{a,b\},\prec) \notin \sigma_1 \otimes \sigma_2$, if both $\{a\} \prec \{b\}$ and $\{b\} \prec \{a\}$, since $\prec$ would be reflexive. The step represented by σ would indeed violate the causality principle as discussed in example 4.

Another complementary information needed in LTS labels concerns the explicit representation of idle steps. Intuitively, idle steps have to be distinguished from undefined reactions, which are like run-time errors. There are two types of LTS labels: *non-idle* (ϵ) and *idle* ($\bar{\epsilon}$).

Definition 21 Let Π be a set of primitive events. We define the set of LTS *labels* $LTS(\Pi) = \{\sigma : \tau \mid \sigma \in \Sigma(\Pi), \tau \in \{\epsilon, \bar{\epsilon}\}\}$.

The operation $\otimes$ can be trivially extended to LTS labels according to the idea that the type of the parallel composition is idle iff both actions are idle. For $\alpha = \sigma : \tau \in LTS(\Pi)$, we *type*$(\alpha)$ is $x\tau$. Let $\tau_1, \tau_2 \in \{\epsilon, \bar{\epsilon}\}$. We define $\tau_1 \otimes \tau_2 = \tau$ with $\tau \in \{\epsilon, \bar{\epsilon}\}$ as $\tau = \epsilon$ if $\tau_i = \epsilon$ for some $i \in \{1,2\}$, while $\tau = \bar{\epsilon}$ otherwise.

Definition 22 Let Π be a set of primitive events. For $\alpha_i = \sigma_i : \tau_i \in LTS(\Pi)$ for $i \in \{1,2\}$, we define $\alpha_1 \otimes \alpha_2 = \{\sigma : \tau_1 \otimes \tau_2 \in LTS(\Pi) \mid \sigma \in \sigma_1 \otimes \sigma_2\}$.

Finally, we introduce an operation of *conflict* which is needed to derive idle steps. The conflict action of a transition label is a causal action which is enabled iff the transition is not enabled.

Definition 23 Let Π be a set of primitive events and let $I/O \in Lab(\Pi)$. We define

$$\#(I/O) = \{(\{e\}, \emptyset) \in \Sigma(\Pi) \mid e \in \Pi \cup \bar{\Pi}, \bar{e} \in I.$$

The idea is that $\sigma \in \#(I/O)$ if for each input $I \subseteq \Pi$, I enables σ if and only if I does not enable I/O. Consider a transitions label $a, b/c$. We have $\#(a, b/c) = \{(\{\bar{a}\}, \emptyset), (\{\bar{b}\}, \emptyset)\}$. If either event a or event b is not produced then the transition is not enabled.

5.2 The LTS construction

We use the notions of section 5.1 to define a LTS semantics of $\mathcal{TSP}$ processes. Timed states are given by a process and by an assignment of time to its locations. The semantics of timed actions is analogous to the one of timed transitions: immediate actions have to be executed when enabled, while timed actions can be delayed until the maximal delay. We need some preliminary concepts.

We introduce a function *loc* which gives for any process a subset of locations corresponding to the current configuration. The definition of $loc(rec\ x.t)$ is based on the assumption that recursion is guarded.

Let $loc(\ell :: t) = loc(t)$ and $loc(\ell :: p_1 \times p_2) = loc(p_1) \cup loc(p_2)$ respectively. For $t \in \mathcal{T}$, $loc(t)$ is given by $loc(\ell :: b) = \{\ell\}$, $loc(rec\ x.t) = loc(t[rec\ x.t/x])$, $loc(t \bigtriangledown p) = loc(t) \cup loc(p)$ and $loc(x) = \emptyset$.

Moreover in order to evaluate the enabling of transitions with respect to time we exploit a function *delay* which gives for any process a constraint on the timed actions delays.

Definition 24 Let C be a set of clocks. A *clock constraint* is inductively defined by the grammar $\omega ::= x\mathcal{R}n \mid x+n\mathcal{R}y+m \mid \omega_1 \wedge \omega_2 \mid \neg\omega$ with $x, y \in C$ and $n, m \in N$, $\mathcal{R} \in \{\geq, \leq\}$.

We denote by $\Omega(C)$ the set of clock constraints over clocks C. Clock constraints of $\Omega(C)$ are evaluated with respect to clock assignments over C. We say that $\pi \models \omega$ if the assignment π satisfies ω.

$$delay(\ell :: p_1 \times p_2) \equiv delay(p_1) \wedge delay(p_2)$$

$$delay(\ell :: rec\, x.t) \equiv delay(\ell :: t[rec\, x.t/x])$$

$$delay(\ell :: t \bigtriangledown p) \equiv delay(\ell :: t) \wedge delay(p)$$

$$delay(\ell_1 :: \ell_2 :: nil) \equiv true$$

$$delay(\ell_1 :: \ell_2 :: [m, M]a.t) \equiv \ell_2 \leq M$$

$$delay(\ell_1 :: \ell_2 :: b_1 + b_2) \equiv delay(\ell_1 :: \ell_2 :: b_1) \wedge delay(\ell_1 :: \ell_2 :: b_2)$$

Definition 25 Let $p \in \mathcal{TSP}$ be a closed process over Π and *Loc*. We define the LTS of p as $LTS(p) = (TS, ts_0, LTS(\Pi), \mapsto)$ where

1. TS is the set of timed states (p', π), such that p' is a closed process and π is an assignment over $loc(p')$, such that $\pi \models delay(p')$;
2. $ts_0 = (p, \pi_0)$ with $\pi_0(\ell) = 0$, for each $\ell \in loc(p)$;
3. the transition relation $\mapsto \subseteq TS \times LTS(\Pi) \times TS$ is obtained by the rules of tables 1 and 2.

Let us explain the semantics. Consider a basic process $\ell_1 :: \ell_2 :: [m, M]a.t$. Action a can be performed non-deterministically within its minimal and maximal delay. The corresponding transitions are labelled by the causal action representing a of type ϵ (rule *non-idle*). Note that the resulting assignment for $\ell_1 :: t$ is obtained by setting to zero the value of the new locations which have been just entered.

Suppose a to be an immediate action (e.g. $\#(a) \neq \emptyset$) which depends on inputs only. In analogy with statecharts semantics, action a has to be executed immediately whenever it is enabled. Therefore, the process can be idle only if the environment does not enable a. This is obtained by rule *idle* by taking an action in conflict with a ($\sigma \in \#(a)$) with type $\bar{\epsilon}$. In contrast, if action a is timed (e.g. $\#(a) = \emptyset$), then a does not depend on the environments inputs and can be in any case delayed until M. This is modeled by the special action σ_0 that is enabled with respect to any environment. The side condition $\pi_1 + 1 \models delay(\ell_1 :: \ell_2 :: [m, M]a.t)$ forces a to be performed at the maximal delay.

$$\frac{\pi_1 \models m \leq \ell_2 \leq M}{(\ell_1 :: \ell_2 :: b, \pi_1) \overset{\mathcal{C}(a):\epsilon}{\mapsto} (\ell_1 :: t, \{loc(t) := 0\})} \text{ non-idle}$$

$$\frac{\pi_1 + 1 \models delay(\ell_1 :: \ell_2 :: b) \quad (\sigma \in \#(a)) \vee (\sigma = \sigma_0 \wedge \#(a) = \emptyset)}{(\ell_1 :: \ell_2 :: b, \pi_1) \overset{\sigma:\bar{\epsilon}}{\mapsto} (\ell_1 :: \ell_2 :: b, \pi_1 + 1)} \text{ idle}$$

$$(\ell_1 :: \ell_2 :: nil, \pi_1) \overset{\sigma_0:\bar{\epsilon}}{\mapsto} (\ell_1 :: \ell_2 :: nil, \pi_1 + 1) \qquad \text{nil}$$

Table 1: Basic Transitions for $b = [m, M]a.t$

$$\frac{(\ell_1 :: \ell_2 :: b_i, \pi) \overset{\alpha}{\mapsto} (p, \pi')}{(\ell_1 :: \ell_2 :: b_1 + b_2), \pi \overset{\alpha}{\mapsto} (p, \pi')} \text{ choice}_1 \qquad type(\alpha) = \epsilon$$

$$\frac{\{(\ell_1 :: \ell_2 :: b_i, \pi) \overset{\alpha}{\mapsto} (\ell_1 :: \ell_2 :: b_i, \pi_i')\}_{i \in \{1,2\}}}{(\ell_1 :: \ell_2 :: b_1 + b_2, \pi) \overset{\alpha}{\mapsto} (\ell_1 :: \ell_2 :: b_1, \pi_1' \cup \pi_2')} \text{ choice}_2 \qquad type(\alpha) = \bar{\epsilon}$$

$$\frac{(\ell :: t, \pi_1) \overset{\alpha}{\mapsto} (p', \pi_1')}{(\ell :: t \bigtriangledown p, \pi_1 \cup \pi_2) \overset{\alpha}{\mapsto} (p', \pi_1')} \bigtriangledown_1 \qquad type(\alpha) = \epsilon$$

$$\frac{(p, \pi_1) \overset{\alpha}{\mapsto} (p', \pi_1')}{(\ell :: t \bigtriangledown p, \pi_1 \cup \pi_2) \overset{\alpha}{\mapsto} (\ell :: t \bigtriangledown p', \pi_1' \cup (\pi_2 + 1))} \bigtriangledown_2 \qquad \begin{array}{l} type(\alpha) = \epsilon \\ \pi_2 + 1 \models delay(\ell :: t) \end{array}$$

$$\frac{\begin{array}{c}(\ell :: t, \pi_1) \overset{\alpha}{\mapsto} (\ell :: t, \pi_1') \\ (p, \pi_2) \overset{\alpha}{\mapsto} (p, \pi_2')\end{array}}{(\ell :: t \bigtriangledown p, \pi_1 \cup \pi_2) \overset{\alpha}{\mapsto} (\ell :: t \bigtriangledown p, \pi_1' \cup \pi_2')} \bigtriangledown_3 \qquad type(\alpha) = \bar{\epsilon}$$

$$\frac{(\ell :: t[rec\, x.t/x], \pi) \overset{\alpha}{\mapsto} (p, \pi')}{(\ell :: rec\, x.t, \pi) \overset{\alpha}{\mapsto} (p, \pi')} \text{ rec}$$

$$\frac{\{(p_i, \pi_i) \overset{\alpha_i}{\mapsto} (p_i', \pi_i')\}_{i \in \{1,2\}}}{(\ell :: p_1 \times p_2, \pi_2 \cup \pi_2) \overset{\alpha}{\mapsto} (\ell :: p_1' \times p_2', \pi_2' \cup \pi_2')} \times \qquad \alpha \in \alpha_1 \otimes \alpha_2$$

Table 2: Complex Transitions

Rule $choice_1$ is standard. Rule $choice_2$ reflects the maximal progress of statecharts reaction: an idle action must be an idle action of both processes. The constructor $\triangledown$ is similar to the one of choice. However, the choice between the two processes is not symmetric, as shown by rules $\triangledown_1$ and $\triangledown_2$. According to the hierarchical structure a non-idle action of the high-level process causes the low level choice to be lost (rule $\triangledown_1$). A non-idle action of the low level process p is performed within the context $\ell :: t$ (rule $\triangledown_2$), whenever all timed actions of the high-level process $\ell :: t$ can be delayed. As in statecharts semantics (see example 10) high-level timed actions have priority.

Rule $\times$ is the classical rule of parallel synchronous composition, where the composition of LTS labels $\otimes$ realizes the communication between parallel components according to the principles of causality and consistency. Because of the explicit representation of idle actions the synchronous rule directly guarantees maximal progress. A more complex rule would be necessary if conditions for idle steps would have to be derived from non-idle ones. This approach would be even more difficult since idle steps corresponding to undefined reactions should be ruled out.

Example 26 Consider the processes of example 14 modelling the statechart of figure 3. Suppose transitions t_1 and t_2 to be labelled by a/b and b/c respectively. The process corresponding to the parallel statechart A is $A :: p_1 \times p_2$ such that $p_1 = B :: rec\, x.D :: [0,\infty]a/b.t_1$ and $p_2 = C :: rec\, x.F :: [0,\infty]b/nil.t_2$ for the processes $t_1 = rec\, y.E :: [0,\infty]a_5.x$ and $t_2 = rec\, y.G :: [0,\infty]a_6.x$. For every clock assignment π_1 and π_2 we have the following transitions

$$\begin{array}{l}(p_1,\pi_1) \overset{\sigma_1:\epsilon}{\mapsto} (B :: t_1, \pi_1') \\ (p_1,\pi_1) \overset{\sigma_4:\bar{\epsilon}}{\mapsto} (p_1, \pi_1 + 1) \\ (p_2,\pi_2) \overset{\sigma_2:\epsilon}{\mapsto} (C :: t_2, \pi_2') \\ (p_2,\pi_2) \overset{\sigma_3:\bar{\epsilon}}{\mapsto} (p_2, \pi_2 + 1)\end{array}$$

where $\sigma_1 = (\{a,b\},\{a\} \prec_1 \{b\})$, $\sigma_2 = (\{b,c\},\{b\} \prec_2 \{c\})$, $\sigma_3 = (\{\bar{b}\},\emptyset)$, $\sigma_4 = (\{\bar{a}\},\emptyset)$ and $\pi_1'(E) = \pi_2'(G) = 0$. By synchronous parallel composition we have

$$\begin{array}{l}(A :: p_1 \times p_2, \pi_1 \cup \pi_2) \overset{\sigma:\epsilon}{\mapsto} (A :: B :: t_1 \times C :: t_2, \pi_1' \cup \pi_2') \\ (A :: p_1 \times p_2, \pi_1 \cup \pi_2) \overset{\sigma':\bar{\epsilon}}{\mapsto} (A :: p_1 \times p_2, (\pi_1 \cup \pi_2) + 1) \\ (A :: p_1 \times p_2, \pi_1 \cup \pi_2) \overset{\sigma'':\epsilon}{\mapsto} (A :: p_1 \times C :: t_2, \pi_1 + 1 \cup \pi_2')\end{array}$$

where $\sigma \in \sigma_1 \otimes \sigma_2$, i.e. $\sigma = (\{a,b,c\},\{a\} \prec \{b\} \prec \{c\})$, $\sigma' \in \sigma_3 \otimes \sigma_4$, i.e. $\sigma' = (\{\bar{a},\bar{b}\},\emptyset)$, and $\sigma'' \in \sigma_2 \otimes \sigma_4$, i.e. $\sigma'' = (\{\bar{a},b,c\},\{b\} \prec \{c\})$. Action σ represents the step where both transitions are taken on inputs containing a, action σ'' represents the step where only transition t_2 is taken on input containing b and not containing a, action σ' represents the step where no transition is taken whenever both a and b are absent. Note that there are no transitions such that only t_1 is performed, since σ_1 and σ_3 are not consistent. Therefore, component C cannot be idle if component B is not idle.

Suppose the high-level transition t_3 to be labelled by $[1,1]nil/nil$ and suppose π to be a clock assignment such that $\pi(A) = 1$. For the refined process $proc(\mathrm{OR}(I,\{A,H\})$ we have (by rule $\triangledown_1$) only a transition corresponding to the execution of transition t_3

$$(proc(\mathrm{OR}(I,\{A,H\})),\pi) \overset{\sigma_0:\epsilon}{\mapsto} (I :: t, \pi')$$

for the proper process t. No transition corresponds to the execution of an internal step of statechart A, since the side condition of rule $\bigtriangledown_2$ is not satisfied by π. Transition t_3 cannot be delayed unless location A is left. Idle steps do not occur since the high-level process does not allow idle steps due to the side condition of rule non-idle.

The equivalence between the semantics of statecharts and corresponding processes is stated by the following theorem. Let $\alpha = \sigma : \tau \in LTS(\Pi)$ and $I \subseteq \Pi$. We say that I triggers α iff, for each $e \in \Pi \cap trigger(\sigma)$, $e \in I$ and, for each $\bar{e} \in \bar{\Pi} \cap trigger(\sigma)$, $e \notin I$.

Theorem 27 Let S be a timed statechart over Π and Loc and let $LTS(proc(S)) = (TS, ts_0, LTS(\Pi), \mapsto)$.

- For each $(c, \pi) \in TC(S)$ and for each $I \subseteq \Pi$, such that (c, π) is reachable from the initial timed configuration $(\Delta(S), \pi_0)$, if $(c, \pi) \stackrel{I,O}{\Longrightarrow} (c', \pi')$, then
  1. there exists $(p, \pi), (p', \pi') \in TS$, such that $loc(p) = c$ and $loc(p') = c'$;
  2. there exists $(p, \pi) \stackrel{\alpha}{\mapsto} (p', \pi')$ such that I triggers α and $action(\alpha) = O - trigger(\alpha)$;
- For each $(p, \pi) \in TS$ and for each $(p, \pi) \stackrel{\alpha}{\mapsto} (p', \pi')$, then
  1. there exist $(c, \pi), (c', \pi') \in TC(S)$, such that $loc(p) = c$ and $loc(p') = c'$;
  2. there exists $(c, \pi) \stackrel{I,O}{\Longrightarrow} (c', \pi')$, for each $I, O \subseteq \Pi$, such that I triggers α and $O - trigger(\alpha) = action(\alpha)$.

This theorem can be proved by induction on the structure of the statechart. Consistency of steps is guaranteed by the definition of actions $\Sigma(\Pi)$, while maximal progress is guaranteed by the definition of conflict (#) and by the semantics of idle actions (rules *non-idle*, *choice*$_2$ and $\bigtriangledown_3$). The most complex case is that of parallel composition, where the definition of $\otimes$ over LTS labels has to be shown to agree with the condition of inseparability. The proof of theorem 27 can be found in [32].

6 The logic RTL_μ

The modal μ-calculus is a very rich logic which is widely used for verifying correctness properties of reactive and concurrent systems. Classical branching temporal modalities are expressed in μ-calculus by combining the next modalities, the existential $< - >$ and universal $[-]$ with least μ) and greatest fix-points (ν).

For instance, the property *eventually* formula A holds ($\exists \mathbf{E} A$) is expressed by the formula $\mu X.(A \vee < - > X)$. On the other hand, the property *always* formula A holds ($\forall \mathbf{A} A$) is expressed by the formula $\nu X.(A \wedge [-]X)$.

We consider a (discrete) real-time version of μ-calculus, where clock constraints and an operator called *reset*, inspired by freeze quantification [3, 22], are introduced in formulas. Intuitively, a formula $x.A$ holds iff A holds whenever the value of clock x is reset to 0.

Definition 28 Let C be a set of specification clocks, let VAR be a set of logical variables, let $PROP$ be a set of propositional symbols and let $\mathcal{A}$ be a set of actions. Formulas of the real-time μ-calculus RTL_μ are defined as follows

$$A ::= X \mid P \mid \neg P \mid A_1 \vee A_2 \mid A_1 \wedge A_2 \mid \omega \mid < \alpha > A \mid [\alpha]A \mid x.A \mid \mu X.A \mid \nu X.A$$

where $\omega \in \Omega(C)$ is a clock constraint, $x \in C$, $\alpha \in \mathcal{A}$, $P \in PROP$, $X \in VAR$.

For an RTL_μ formula A we denote by $fv(A) = \{x \in C \mid x$ does not occur inside the scope of $x.\}$, $FV(A) = \{X \in VAR \mid X$ does not occur inside the scope of $\mu X.$ and $\nu X.\}$. Formula A is closed if $FV(A) = \emptyset$.

The modal operator $< \alpha >$ is the existential next state corresponding also to next time, while $[\alpha]$ is its dual. The quantifier $x.$ is the *reset*. Formulas $\mu X.A$ and $\nu X.A$ stand for least and greatest fix-points respectively.

Formulas are interpreted over a model $\mathcal{M} = ((TS, ts_o, \mathcal{A}, \mapsto), \gamma)$, where γ is an evaluation function for propositional symbols ($\gamma : PROP \to 2^{TS}$) and $(TS, ts_o, \mathcal{A}, \mapsto)$ is a labelled transition system, such that

- TS is a set of timed states (s, π), where $s \in S$ is a state and $\pi : C' \to N$ is a clock assignment over C', such that $C \cap C' = \emptyset$;
- $ts_0 = (s_0, \pi_0)$ is the initial state;
- $\mapsto \subseteq TS \times \mathcal{A} \times TS$ is the next state relation.

Note that the clocks of the labelled transition system must be distinct from the specification clocks of formulas. Only specification clocks of formulas can actually be modified by an operation of reset.

Let $\rho : VAR \to 2^{TS}$ be a valuation for propositional symbols and let $\delta : C \to N \cup \{\perp\}$ be a (partial) clock assignment over C. The semantics of a formula A with respect to a model $\mathcal{M} = ((TS, ts_o, \mathcal{A}, \mapsto), \gamma)$ and to ρ and δ is inductively defined in table 3

The semantics of the logic is standard except for the treatment of time by means of the clock assignment δ. Note that the clock assignment δ is modified in the rules for next modalities and in the rule of reset only. We use the notation $(s, \pi) \models_{\mathcal{M}} A$ if $(s, \pi) \in [\![A]\!]_{\mathcal{M},\rho,\delta}$ for each valuation ρ and clock assignment δ. For a closed formula A such that $fv(A) = \emptyset$, $(s, \pi) \models_{\mathcal{M}} A$ stands for $(s, \pi) \in [\![A]\!]_{\mathcal{M},\rho_0,\delta_0}$, where $\rho_o(X) = \emptyset$ for each $X \in VAR$ and $\delta_0(x) = \perp$ for each $x \in C$.

Clock constraints and reset allows us to express properties concerning the quantitative aspect of time. The logic RTL_μ subsumes indeed the discrete real-time extension of CTL [16]. For instance, the time-bounded property $\exists \mathbf{E}_{\leq 3} A$ is expressed by the formula $x.\mu X.((A \wedge x \leq 3) \vee < - > X)$, while a timed invariance property $\forall \mathbf{A}_{\leq 3} A$ is expressed by the formula $x.\nu X.(((A \wedge x \leq 3) \vee (x > 3)) \wedge [-]X)$.

Properties of $\mathcal{TSP}$ processes over events Π and locations Loc can suitably be expressed by RTL_μ formulas over propositional symbols Loc, actions $LTS(\Pi)$ and specifications clocks C such that $C \cap Loc = \emptyset$. Let $LTS(p) = (TS, ts_0, \Sigma(\Pi), \mapsto)$, and let $\mathcal{M}(p) = (LTS(p), \gamma)$ where $\gamma : Loc \to 2^{TS}$ is such that $\gamma(\ell) = \{(p, \pi) \in TS \mid \ell \in loc(p)\}$ for each $\ell \in Loc$. For a timed state $(p', \pi) \in TS$ and a closed formula A and for a clock assignment δ, we say that $(p', \pi) \models_\delta A$ iff $(p', \pi) \models_{\mathcal{M}(p),\rho_0,\delta} A$.

$$[\![P]\!]_{\mathcal{M},\rho,\delta} = \gamma(P)$$

$$[\![\neg P]\!]_{\mathcal{M},\rho,\delta} = TS \setminus \gamma(P)$$

$$[\![\omega]\!]_{\mathcal{M},\rho,\delta} = \{TS \mid \delta \models \omega\} \cup \{\emptyset \mid \delta \not\models \omega\}$$

$$[\![x.A]\!]_{\mathcal{M},\rho,\delta} = \{(s,\pi) \in TS \mid (s,\pi) \in [\![A]\!]_{\mathcal{M},\rho,\delta[x:=0]}\}$$

$$[\![< \alpha > A]\!]_{\mathcal{M},\rho,\delta} = \{(s,\pi) \in TS \mid \exists s', \pi' \ (s,\pi) \overset{\alpha}{\mapsto} (s',\pi') \text{ and } (s',\pi') \in [\![A]\!]_{\mathcal{M},\rho,\delta+1}\}$$

$$[\![[\alpha]A]\!]_{\mathcal{M},\rho,\delta} = \{(s,\pi) \in TS \mid \forall s', \pi' \ (s,\pi) \overset{\alpha}{\mapsto} (s',\pi') \implies (s',\pi') \in [\![A]\!]_{\mathcal{M},\rho,\delta+1}\}$$

$$[\![\mu X.A]\!]_{\mathcal{M},\rho,\delta} = \mu V.([\![A]\!]_{\mathcal{M},\rho[V/X],\delta})$$

$$[\![\nu X.A]\!]_{\mathcal{M},\rho,\delta} = \nu V.([\![A]\!]_{\mathcal{M},\rho[V/X],\delta})$$

Table 3: The Semantics of the Logic

We use standard generic modalities $< - >$ and $[-]$ where $-$ stands for any action. We omit the formal definition the semantics of $< - > A$ and of $[-]A$, which is the obvious generalization of the one of the existential modality and of the universal modality shown in table 3.

7 An Introduction to the Proof System

In this section we informally present the compositional proof system to check RTL_μ properties of $\mathcal{TSP}$ processes. The proof system is defined on the lines of the proof system of [9], which combines local model checking [49] with compositional rules for a simple process algebra and standard μ-calculus. Two main aspects have to be modified and adapted:

1. the compositional rules have to defined for the specific and complex process operators of statecharts processes;
2. timed features have to be added.

In the following we will introduce the tagged fix-point method of [49] and the compositional approach of [9] and we will discuss the main solutions to adapt these techniques to our framework. The details of the proof system are shown in section 8.

7.1 Tagged Fix-points

Tagged fix-points have been proposed by Winskel [49] as a method of local model checking for μ-calculus. Local model checking consists essentially of applying forward reasoning instead of backward reasoning for exploring the state-space. For instance, proving that a state s satisfies $< \alpha > A$ is reduced to prove that s' satisfies A for some s' such that $s \overset{\alpha}{\mapsto} s'$. The advantage is that only a reduced part of the global state-space may be explored in this way, since it is possible to establish if a state satisfies a formula without computing all the states which satisfy the formula.

The main problem of local model checking is that of dealing with fix-points. Tagged fix-points allow simple unfolding rules to handle fix-points. The idea is that of recording in formulas the states of the model for which the fix-points unfolding has been already done.

Let $\mu X\{S\}.A$ and $\nu X\{S\}.A$ be fix-points labelled by the set of states S (called the tag). Proving state s to satisfy formula $\psi X\{S\}.A$ ($\psi \in \{\mu, \nu\}$) is reduced to check if s satisfies its unfolding $A[\psi X\{S\} \cup \{s\}.A/X]$ where the tag has been updated with s. Simple side conditions on tags are introduced to ensure that the unfolding rule can be applied only once for each state, i.e. to guarantee termination conditions. The method can be formalized as a proof system in a goal directed fashion (top down).

$$\frac{s \vdash \psi X\{S\}.A}{s \vdash A[\psi X\{s\} \cup \{S\}.A/X]}\ \psi \in \{\nu, \mu\}, s \notin S$$

$$\frac{s \vdash \nu X\{S, s\}.A}{}$$

The first rule is the unfolding and can be applied only if the state has not already been considered. The second rule states the proof to be successful for greatest fix-points whenever the current state is already in the tag.

Example 29 Consider an LTS over states $\{s_1, s_2\}$ such that $s_1 \overset{\alpha}{\mapsto} s_2$ and $s_2 \overset{\alpha}{\mapsto} s_1$.

Suppose a proposition P to hold both in state s_2 and s_1. It holds $s_1 \models \nu X.P \wedge [-]X$ (*always* P). The proof has the following shape:

$$\dfrac{s_1 \vdash \nu X.P \wedge [-]X}{\dfrac{s_1 \vdash P \wedge [-](\nu X\{s_1\}.P \wedge [-]X)}{\dfrac{\dfrac{s_1 \vdash P}{} \qquad \dfrac{\dots}{s_2 \vdash \nu X\{s_1\}.P \wedge [-]X}}{\dfrac{s_2 \vdash P \wedge [-](\nu X\{s_1, s_2\}.P \wedge [-]X)}{\dfrac{\dots}{\dfrac{s_1 \vdash \nu X\{s_1, s_2\}.P \wedge [-]X}{}}}}\ \text{unfolding}}}\ \text{unfolding}$$

Note that $s_1 \vdash \nu X\{s_1, s_2\}.P \wedge [-]X$ is an axiom since the invariance property has been already proved to hold in state s_1.

The same termination condition does not hold for least fix-point expressing liveness properties rather than invariance properties. Suppose a proposition P to hold in state s_2 only. It holds $s_1 \models \mu X.P \vee [-]X$ expressing the property that from s_1 for every path a state satisfying P is reached. The proof has the following shape:

$$
\begin{array}{c}
s_1 \vdash \mu X.P \vee [-]X \\
\hline
s_1 \vdash P \vee [-](\mu X\{s_1\}.P \vee [-]X) \\
\hline
\cdots \\
\hline
s_2 \vdash \mu X\{s_1\}.P \vee [-]X \\
\hline
s_2 \vdash P \vee [-](\mu X\{s_1, s_2\}.P \vee [-]X) \\
\hline
s_2 \vdash P \\
\hline
\end{array}
\begin{array}{l}
\text{unfolding} \\
\\
\\
\\
\text{unfolding} \\
(1)
\end{array}
$$

Note that in (1) between the two alternative $s_2 \vdash P$ and $s_2 \vdash [-](\mu X\{s_1, s_2\}.P \vee [-]X)$ the first one is chosen which is successful. Suppose P to not hold in s_2. It would not possible to have a proof since both alternatives in (1) would fail. Actually, $s_2 \vdash [-](\mu X\{s_1, s_2\}.P \vee [-]X)$ reduces to $s_1 \vdash \mu X\{s_1, s_2\}.P \vee [-]X$ which is ruled out by the tags condition.

The soundness of the method is based on well-known properties of fixed-points [28] (fix-points induction). For finite-state systems the tagged fix-points method is also complete. An alternative formulation of tagged fix-points is the one of [44] where the same information is recorded in a more complex way by means of constants definitions recording the checked states.

Our goal is that of extending the tagged fix-point method to timed statecharts processes and to RTL_μ formulas. Time has to introduced carefully in order to guarantee that the proof system is complete at least for processes corresponding to statecharts (finite-states). Even finite-states processes have actually an infinite LTS due to time.

The basic idea is to consider proofs statements such as $p, \omega \vdash A$, where $p \in \mathcal{TSP}$ is a closed process, $\omega \in \Omega(loc(p)) \cup C)$ is a clock constraint on the locations of p and on the specification clocks C, and A is a closed RTL_μ formula. The intuitive interpretation of $p, \omega \vdash A$ is that $(p, \pi) \models_\delta A$ for clock assignments π over $loc(p)$ and δ over C such that $\pi \cup \delta \models \omega$. With an abuse of notation we use $(p, \pi \cup \delta) \models A$ to denote $(p, \pi) \models_\delta A$.

Analogously to proofs statements we can consider fix-points formulas where tags depend on clock constraints. Therefore, tags are sets of pairs (p, ω) where p is a process and ω is a clock constraint. A pair (p, ω) implicitly represents the set of timed processes (p, π) such that $\pi \models \omega$.

The unfolding rules with tags can suitably be adapted to the timed framework with a slight modification of the side conditions. Termination conditions dealing with constraints representing (even infinite) set of states, instead that with single states, have actually to be adopted.

Example 30 Consider for instance a process $p = \ell_1 :: rec\ x.\ell_2 :: [2, \infty]a.x$. We have

$p, \ell_2 \geq 2 \models \nu X.\ell_2 \wedge [-]X$. A proof of *always* ℓ_2 is obtained as follows.

$$\dfrac{p, \ell_2 \geq 2 \vdash \nu X.\ell_2 \wedge [-]X}{\dfrac{p, \ell_2 \geq 2 \vdash \ell_2 \wedge [-]\nu X\{(p, \ell_2 \geq 2)\}.(\ell_2 \wedge [-]X)}{\dfrac{}{p, \ell_2 \geq 2 \vdash \ell_2} \quad \dfrac{p, \ell_2 \geq 2 \vdash [-]\nu X\{(p, \ell_2 \geq 2)\}.(\ell_2 \wedge [-]X)}{\dfrac{p, \ell_2 \geq 3 \vdash \nu X\{(p, \ell_2 \geq)\}.(\ell_2 \wedge [-]X)}{}}}} \text{ unfolding}$$

The side condition for the greatest fix-points axiom exploits the fact that $\models (\ell_2 \geq 3 \supset \ell_2 \geq 2)$. Therefore, the set of timed states represented by $(p, \ell_2 \geq 3)$ has been already checked. Similar conditions are used to guarantee the unfolding rule to be applied only once for each timed state.

Note that the use of clock constraints both in tags and in proofs statements is essential to guarantee the method of tagged fix-points to be complete for timed statecharts processes. Clock constraints allow actually to properly represent infinite set of timed states.

7.2 The Compositional Technique

The proof system of [9] for checking μ-calculus properties of a simple process algebra combines tagged fix-points with compositional reasoning. The proof system is obtained from the one of [49] by replacing the rules for next modalities (which refer to LTS transitions) with compositional rules that allow to derive a proof for a compound process from proofs for derived properties of its components.

Let us introduce the basic ideas. Compositional reasoning for the next modalities is quite simple for process constructors like prefixing and choice with the standard semantics.

$$\frac{a.p \vdash < a > A}{p \vdash A} \text{ pref}$$

$$\frac{p_1 + p_2 \vdash < a > A}{p_i \vdash < a > A}\, i \in \{1, 2\} \text{ choice}$$

The most difficult case is that of parallel composition. Let us assume, as is [9], the parallel composition operator to be modelled by the following rule requiring the synchronous cooperation of processes.

$$\frac{p_1 \stackrel{a}{\mapsto} p_1' \quad p_2 \stackrel{a}{\mapsto} p_2'}{p_1 \times p_2 \stackrel{a}{\mapsto} p_1' \times p_2'}$$

The goal is that of establishing $p_1 \times p_2 \vdash < a > A$ without building LTS transitions. While for choice the proof can simply be deduced from a proof for one of the two processes, in this case both processes have to participate. The proposed solution to solve this problem consists of reducing the proof of property $< a > A$ for process $p_1 \times p_2$ to a proof for one component of a proper formula $< a > A'$ where A' is an *extended formula* containing information on the residual component. Let an extended assertion $(A)_{\times p_1}$ have the following meaning: "formula A holds whenever process p_1 is put in parallel".

Example 31 Let us consider a simple case where $p_2 = a.p'_2$ is a basic not further decomposable process. The following proof is constructed

$$\begin{array}{c}
\dfrac{p_1 \times p_2 \vdash < a > A}{p_1 \vdash < a > (A)_{\times p'_2}}\,(2) \\ \hline
\cdots \\ \hline
\dfrac{p'_1 \vdash (A)_{\times p'_2}}{p'_1 \times p'_2 \vdash A}\,\text{shift} \\ \hline
\cdots
\end{array}$$

The proof for process $p_1 \times p_2$ is derived from a proof for process p_1 of the extended formula $< a > (A)_{\times p'_2}$, where the process resulting by p_2 after the execution of a is recorded. Suppose to be able to check (in a compositional way) the existence of an a transition for process p_1. The whole proof is reduced to establish $p'_1 \vdash (A)_{\times p'_2}$ for the process derived from p_1 after the execution of a. A special rule called *shift* is then applied to build up the complete process $p'_1 \times p'_2$ which has to satisfy A according to the meaning of the extended formula. Note that the decomposition rule (2) can be applied since it is possible to deduce by its syntax that process p_2 performs action a and that the resulting process is p'_2. Whenever p_2 is a structured process it has to be further decomposed before applying rule (2).

The proof system for $\mathcal{TSP}$ and the real-time logic is obtained by defining for each process constructor compositional rules and shift rules similar to the previous ones. Both the complex features of the new operators and time have to be taken into account in the rules to decompose the processes structure and in the shift rules. For untimed statecharts processes the problem has been solved in [32, 33]. We discuss informally three main cases: prefixing, refinement and parallel composition.

We need some preliminary definitions. Assume $\omega \in \Omega(C)$. We denote by $\omega + \delta$ (resp. $\omega - \delta$) for $\delta \in N$, the clock constraint $\omega[x + \delta/x]$ (resp. $\omega[x - \delta/x]$), for each clock $x \in C$. For $C_1 \subseteq C$, we denote by $pro_{C_1}(\omega)$ the first order formula $\exists x_1, \ldots, x_n\ (\omega)$, where $C \setminus C_1 = \{x_1, \ldots x_n\}$. Let $C_1 \subseteq C$, $C_2 \cap C_1 = \emptyset$. We denote by $reset(\omega, C_1, C_2) \equiv pro_{C_1}(\omega) \wedge \bigwedge_{x \in C_2} x = 0$. We use the notation $\forall(\omega)$ for $\forall x_1, \ldots, \forall x_n(\omega)$, where $C = \{x_1, \ldots, x_n\}$.

Intuitively these operations on constraints are the counterparts of the corresponding operations over clock assignments.

7.2.1 Prefixing

Consider a process $p = \ell_1 :: \ell_2 :: [1,4]a.\ell_3 :: nil$ and a clock constraint $\omega \equiv 1 \leq \ell_2 \leq 3$. We have $p, \omega \models < \alpha > \ell_3$ for $\alpha = \sigma : \epsilon$ with $\sigma = \mathcal{C}(a)$. The formula $< \alpha > \ell_3$ express the ability to perform the action a and to reach a configuration containing location ℓ_3. The simple rule for prefixing can be adapted to reflect rule $non-idle$ of the LTS by taking into account the enabling of actions with respect to time. We have a proof as follows.

$$\cfrac{\cfrac{p, \omega \vdash < \alpha > \ell_3}{\ell_1 :: \ell_3 :: nil, \ell_3 = 0 \vdash \ell_3} \; \ell_3 \in loc(\ell_1 :: \ell_3 :: nil)}{} \models (\omega \supset 1 \leq \ell_2 \leq 4)$$

The side condition $\models (\omega \supset 1 \leq \ell_2 \leq 4)$ ensures that rule $non-idle$ is indeed applicable for all assignments satisfying ω. Note that we obtain the clock constraint $\ell_3 = 0$, since location ℓ_3 has been just entered.

7.2.2 Refinement

Consider a process $\ell :: t \bigtriangledown p$, where $t = \ell_1 :: [1,3]a.\ell_2 :: nil$ and $p = \ell_3 :: \ell_4 :: [0,\infty]b.\ell_5 :: nil$. Let $\omega \equiv 1 \leq \ell_1 \leq 2$ and $\alpha_1 = \sigma : \epsilon$ where $\sigma = \mathcal{C}(a)$. We have $\ell :: t \bigtriangledown p, \omega \models < \alpha_1 > true$, since for each $\pi \models \omega$ by rule $\bigtriangledown_1$ $(\ell :: t \bigtriangledown p, \pi) \overset{\alpha_1}{\mapsto} (\ell :: \ell_2 :: nil, \ell_2 = 0)$. According to rule $\bigtriangledown_1$ the proof can be decomposed as follows.

$$\cfrac{\cfrac{\ell :: t \bigtriangledown p, \omega \vdash < \alpha_1 > true}{\ell :: t, \omega \vdash < \alpha_1 > true} \; type(\alpha_1) = \epsilon}{} \models (\omega \supset 1 \leq \ell_1 \leq 3)$$

Note that the side condition $type(\alpha_1) = \epsilon$ is necessary to ensure $\bigtriangledown_1$ to be applicable.

We have also $\ell :: t \bigtriangledown p, \omega \models < \alpha_2 > \ell_1$ expressing the property that the outermost location ℓ_1 has not been left after α_2. Actually by rule $\bigtriangledown_2$ for each $\pi_1 \cup \pi_2 \models \omega$ and for $\alpha_2 = \sigma : \epsilon$ with $\sigma = \mathcal{C}(b)$ we have $(\ell :: t \bigtriangledown p, \pi_1 \cup \pi_2) \overset{\alpha_2}{\mapsto} (\ell :: t \bigtriangledown p', \ell_5 = 0 \cup (\pi_1 + 1))$ with $p' = \ell_4 :: \ell_5 :: nil$.

Because of the hierarchy the resulting process after the execution of α_2 is $\ell :: t \bigtriangledown p'$ rather than p' (see rule $\bigtriangledown_2$). Therefore it would not be correct to deduce $\ell :: t \bigtriangledown p, \omega \vdash < \alpha_2 > \ell_1$ from $p, \omega \vdash < \alpha_2 > \ell_2$ as in the previous case. Since $p, \omega \models < \alpha_2 > \neg \ell_1$ we would actually derive also $\ell :: t \bigtriangledown p, \omega \vdash < \alpha_2 > \neg \ell_1$.

Extended assertions and shift rules can properly be used to record the residual timed process after the decomposition and to afterwards build up the whole process structure. Let $(A)_{(p,\omega)\bigtriangledown}$ be an extended assertion which is satisfied by all processes (p', π') such that $(p \bigtriangledown p', \pi' \cup \pi) \models A$ for each $\pi \models \omega$. Such a formula can be exploited to decompose the proof as follows by assuming $\omega_1 \equiv pro_{loc(p)}(\omega) \equiv true$ and $\omega_2 \equiv 2 \leq \ell_1 \leq 3 \equiv pro_{loc(\ell::t)}(\omega) - 1$.

$$\cfrac{\cfrac{\cfrac{\cfrac{\ell :: t \bigtriangledown p, \omega \vdash < \alpha_2 > \ell_1}{p, \omega_1 \vdash < \alpha_2 > (\ell_1)_{(\ell::t,\omega_2)\bigtriangledown}}}{p', \ell_5 = 0 \vdash (\ell_1)_{(\ell::t,\omega_2)\bigtriangledown}}}{\ell :: t \bigtriangledown p', \omega_2 \wedge \ell_5 = 0 \vdash \ell_2} \; \text{shift}}{} \qquad \begin{array}{l} \models \forall(\omega_2 \supset delay(\ell :: t)) \\ type(\alpha_2) = \epsilon \end{array}$$

The residual after decomposition $(\ell :: t, \omega_2)$ represents the set of high-level processes $\ell :: t$ corresponding to time assignments which satisfy $2 \leq \ell_1 \leq 3$ since time progress of one unit. The side condition $\models \forall(\omega_2 \supset delay(\ell :: t))$ guarantees rule $\bigtriangledown_2$ to be indeed applicable. Shift rules to build up the whole process structure can trivially be adapted to the timed framework by taking into account also the additional clock constraint.

7.2.3 Parallel Composition

The case of parallel composition is the most complex.

Example 32 Suppose that $p_1 \times p_2, \omega \models < \alpha > A$ where $p_2 = \ell_1 :: \ell_2 :: [m, M]a.t$ is a basic process.

A compositional proof of $p_1 \times p_2, \omega \vdash < \alpha > A$ can be simply obtained by using proper extended assertions as in example 31. Let $(A)_{\ell \times (p_2, \omega_2)}$ be an extended assertion which is satisfied by each (p_1, π_1) such that $\ell :: p_1 \times p_2, \pi_1 \cup \pi_2 \models A$ for each $\pi_2 \models \omega_2$. The decomposition of rule (2) in example 31 can be adapted by exploiting the following property of LTS transitions. Since $(p_2, \pi_2) \overset{\sigma:\epsilon}{\mapsto} (p_2', \pi_2)$ with $p_2' = \ell_1 :: t$, $\pi_2' = \{loc(t) := 0\}$ and $\sigma = \mathcal{C}(a)$, for every $\pi_1 \cup \pi_2 \models \omega$, such that $\alpha \in \alpha_1 \otimes \alpha_2$ and $\alpha_2 = (\sigma : \epsilon)$, we have

$$(\ell :: p_1 \times p_2, \pi_1 \cup \pi_2) \overset{\alpha}{\mapsto} (\ell :: p_1' \times p_2', \pi_1' \cup \pi_2') \Longleftrightarrow (p_1, \pi_1) \overset{\alpha_1}{\mapsto} (p_1', \pi_1').$$

Due to the complex semantics of parallel composition additional conditions as to be checked to ensure a correct decomposition. Let $\omega_1 \equiv pro_{C \cup loc(p_1)}(\omega)$ and $\omega_2 \equiv loc(t) = 0$.

$$\cfrac{\ell :: p_1 \times p_2, \omega \vdash < \alpha > A}{\cfrac{p_1, \omega_1 \vdash < \alpha_1 > (A)_{\ell \times (p, \omega_2 - 1)}}{\cfrac{\cdots}{\cfrac{\cfrac{p_1', \omega_1' \vdash (A)_{\ell \times (p, \omega_2 - 1)}}{\ell :: p_1' \times p_2', \omega_1' \wedge (\omega_2 - 1) \vdash A}\ \text{shift}}{\cdots}}}} \qquad \begin{array}{l} \models \forall(\omega_2 \supset m \leq \ell_1 \leq M) \\ \alpha \in \alpha_1 \otimes \alpha_2 \end{array}$$

The side condition $\models \forall(\omega_2 \supset m \leq \ell_1 \leq M)$ has to be checked to ensure process p_2 to be enabled to perform the timed action.

In the previous example an extended assertion is properly used to handle parallel composition by remembering the residual process resulting from component p_2 after the execution of the corresponding action. Whenever p_2 is not a basic process it is necessary (as in [9]) to give rules depending on structure of p_2 such as

$$\frac{\ell :: p_1 \times p_2, \omega \vdash < \alpha > A}{\{\ell :: p_1 \times p_{2,j}, \omega_j \vdash < \alpha_j > A_j\}_{j \in \{1,n\}}}$$

where $p_{2,j}$ are components of p_2 and ω_j, A_j are obtained from ω and A respectively.

Example 33 Suppose that $p_2 = \ell_1 :: \ell_2 :: b_1 + b_2$ and that $\ell :: p_1 \times p_2, \omega \models < \alpha > A$ for an action α such that $type(\alpha) = \epsilon$. One might think of deriving a proof of $\ell :: p_1 \times p_2, \omega \vdash < \alpha > A$ from a proof of $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > A$ for some $i \in \{1, 2\}$. However, this kind of deduction is not correct in general because of the special semantics of choice. Suppose that $(\ell_1 :: \ell_2 :: b_1, \pi_2) \overset{\alpha_2}{\mapsto} (p'_2, \pi'_2)$ and that $(p_1, \pi_1) \overset{\alpha_1}{\mapsto} (p'_1, \pi'_1)$ such that $\alpha \in \alpha_1 \otimes \alpha_2$. We have $(\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1, \pi_1 \cup \pi_2) \overset{\alpha}{\mapsto} (\ell :: p'_1 \times p'_2, \pi'_1 \cup \pi'_2)$. Therefore, if $\ell :: p'_1 \times p'_2, \pi'_1 \cup \pi'_2 \models A$ then $(\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1, \pi_1 \cup \pi_2) \models < \alpha > A$. We would thus derive $\ell :: p_1 \times (\ell_1 :: \ell_2 :: b_1 + b_2), \omega \vdash < \alpha > A$.

However, it may be the case that for some $\pi_1 \cup \pi_2 \models \omega$, $(\ell :: p_1 \times (\ell_1 :: \ell_2 :: b_1 + b_2), \pi_1 \cup \pi_2) \not\models < \alpha > A$, since $(\ell :: p_1 \times (\ell_1 :: \ell_2 :: b_1 + b_2), \pi_1 \cup \pi_2) \overset{\alpha}{\not\mapsto}$. In particular, this situation arises whenever $(\ell_1 :: \ell_2 :: b_2, \pi_2) \overset{\alpha_2}{\not\mapsto}$ with $type(\alpha_2) = \bar{\epsilon}$, since by rule $choice_2$ the two alternative processes have to agree on the idle action α_2.

Therefore, the suggested decomposition of process $\ell :: p_1 \times (\ell_1 :: \ell_2 :: b_1 + b_2)$ can be correctly applied only whenever the type of the action of process p_2 is non-idle. Suppose α to be an action with $type(\alpha) = \epsilon$. For $\alpha \in \alpha_1 \otimes \alpha_2$ both $type(\alpha_2) = \epsilon$ and $type(\alpha_2) = \bar{\epsilon}$ are possible, since $\epsilon \otimes \bar{\epsilon} = \epsilon$, $\bar{\epsilon} \otimes \epsilon = \epsilon$ and $\epsilon \otimes \epsilon = \epsilon$. No information on which component is moving is available in action α.

To solve the problem it is necessary to add more information in actions attached to next modalities. We will adopt *extended actions* allowing us to look at the type of the two parallel components. With this additional information the proof can be properly decomposed by requiring the type of the second component to be non-idle. Let $\alpha = \sigma : \epsilon$.

$$\cfrac{\ell :: p_1 \times (\ell_1 :: \ell_2 :: b_1 + b_2), \omega \vdash < \sigma : \epsilon > A}{\cfrac{\ell :: p_1 \times (\ell_1 :: \ell_2 :: b_1 + b_2), \omega \vdash < \sigma : \epsilon \times \epsilon > A}{\cfrac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1, \omega \vdash < \sigma : \epsilon \times \epsilon > A}{\ldots} \; type(proj_2(\alpha)) = \epsilon}}$$

where $type(proj_2(\alpha))$ denotes the type of the second component of the extended action $\sigma : \epsilon \times \epsilon$. Note that in the first step a type for the parallel composition $\epsilon \times \epsilon$ consistent with the result ϵ ($\epsilon \otimes \epsilon = \epsilon$) has to be guessed before decomposition.

Remark 34 It is worth mentioning that the proof system, as well as the one of [9], is not completely compositional: the rules to prove $p_1 \times p_2 \vdash < \alpha > A$ depend on the structure of p_2. As a consequence, whenever for instance p_2 is replaced by another process, it may be not clear which parts of the proof are affected. Let us show some cases:

1. Suppose for instance to replace process p_2 of example 31 with $a.p''_2$. Only the proof of $p'_1 \times p''_2 \vdash A$ has to be done;

2. Suppose for instance to replace process p_2 of example 31 with $a.p'_2 + b.p'_2$. The proof for the modified process requires only one step more to decompose $a.p'_2 + b.p'_2$:

$$\cfrac{p_1 \times (a.p'_2 + b.p'_2) \vdash < a > A}{\cfrac{p_1 \times a.p'_2 \vdash < a > A}{\ldots}}$$

Therefore, only minor modifications in the proof may be necessary whenever a part of the system is modified. Another advantage, with respect to the standard global approach, is that the underlying construction of LTS transitions may limited.

Useless cases can actually be ruled out by exploiting the structure both of the formula and of the process. Suppose for instance process p_2 of example 31 to be $b.p_2'$ and to be interested in proving $p_1 \times p_2 \vdash [a]A$. Since process p_2 does not agree on action a the decomposition rule (2) for parallel composition does not apply. Therefore, the property can be directly proved without considering at all the transitions of p_1.

8 The Proof System

In this section we formally present the complete set of proof rules. We consider proofs statements such as $p, \omega \vdash A$, where p is a closed $\mathcal{TSP}$ process and $\omega \in \Omega(loc(p)) \cup C)$ is a clock constraint on the locations of p and on the specification clocks C, such that for each basic constraint $x + m\mathcal{R}y + m$ of ω, clocks $x, y \in C$. Formula A is a closed formula of RTL_μ eventually with tagged fix-points, extended assertions and extended actions as explained in section 7.2.

Extended actions differ from LTS labels since the type of the two parallel components may be specified.

Definition 35 Let Π be a set of primitive events. We define $\mathcal{E}(\Pi)$ as the set of *extended actions* $\sigma : \tau$, where $\sigma \in \Sigma(\Pi)$ and τ is an *extended* type. Basic extended types are ϵ and $\bar{\epsilon}$ and, if τ_1 is an extended type and τ_2 is a basic extended type, then $\tau_1 \times \tau_2$ is an extended type.

For an extended type τ, let $type(\tau) = \tau$ if τ is a basic type and $type(\tau_1 \times \tau_2) = type(\tau_1) \otimes \tau_2$ otherwise. Intuitively, $type(\sigma : \tau)$ says if the whole action is idle or non-idle according to the types of the parallel components.

For $\sigma : \tau \in \mathcal{E}(\Pi)$, let $type(\sigma : \tau) = type(\tau)$ and for $\alpha \in \mathcal{E}(\Pi)$, let $proj_i(\alpha) = \sigma : \tau_i$ with $i \in \{1, 2\}$ if $\alpha = \sigma : \tau_1 \times \tau_2$.

The operation $\otimes$ of parallel composition of LTS labels can simply be adapted to extended actions. For $\sigma_1 : \tau_1 \in \mathcal{E}(\Pi)$, $\sigma_2 : \tau_2 \in LTS(\Pi)$, we define $(\sigma_1 : \tau_1) \otimes (\sigma_2 : \tau_2) = \{\sigma : \tau_1 \times \tau_2 \mid \sigma \in \sigma_1 \otimes \sigma_2\}$.

Definition 36 Let C be a set of specification clocks, VAR a set of variables, Π be a set of primitive events and Loc be a set of locations such that $Loc \cap C = \emptyset$. Formulas are inductively defined as follows.

$$A ::= X \mid \ell \mid \neg\ell \mid A_1 \vee A_2 \mid A_1 \wedge A_2 \mid \omega \mid < \alpha > A \mid [\alpha]A \mid \mu X\{U\}.A \mid \nu X\{U\}.A \mid E$$

$$E ::= (A)_{(\ell::t,\omega)\bigtriangledown} \mid (A)_{\ell\times(p,\omega)} \mid (A)_{\times+b} \mid (A)_{\times(\ell::t,\omega)\bigtriangledown} \mid (A)_{\times\bigtriangledown(p,\omega)} \mid (A)_{\times}.$$

for $\ell \in Loc$, $X \in VAR$, $\alpha \in \mathcal{E}(\Pi)$, $\omega \in \Omega(C)$, $U \subseteq \{(p,\omega) \mid p \in \mathcal{TSP}, \omega \in \Omega(C \cup loc(p))\}$.

In the following the standard formulas $\mu X.A$ and $\nu X.A$ stand for $\mu X\{\emptyset\}.A$ and $\nu X\{\emptyset\}.A$ respectively.

8.1 Logical Rules

Logical rules work on the structure of formulas for each logical connective except next modalities.

$$\frac{p,\omega \vdash P}{}\,prop_1 \quad P \in loc(p) \qquad\qquad \frac{p,\omega \vdash \neg P}{}\,prop_2 \quad P \notin loc(p)$$

$$\frac{p,\omega \vdash A_1 \wedge A_2}{p,\omega \vdash A_1 \quad p,\omega \vdash A_2}\wedge \qquad \frac{p,\omega \vdash A_1 \vee A_2}{p,\omega \vdash A_i}\vee$$

$$\frac{p,\omega_1 \vdash A}{p,\omega_2 \vdash A}\,thin \quad \models \forall(\omega_1 \supset \omega_2) \qquad\qquad \frac{p,\omega_1 \vee \omega_2 \vdash A}{p,\omega_1 \vdash A \quad p,\omega_1 \vdash A}\,disj$$

$$\frac{p,\omega_1 \vdash \omega_2}{}\,taut \quad \models \forall(\omega_1 \supset \omega_2) \qquad\qquad \frac{p,\omega \vdash A}{}\,empty \quad \models \forall(\omega \wedge delay(p) \supset false)$$

$$\frac{p,\omega \vdash x.A}{p, reset(\omega, (loc(p) \cup C) \setminus \{x\}, \{x\}) \vdash A}\,reset$$

Basic logical rules are quite simple. In rule *empty* the idea that timed states (p,π) are admissible iff the maximal delay of p is satisfied by π. Rule *reset* is based on the semantics of quantifier $x.$ and exploits the property that $\pi \models \omega$ iff $\pi[x := 0] \models reset(\omega, (loc(p) \cup C) \setminus \{x\}, \{x\})$.

8.2 Fix-points

For fix-points formulas the tagged fix-points method [49] is properly extended to timed states as explained in section 7.1. Let $\{(p,\omega) \mid (p,\omega) \in U\} = \{(p,\omega_i)\}_{i\in\{1,n\}}$.

$$\frac{p,\omega \vdash \nu X\{U\}.A}{}\,\nu_1 \qquad \models (\omega \supset \bigvee_{i\in\{1,n\}} \omega_i)$$

$$\frac{p,\omega \vdash \nu X\{U\}.A}{p,\omega \vdash A[\nu X\{U \cup \{(p,\omega)\}\}.A/X]}\,\nu_2 \quad \forall i \in \{1,n\} \models \forall(\omega \wedge \omega_i \supset false)$$

$$\frac{p,\omega \vdash \mu X\{U\}.A}{p,\omega \vdash A[\mu X\{U \cup \{(p,\omega)\}\}.A/X]}\,\mu \quad \forall i \in \{1,n\} \models \forall(\omega \wedge \omega_i \supset false)$$

Rules ν_2 and μ realize the unfolding of the fix-points formula where the tag is properly updated. The side condition guarantees the unfolding rule to be applied only once for every timed state. Rule ν_1 gives the termination condition for greatest fix-points.

8.3 Shift Rules

For each extended assertion the corresponding shift rule builds up the whole process.

$$\frac{p, \omega_1 \vdash (A)_{(\ell::t,\omega_2)\bigtriangledown}}{\ell :: t \bigtriangledown p, \omega_1 \wedge \omega_2 \vdash A} \qquad \frac{p_1, \omega_1 \vdash (A)_{\ell \times (p_2,\omega_2)}}{\ell :: p_1 \times p_2, \omega_1 \wedge \omega_2 \vdash A}$$

$$\frac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1, \omega_1 \vdash (A)_{\times + b}}{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b, \omega_1 \vdash A} \qquad \frac{\ell :: p_1 \times p_2, \omega_1 \vdash (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}}{\ell :: p_1 \times (\ell_1 :: t \bigtriangledown p_2), \omega_1 \wedge \omega_2 \vdash A}$$

$$\frac{\ell :: p_2 \times \ell_1 :: t, \omega_1 \vdash (A)_{\times \bigtriangledown (p_1,\omega_2)}}{\ell :: p_2 \times (\ell_1 :: t \bigtriangledown p_1), \omega_1 \wedge \omega_2 \vdash A} \qquad \frac{\ell :: \ell_1 :: (p_1 \times p_2) \times p_3, \omega_2 \vdash (A)_{\times}}{\ell :: p_1 \times \ell_1 :: (p_2 \times p_3), \omega_1 \wedge \omega_2 \vdash A}$$

8.4 Compositional Rules

We present the rules to decompose the proofs for the next modalities depending on the process structure which have been informally discussed in section 7.2.

8.4.1 Rules for *nil*

$$\frac{\ell_1 :: \ell_2 :: nil, \omega_1 \vdash < \sigma_0 : \bar{\epsilon} > A}{\ell_1 :: \ell_2 :: nil, \omega_1 - 1 \vdash A} <> nil$$

$$\frac{\ell_1 :: \ell_2 :: nil, \omega_1 \vdash [\sigma : \epsilon] A}{\quad} []nil_1$$

$$\frac{\ell_1 :: \ell_2 :: nil, \omega_1 \vdash [\sigma_0 : \bar{\epsilon}] A}{\ell_1 :: \ell_2 :: nil, \omega_1 - 1 \vdash A} []nil_2$$

8.4.2 Rules for timed prefixing

Rules for prefixing are based on rules $idle$ and $non - idle$ of the LTS. Remind that timed actions ($\#(a) = \emptyset$) can be arbitrary delayed until the maximal delay, while immediate actions have to executed whenever the environments produces the required input. Let $\omega_2 \equiv reset(\omega_1 - 1, C, loc(t))$.

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash < \sigma : \epsilon > A}{\ell :: t, \omega_2 \vdash A} <> \epsilon \qquad \begin{array}{l} \models \forall(\omega_1 \supset m \leq \ell_1 \leq M) \\ \sigma = \mathcal{C}(a) \end{array}$$

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash < \sigma : \bar{\epsilon} > A}{\ell :: \ell_1 :: [m,M]a.t, \omega_1 - 1 \vdash A} <> \bar{\epsilon} \qquad \begin{array}{l} (\sigma = \sigma_0 \wedge \#(a) = \emptyset) \vee (\sigma \in \#(a)) \\ \models \forall(\omega_1 - 1 \supset \ell_1 \leq M) \end{array}$$

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash [\sigma : \epsilon]A}{\ell :: t, \omega_2 \vdash A} []\epsilon_1 \qquad \begin{array}{l} \models \forall(\omega_1 \supset m \leq \ell_1 \leq M) \\ \sigma = \mathcal{C}(a) \end{array}$$

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash [\sigma : \bar{\epsilon}]A}{\ell :: \ell_1 :: [m,M]a.t, \omega_1 - 1 \vdash A} []\bar{\epsilon}_1 \qquad (\sigma = \sigma_0 \wedge \#(a) = \emptyset) \vee \sigma \in \#(a)$$

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash [\sigma : \epsilon]A}{} []\epsilon_2 \qquad \sigma \neq \mathcal{C}(a)$$

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash [\sigma : \bar{\epsilon}]A}{} []\bar{\epsilon}_2 \qquad (\sigma \notin \#(a) \wedge \#(a) \neq \emptyset)$$

$$\frac{\ell :: \ell_1 :: [m,M]a.t, \omega_1 \vdash [\sigma : \epsilon]A}{} []\epsilon_3 \qquad \models \forall(\omega_1 \wedge m \leq \ell_1 \leq M \supset false)$$

The side conditions are due to the corresponding side conditions of rules *idle* and $non - idle$. Constraint ω_2 is obtained by resetting the locations of t and by letting time to proceed of one unit for the specification clocks C.

8.4.3 Rules for choice

There are two sets of rules: rules applicable when the type of the action is non-idle, and rules applicable when the type of the action is idle. This because it is necessary distinguish whether rule $choice_1$ or rule $choice_2$ is applicable.

1. Rules applicable iff $type(\alpha) = \epsilon$.

$$\frac{\ell :: \ell_1 :: b_1 + b_2, \omega_1 \vdash < \alpha > A}{\ell :: \ell_1 :: b_i, \omega_1 \vdash < \alpha > A} + <> \epsilon$$

$$\frac{\ell :: \ell_1 :: b_1 + b_2, \omega_1 \vdash [\alpha]A}{\{\ell :: \ell_1 :: b_i, \omega_1 \vdash [\alpha]A\}_{i \in \{1,2\}}} +[]\epsilon$$

2. Rules applicable iff $type(\alpha) = \bar{\epsilon}$.

$$\frac{\ell :: \ell_1 :: b_1 + b_2, \omega_1 \vdash < \alpha > A}{\{\ell :: \ell_1 :: b_i, \omega_1 \vdash < \alpha > true\}_{i \in \{1,2\}} \quad \ell :: \ell_1 :: b_1 + b_2, \omega_1 - 1 \vdash A} + <> \bar{\epsilon}$$

$$\frac{\ell :: \ell_1 :: b_1 + b_2, \omega_1 \vdash [\alpha]A}{\ell :: \ell_1 :: b_1 + b_2, \omega_1 - 1 \vdash A} +[]\bar{\epsilon}_1$$

$$\frac{\ell :: \ell_1 :: b_1 + b_2, \omega_1 \vdash [\alpha]A}{\ell :: \ell_1 :: b_i, \omega_1 \vdash [\alpha]false} +[]\bar{\epsilon}_2$$

Rules for non-idle actions simply derive from rule $choice_1$ of the operational semantics. A non-idle action is a non-idle action of one of the two processes. By contrast, by rule $choice_2$ the two processes must agree on idle actions.

8.4.4 Rules for refinement

Rules for refinement are similar to the ones for choice except that for the special case of hierarchy modelled by rule $\bigtriangledown_2$. Let $\omega_2 \equiv pro_{loc(\ell::t)}(\omega_1) - 1$, $\omega_p \equiv pro_{C \cup loc(p)}(\omega_1)$ and $\omega_t \equiv pro_{C \cup loc(\ell::t)}(\omega_1)$.

1. Rules applicable iff $type(\alpha) = \epsilon$.

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash < \alpha > A}{\ell :: t, \omega_t \vdash < \alpha > A} \bigtriangledown <> \epsilon_1$$

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: t, \omega_t \vdash [\alpha]A \quad p, \omega_p \vdash [\alpha](A)_{(\ell::t,\omega_2)\bigtriangledown}} \bigtriangledown[]\epsilon_1 \qquad \models \forall(\omega_2 \supset delay(\ell :: t))$$

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: t, \omega_t \vdash [\alpha]A} \bigtriangledown[]\epsilon_2 \qquad \models \forall(\omega_2 \wedge delay(\ell :: t) \supset false)$$

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash < \alpha > A}{p, \omega_p \vdash < \alpha > (A)_{(\ell::t,\omega_2)\bigtriangledown}} \bigtriangledown <> \epsilon_2 \qquad \models \forall(\omega_2 \supset delay(\ell :: t))$$

2. Rules applicable iff $type(\alpha) = \bar{\epsilon}$.

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash < \alpha > A}{\ell :: t, \omega_t \vdash < \alpha > true \quad p, \omega_p \vdash < \alpha > true \quad \ell :: t \bigtriangledown p, \omega_1 - 1 \vdash A} \bigtriangledown <> \bar{\epsilon}$$

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: t \bigtriangledown p, \omega_1 - 1 \vdash A} \bigtriangledown[]\bar{\epsilon}_1 \qquad \frac{\ell :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: t, \omega_t \vdash [\alpha]false} \bigtriangledown[]\bar{\epsilon}_2$$

$$\frac{\ell :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{p, \omega_p \vdash [\alpha]false} \bigtriangledown[]\bar{\epsilon}_3$$

Rules for non-idle actions derive from rules $\bigtriangledown_1$ and $\bigtriangledown_2$ of the operational semantics. In order to model the hierarchy of rule $\bigtriangledown_2$ extended assertions $(A)_{(\ell::t,\omega_2)\bigtriangledown}$ are used in rules $\bigtriangledown <> \epsilon_2$, $\bigtriangledown[]\epsilon_1$ and $\bigtriangledown[]\epsilon_2$. Note that in the clock constraint ω_2 time has advanced of one unit for $\ell :: t$. The side conditions on time are due to the semantics of refinement: actions of process p can be performed only when the maximal delay of $\ell :: t$ has not been reached.

The rules for idle actions derives from rule $\bigtriangledown_3$ of the operational semantics and are analogous to the one of choice. Note that the projections of ω_1 with respect to $loc(p)$ and $loc(\ell :: t)$ have been considered in the decomposition, since $loc(\ell :: t \bigtriangledown p) = loc(\ell :: t) \cup loc(p)$ such that $loc(p) \cap loc(\ell :: t) = \emptyset$.

8.4.5 Rules for recursion

$$\frac{\ell :: rec\ x.t, \omega_1 \vdash \sigma A}{\ell :: t[rec\ x.t/x], \omega_1 \vdash \sigma A}\, rec \quad \sigma \in \{<\alpha>, [\alpha]\}$$

8.4.6 Rules for parallel composition

Rules for parallel composition depend on the syntax of the second component as we have discussed in examples 32 and 33. Because of the problems shown in example 33 extended actions have to be introduced. For a label $\sigma : \tau \in LTS(\Pi)$, we define $\mathcal{E}(\sigma : \tau) = \{\sigma : \tau_1 \times \tau_2 \in \mathcal{E}(\Pi) \mid \tau_1$ and τ_2 are basic extended types and $type(\tau_1 \times \tau_2) = \tau\}$.

The following rules realize the expansion of a basic type into a consistent type for the parallel composition.

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash <\alpha> A}{\ell :: p_1 \times p_2, \omega_1 \vdash <\beta> A}\, \mathcal{E}<> \quad \beta \in \mathcal{E}(\alpha)$$

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash [\alpha] A}{\{\ell :: p_1 \times p_2, \omega_1 \vdash [\beta_i] A\}_{i \in \{1,n\}}}\, \mathcal{E}[] \quad \{\beta_i\}_{i \in \{1,n\}} = \mathcal{E}(\alpha)$$

8.4.7 Rules for *nil*

These rules are based on the idea that a *nil* process can perform only idle action $\sigma_0 : \bar{\epsilon}$. Let $p_2 = \ell_1 :: \ell_2 :: nil$ and $\omega_2 \equiv pro_{\{\ell_2\}}(\omega_1) - 1$, $\omega_{p_1} \equiv pro_{C \cup loc(p_1)}(\omega_1)$ and $\{\beta_i\}_{i \in \{1,k\}} = \{\beta \mid \alpha \in \beta \otimes (\sigma_0 : \bar{\epsilon})\}$.

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash <\alpha> A}{p_1, \omega_{p_1} \vdash <\beta_i> (A)_{\ell \times (p_2, \omega_2)}}\, \times <> nil \quad i \in \{1,n\}$$

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash [\alpha] A}{\{p_1, \omega_{p_1} \vdash [\beta_i](A)_{\ell \times (p_2, \omega_2)}\}_{i \in \{1,k\}}}\, \times [] nil$$

8.4.8 Rules for prefixing

These rules are based on the decomposition shown in example 32. The idea that if a is an immediate action then p_2 either performs non-idle action a or it performs an idle action in conflict with a. The side conditions guarantees that the timing conditions for the enabling of a and for the delay of a whenever a is a timed action are satisfied. Let $p_2 = \ell_1 :: \ell_2 :: [m, M]a.t$, $\omega_2 \equiv loc(t) = 0$, $\omega_3 \equiv pro_{loc(p_2)}(\omega_1) - 1$ and $\omega_{p_1} \equiv pro_{C \cup loc(p_1)}(\omega_1)$. Let $\{\alpha_i\}_{i \in \{1,n\}} = \{\alpha_i \mid \alpha \in \alpha_i \otimes (\mathcal{C}(a) : \epsilon)\}$ and $\{\beta_i\}_{i \in \{1,k\}} = \{\beta_i \mid \alpha \in \beta_i \otimes (\sigma : \bar{\epsilon}), (\sigma = \sigma_0 \wedge \#(a) = \emptyset) \vee b \in \#(a)\}$.

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash < \alpha > A}{p_1, \omega_{p_1} \vdash < \beta > (A)_{\ell \times (\ell_1 :: t, \omega_2)}} \times <>_1 \qquad \begin{array}{l} \alpha \in \beta \otimes (\mathcal{C}(a) : \epsilon) \\ \models \forall (\omega_1 \supset m \leq \ell_2 \leq M) \end{array}$$

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash < \alpha > A}{p_1, \omega_{p_1} \vdash < \beta > (A)_{\ell \times (p_2, \omega_3)}} \times <>_2 \qquad \begin{array}{l} \alpha \in \beta \otimes (\sigma : \bar{\epsilon}) \\ (\sigma = \sigma_0 \wedge \#(a) = \emptyset) \vee \sigma \in \#(a) \\ \models \forall (\omega_3 \supset \ell_2 \leq M) \end{array}$$

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash [\alpha] A}{\begin{array}{l} \{p_1, \omega_{p_1} \vdash [\alpha_i](A)_{\ell \times (\ell_1 :: t, \omega_2)}\}_{i \in \{1,n\}} \\ \{p_1, \omega_{p_1} \vdash [\beta_i](A)_{\ell \times (p_2, \omega_3)}\}_{i \in \{1,k\}} \end{array}} \times []_1 \qquad \models \forall (\omega_1 \supset m \leq \ell \leq M)$$

$$\frac{\ell :: p_1 \times p_2, \omega_1 \vdash [\alpha] A}{\{p_1, \omega_{p_1} \vdash [\beta_i](A)_{\ell \times (p_2, \omega_3)}\}_{i \in \{1,k\}}} \times []_2 \qquad \models \forall (\omega_1 \wedge m \leq \ell \leq M \supset false)$$

8.4.9 Rules for choice

The proof rules for choice depend on the type of the action corresponding to the second component in order to handle the problems shown in example 33. If the type of the action corresponding to the choice process is non-idle, then the standard decomposition applies. By contrast, for idle actions the two processes must agree. Therefore, extended assertions $(A)_{\times + b_j}$ are used.

1. Rules applicable iff $type(proj_2(\alpha)) = \epsilon$.

$$\frac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2, \omega_1 \vdash < \alpha > A}{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega_1 \vdash < \alpha > A} \times <> +\epsilon$$

$$\frac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2, \omega_1 \vdash [\alpha] A}{\{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega_1 \vdash [\alpha] A\}_{i \in \{1,2\}}} \times [] + \epsilon$$

2. Rules applicable iff $type(proj_2(\alpha)) = \bar{\epsilon}$.

$$\frac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2, \omega_1 \vdash < \alpha > A}{\{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega_1 \vdash < \alpha > (A)_{\times + b_j}\}_{i,j \in \{1,2\}, i \neq j}} \times <> +\bar{\epsilon}$$

$$\frac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2, \omega_1 \vdash [\alpha] A}{\{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega_1 \vdash [\alpha](A)_{\times + b_j}\}_{i,j \in \{1,2\}, i \neq j}} \times [] + \bar{\epsilon}_1$$

$$\frac{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2, \omega_1 \vdash [\alpha] A}{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega_1 \vdash [\alpha] false} \times [] + \bar{\epsilon}_2$$

8.4.10 Rules for refinement

The proof rules for refinement as well as the one of choice require to distinguish idle and non-idle actions of the second component. The use of extended actions allows to apply similar techniques. Let $\omega_p \equiv pro_{C\cup(loc(\ell::p_1\times p))}(\omega_1)$, $\omega_t \equiv pro_{C\cup loc(\ell::p_1\times\ell_1::t)}(\omega_1)$, $\omega_2 \equiv pro_{loc(\ell_1::t)}(\omega_1)-1$ and $\omega_3 \equiv pro_{loc(p)}(\omega_1) - 1$.

1. Rules applicable iff $type(proj_2(\alpha)) = \epsilon$.

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash < \alpha > A}{\ell :: p_1 \times \ell_1 :: t, \omega_t \vdash < \alpha > A} \times <> \bigtriangledown\epsilon_1$$

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash < \alpha > A}{\ell :: p_1 \times p, \omega_p \vdash < \alpha > (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}} \times <> \bigtriangledown\epsilon_2 \quad \models \forall(\omega_2 \supset delay(\ell_1 :: t))$$

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: p_1 \times p, \omega_p \vdash [\alpha](A)_{\times(\ell_1::t,\omega_2)\bigtriangledown} \quad \ell :: p_1 \times \ell_1 :: t, \omega_t \vdash [\alpha]A} \times [] \bigtriangledown \epsilon_2$$

2. Rules applicable iff $type(proj_2(\alpha)) = \bar{\epsilon}$.

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash < \alpha > A}{\begin{array}{l}\ell :: p_1 \times \ell_1 :: t, \omega_t \vdash < \alpha > (A)_{\times\bigtriangledown(p,\omega_3)} \\ \ell :: p_1 \times p, \omega_p \vdash < \alpha > (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}\end{array}} \times <> \bigtriangledown\bar{\epsilon}$$

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\begin{array}{l}\ell :: p_1 \times \ell_1 :: t, \omega_t \vdash [\alpha](A)_{\times\bigtriangledown(p,\omega_3)} \\ \ell :: p_1 \times p, \omega_p \vdash [\alpha](A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}\end{array}} \times [] \bigtriangledown \bar{\epsilon}_1$$

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: p_1 \times \ell_1 :: t, \omega_t \vdash [\alpha]false} \times [] \bigtriangledown \bar{\epsilon}_2$$

$$\frac{\ell :: p_1 \times \ell_1 :: t \bigtriangledown p, \omega_1 \vdash [\alpha]A}{\ell :: p_1 \times p, \omega_p \vdash [\alpha]false} \times [] \bigtriangledown \bar{\epsilon}_3$$

8.4.11 Rules for recursion

$$\frac{\ell :: p_1 \times \ell_2 :: rec\ x.t, \omega_1 \vdash \sigma A}{\ell :: p_1 \times \ell_2 :: t[rec\ x.t/x], \omega_1 \vdash \sigma A} \times rec \quad \sigma \in \{< \alpha >, [\alpha]\}$$

8.4.12 Rules for parallel composition

The basic idea for parallel composition is that it is sufficient to associate on the left by the associative property of parallel composition. Types have to be arranged according to the shift of processes. For $\alpha \in \mathcal{E}(\Pi)$, $left(\alpha) = \{\sigma : \tau$ such that $\tau = (\tau_1 \times \tau_{2,1}) \times \tau_{2,2}$ and $type(\tau_{2,1} \times \tau_{2,2}) = \tau_2$, whenever $type(proj_i(\alpha)) = \tau_i$ for each $i \in \{1, 2\}\}$.

$$[\![(A)_{(\ell::t,\omega)\bigtriangledown}]\!]_{\rho,\delta} = \{(p,\pi) \mid (\ell :: t \bigtriangledown p, \pi \cup \pi') \in [\![A]\!]_{\rho,\delta}, \forall \pi' \models \omega\}$$

$$[\![(A)_{\ell\times(p,\omega)}]\!]_{\rho,\delta} = \{(p',\pi') \mid (\ell :: p' \times p, \pi \cup \pi') \in [\![A]\!]_{\rho,\delta}, \forall \pi \models \omega\}$$

$$[\![(A)_{\times+b}]\!]_{\rho,\delta} = \{(\ell :: p_1 \times \ell_1 :: \ell_2 :: b', \pi) \mid (\ell :: p_1 \times \ell_1 :: \ell_2 :: b' + b, \pi) \in [\![A]\!]_{\rho,\delta}\}$$

$$[\![(A)_{\times(\ell::t,\omega)\bigtriangledown}]\!] = \{(\ell_1 :: p_1 \times p_2, \pi) \mid (\ell_1 :: p_1 \times (\ell :: t \bigtriangledown p), \pi \cup \pi') \in [\![A]\!]_{\rho,\delta}, \forall \pi' \models \omega\}$$

$$[\![(A)_{\times\bigtriangledown(p,\omega)}]\!]_{\rho,\delta} = \{(\ell_1 :: p_1 \times \ell :: t, \pi) \mid (\ell_1 :: p_1 \times (\ell :: t \bigtriangledown p), \pi \cup \pi') \in [\![A]\!]_{\rho,\delta}, \forall \pi' \models \omega\}$$

$$[\![(A)_{\times}]\!]_{\rho,\delta} = \{(\ell_1 :: \ell_2 :: (p_1 \times p_2) \times p_3, \pi) \mid (\ell_1 :: p_1 \times \ell_2 :: (p_2 \times p_3)) \in [\![A]\!]_{\rho,\delta}\}.$$

Table 4: Extended Formulas

$$\frac{\ell :: p_1 \times \ell_2 :: (p_3 \times p_4), \omega_1 \vdash < \alpha > A}{\ell :: \ell_2 :: (p_1 \times p_3) \times p_4, \omega_1 \vdash < \beta > (A)_\times} \times\times <> \quad \beta \in left(\alpha)$$

$$\frac{\ell :: p_1 \times \ell_2 :: (p_3 \times p_4), \omega_1 \vdash [\alpha]A}{\{\ell :: \ell_1 :: (p_1 \times p_3) \times p_4, \omega_1 \vdash [\beta_i](A)_\times\}_{i\in\{1,n\}}} \times\times [] \quad left(\alpha) = \{\beta_i\}_{i\in\{1,n\}}$$

9 Soundness and Completeness

The proof system is sound in general, while it is complete for the class of *regular* processes including the ones corresponding to timed statecharts. Before introducing the main theorems the semantics of the new formulas has to be formally defined. In the following let p be a $\mathcal{TSP}$ closed process and $LTS(p) = (TS, ts_0, LTS(\Pi), \mapsto)$. We omit the reference to the corresponding model $\mathcal{M}(p)$ (see section 6).

The semantics of tagged fix-points formulas is the straightforward generalization of the standard one [49]. For a tag U, let $[\![U]\!] = \{(p,\pi) \in TS \mid (p,\omega) \in U, \pi \models \omega\}$. For a valuation for propositional symbols ρ and for a clock assignment δ we have

$$[\![\mu X\{U\}.A]\!]_{\rho,\delta} = \mu V.([\![A]\!]_{\rho[V/X],\delta} \setminus [\![U]\!])$$

$$[\![\nu X\{U\}.A]\!]_{\rho,\delta} = \nu V.([\![A]\!]_{\rho[V/X],\delta} \cup [\![U]\!])$$

The semantics of extended assertions is defined in table 4 for a valuation for propositional symbols ρ and for a clock assignment δ.

It remains to define the semantics of next modalities over extended actions. Intuitively, $(p,\pi) \models < \alpha > A$ with $\alpha = \sigma : \tau_1 \times \tau_2$ iff there exists a transition $(p,\pi) \overset{\alpha'}{\mapsto} (p',\pi')$, such that $(p',\pi') \models A$, $\alpha' = \sigma : \tau$ and types τ and $\tau_1 \times \tau_2$ agree. For instance, if $\tau_1 = \epsilon$ and $\tau_2 = \bar{\epsilon}$, then transition $(p,\pi) \overset{\sigma:\epsilon}{\mapsto} (p',\pi')$ is consistent iff it has been obtained from rule $\times$ applied to premises $\{(p_i,\pi_i') \overset{\alpha_i'}{\mapsto} (p_i',\pi_i')\}_{i\in\{1,2\}}$, where $type(\alpha_1') = \epsilon$ and $type(\alpha_2') = \bar{\epsilon}$.

Definition 37 Let Π be a set of primitive events, let $\alpha = \sigma : \tau \in \mathcal{E}(\Pi)$ and $\alpha' = \sigma : \tau' \in LTS(\Pi)$. A transition $(p, \pi) \overset{\alpha'}{\mapsto} (p', \pi')$ is *compatible* with α iff either $\tau = \tau'$ or there exist $\{(p_i, \pi_i) \overset{\alpha'_i}{\mapsto} (p'_i, \pi'_i)\}_{i\in\{1,2\}}$, such that $p' = p'_1 \times p'_2$, $\pi' = \pi'_1 \cup \pi'_2$, $\alpha' \in \alpha'_1 \otimes \alpha'_2$, and for each $i \in \{1, 2\}$ $(p_i, \pi_i) \overset{\alpha'_i}{\mapsto} (p'_i, \pi'_i)$ is compatible with $\alpha_i = \sigma_i : \tau_i$ where $\alpha'_i = \sigma_i : \tau'_i$ and $type(proj_i(\alpha)) = \tau_i$.

We adopt the notation $(p, \pi) \overset{\alpha'}{\mapsto}_\alpha (p', \pi')$ to say that transition $(p, \pi) \overset{\alpha'}{\mapsto} (p', \pi')$ is compatible with α. The semantics is formally defined as follows for a valuation for propositional symbols ρ and for a clock assignment δ

$$[\![< \alpha > A]\!]_{\rho,\delta} = \left\{ (p, \pi) \mid \exists p', \pi' \ (p, \pi) \overset{\alpha'}{\mapsto}_\alpha (p', \pi') \text{ such that } (p', \pi') \in [\![A]\!]_{\rho,\delta+1} \right\}$$

$$[\![[\alpha]A]\!]_{\rho,\delta} = \left\{ (p, \pi) \mid \forall p', \pi' \ (p, \pi) \overset{\alpha'}{\mapsto}_\alpha (p', \pi') \Longrightarrow (p', \pi') \in [\![A]\!]_{\rho,\delta+1} \right\}$$

Soundness and completeness of tagged fix-points is based on the following *reduction lemma*.

Lemma 38 Let ψ be a monotonic function on the lattice $\mathcal{P}(D)$. For $P \in \mathcal{P}(D)$,

1. $P \subseteq \psi(\mu V.(\psi(V) \setminus P)) \Rightarrow P \subseteq \mu V.(\psi(V))$;
2. $P \subseteq \psi(\nu V.(\psi(V) \cup P)) \Leftrightarrow P \subseteq \nu V.(\psi(V))$.
3. If $\mu V.\psi(V) = \psi^k(\emptyset)$ for some integer k, then there exists $\{P_i\}_{i\in\{1,n\}}$, such that $P = \bigcup_{i\in\{1,n\}} P_i$, and, for each $i \in \{1, n\}$, $P_i \subseteq \mu V.\psi(V) \Rightarrow P_i \subseteq \psi(\mu V.(\psi(V) \setminus P_i))$.

Proof of lemma 38 is shown in the appendix A.

Theorem 39 Let p be a closed $\mathcal{TSP}$ process, A be a closed formula and $\omega \in \Omega(C \cup loc(p))$. If $p, \omega \vdash A$, then $p, \omega \models A$.

Proof of theorem 39 can be found in [32] and is obtained by showing that each rule is backward sound. Soundness of fix-points formulas relies on lemma 38.

The proof of completeness for regular processes (see section 4) is by induction on a relation $\prec$ over formulas that is well-founded for regular processes because of the following properties.

Let $p \in \mathcal{TSP}$ be a closed process and $LTS(p) = (TS, ts_0, LTS(\Pi), \mapsto)$. We denote by $\mathcal{R}_p$ the minimal subset of closed processes such that $p \in \mathcal{R}_p$ and, for each $p' \in \mathcal{R}_p$ and for each $(p', \pi') \overset{\alpha}{\mapsto} (p'', \pi'')$, $p'' \in \mathcal{R}_p$. Let $\mathcal{S}_p$ be the minimal subset of closed processes such that, if p' is a sub-process of p, then $\mathcal{R}_{p'} \subseteq \mathcal{S}_p$.

For regular processes both $\mathcal{R}_p$ and $\mathcal{S}_p$ are finite. Even if $\mathcal{R}_p$ is finite for regular processes the state space to be explored can be infinite because of time. We show that there exists a finite quotient of $LTS(p)$ that is enough to prove a formula A. The idea is that of constructing equivalence classes of formulas by using a clock region technique [2, 22, 1].

Definition 40 Let p be a closed $\mathcal{TSP}$ process and let $\omega \in \Omega(loc(p) \cup C)$ and let A be a closed formula. We say that ω is a (p, A)-*constraint* iff the greatest constant of ω is k, where k is the greatest constant that occurs in p or in A. A set of timed states $\mathcal{R}$ is a (p, A)-*region* iff there exists a (p, A)-constraint ω such that $\mathcal{R} = \{(p', \pi') \mid p' \in \mathcal{R}_p$ and $\pi' \models \omega\}$.

A (p, A)-region $\mathcal{R}$ is minimal if there is no other (p, A)-region $\mathcal{R}'$ such that $\mathcal{R}' \subseteq \mathcal{R}$. The main result is that the set of states satisfying a formula A is a set of minimal (p, A)-regions. As shown in [1] there are only finitely many minimal (p, A)-constraints, so that for regular processes there are only finitely many (p, A)-regions. Therefore, it is possible to construct a finite quotient of the infinite labelled transition system that is adequate to verify whether p satisfies A.

Lemma 41 Let p be a closed $\mathcal{TSP}$ process p and let A be a closed formula such that $fv(A) = \emptyset$. We have that $[\![A]\!]$ is a set of minimal (p, A)-regions.

The proof of lemma 41 is shown in the appendix A.

By exploiting the previous properties the following ordering on formulas can be used to prove that fix-points can be computed within a finite number of steps. The ordering is actually well-founded.

Definition 42 Let p be a closed $\mathcal{TSP}$ process and let A, A' be closed formulas with tags (p', ω), where ω is a (p, A)-constraint and $p' \in \mathcal{S}_p$. We define $A' \prec A$ iff either A' is a proper sub-formula of A or for $\sigma \in \{\mu, \nu\}$ $A = \sigma X\{U\}B$ and $A' = B[\sigma X\{U \cup \{(p', \omega)\}.B/X]$ where $\omega \in \Omega(loc(p') \cup C)$ is a (p, A)-constraint, such that $[\![(p', \omega)]\!] \cap [\![U]\!] = \emptyset$ and $p' \in \mathcal{S}_p$.

Completeness for the next modalities relies on the following lemma. Intuitively, this lemma shows that a proof $p, \omega \vdash <\alpha> A$ can be obtained from premises $\{p_i, \omega_i \vdash A_i\}_{i \in \{1,n\}}$ such that $A_i \prec A$.

Lemma 43 Let p be a closed $\mathcal{TSP}$ process, A be a closed formula and $\omega \in \Omega(loc(p) \cup C)$. For $\alpha \in \mathcal{E}(\Pi)$, if $p, \omega \models <\alpha> A$ (resp. $p, \omega \models [\alpha]A$)

1. there exist sets $\{p_i \in \mathcal{S}_p\}_{i \in \{1,n\}}$ and $\{p'_j \in \mathcal{S}_p\}_{j \in \{1,k\}}$, such that $\{p_i, \omega_i \models A\}_{i \in \{1,n\}}$ and $\{p'_j, \omega'_j \models true\}_{j \in \{1,k\}}$, for constraints $\{\omega_i\}_{i \in \{1,n\}}$ and $\{\omega'_j\}_{j \in \{1,k\}}$;
2. $p, \omega \vdash <\alpha> A$ (resp. $p, \omega \vdash [\alpha]A$)) can be derived from $\{p_i, \omega_i \vdash A\}_{i \in \{1,n\}}$ and $\{p'_j, \omega'_j \vdash true\}_{j \in \{1,k\}}$.

The proof of lemma 43 can be found in the appendix A.

Theorem 44 Let p be a closed and regular $\mathcal{TSP}$ process, A be a closed formula and $\omega \in \Omega(C \cup loc(p))$. If $p, \omega \models A$, then $p, \omega \vdash A$.

The proof of theorem 44 can be found in the appendix A.

10 A Proof

In this section we show the application of the proof system for checking a simple property of the statechart of figure 3 with the following label of transitions $t_1 : [0, \infty]a/b, t_5 : [0, \infty]a/b$, $t_2 : [0, \infty]b/c, t_6 : [0, \infty]b/c$, $t_3 : [1, 1]nil/d, t_4 : [1, 1]nil/d$. In the following a_i represents the timed action corresponding to t_i for each $i \in \{1, 6\}$.

Let us consider the property $\phi \equiv x.\mu X.(\neg A \wedge x \leq 2) \vee [-]X$. Such a property means that for each computation state A is eventually left within two units of time. As shown in example 14 the $\mathcal{TSP}$ process corresponding to that statechart is the following process p:

$$\begin{aligned}
&p = I :: t \\
&t = rec\, x.\Big(A :: a_3.rec\, y.(H :: a_4.x \bigtriangledown H :: nil) \bigtriangledown A :: t_A\Big) \\
&t_A = p_1 \times p_2 \\
&p_1 = B :: rec\, x.D :: a_1.rec\, y.E :: a_5.x \\
&p_2 = C :: rec\, x.F :: a_2.rec\, y.G :: a6.x
\end{aligned}$$

We show the sketch of the proof of $p, \omega_0 \vdash \phi$ for $\omega_0 \equiv A = 0 \wedge D = 0 \wedge E = 0$. In the following with an abuse of notation we will adopt modalities $[K]$ and $< K >$ for $K \subseteq \Sigma(\Pi)$ with the obvious meaning. The proof rules can simply be adapted to handle those modalities.

$$\dfrac{p, \omega_0 \vdash \phi}{\dfrac{p, \omega_0 \wedge x = 0 \vdash \phi}{\dfrac{p, \omega_1 \vdash \phi}{\dfrac{p, \omega_1 \vdash (\neg A \wedge x \leq 2) \vee [-]\phi_0}{\dfrac{p, \omega_1 \vdash [-]\phi_0}{\Pi}}}\,\mu}\,thin}\,reset$$

where $\omega_1 \equiv 0 \leq A \leq 1 \wedge x = 0$ and $\phi_0 \equiv \nu X\{(p, \omega_1)\}.((\neg A \wedge x \leq 2) \vee [-]X)$.

Proof Π

$$\dfrac{p, \omega_1 \vdash [-]\phi_0}{\dfrac{I :: t_1 \bigtriangledown A :: t_A, \omega_1 \vdash [\epsilon]\phi_0 \wedge [\bar{\epsilon}]\phi_0}{\dfrac{I :: t_1 \bigtriangledown A :: t_A, \omega_1 \vdash [\epsilon]\phi_0}{\Pi_1} \qquad \dfrac{I :: t_1 \bigtriangledown A :: t_A, \omega_1 \vdash [\bar{\epsilon}]\phi_0}{\Pi_2}}}$$

where $t_1 = A :: a_3.rec\, y.(H :: a_4.t \bigtriangledown nil)$. Modalities $[\epsilon]$ and $[\bar{\epsilon}]$ stand for $[K_1]$ and $[K_2]$ such that $K_1 = \{\alpha \mid type(\alpha) = \epsilon\}$ and $K_2 = \{\alpha \mid type(\alpha) = \bar{\epsilon}\}$, respectively.

Proof Π_1

Proof Π_1 is given by two cases obtained by splitting (by rule $disj$) the constraint ω_1 as $x = 0 \wedge A = 1$ and $x = 0 \wedge A = 0$.

1. The first one is as follows.

$$\cfrac{\cfrac{\cfrac{I :: t_1 \bigtriangledown A :: t_A, x = 0 \wedge A = 1 \vdash [\epsilon]\phi_0}{I :: t_1, x = 0 \wedge A = 0 \vdash [\epsilon]\phi_0} \bigtriangledown[]\epsilon_2}{I :: t_2, x = 1 \wedge H = 0 \vdash \phi_0}}{\cfrac{I :: t_2, x = 1 \wedge H = 0 \vdash \neg A \wedge x \leq 2}{}}$$

where $t_2 = rec\, y.(H :: a_4.t \bigtriangledown H :: nil)$.

Note that rule $\bigtriangledown[]\epsilon_2$ is applied since $A = 1$; only the high-level transition can thus be executed.

2. The second one is as follows.

$$\cfrac{I :: t_1 \bigtriangledown A :: t_A, A = 0 \wedge x = 0 \vdash [\epsilon]\phi_0}{\cfrac{I :: t_1, \omega_1 \vdash [\epsilon]\phi_0}{} []\epsilon_3 \qquad \cfrac{A :: t_A, \omega_{1,1} \vdash [\epsilon](\phi_0)_{(I::t_1, A=1)\bigtriangledown}}{\cfrac{A :: t_A, \omega_{1,1} \vdash [\epsilon]\phi_0'}{\cfrac{p_1, \omega_{1,1} \vdash [K](\phi_0')_{A\times(p_4, G=0)}}{\Pi_1'}}}} \bigtriangledown[]\epsilon_1$$

where $\omega_{1,1} \equiv x = 0$, $p_4 = C :: rec\, z.G :: a_6.p_2$ and $p_3 = B :: rec\, z.E :: a_5.p_1$, $K = \{\alpha \mid \alpha \otimes (\mathcal{C}(\chi(t_2)) : \epsilon) = \alpha'$ such that $type(\alpha') = \epsilon\}$ and $\phi_0' \equiv (\phi_0)_{(I::t_1, A=1)\bigtriangledown}$.

First rule $\bigtriangledown[]\epsilon_1$ is applied to decompose the refined process into the high-level and the low-level process. Since $A = 0$ only actions of the low-level process have effectively to be considered. The parallel process $p_1 \times p_2$ is then decomposed according to the required type non-idle. The constraint used in the extended assertion $(\phi_0)_{(I::t_1, A=1)\bigtriangledown}$ is $A = 1$ since one unit of time passed in location A.

Proof Π_1'

$$\cfrac{p_1, \omega_{1,1} \vdash [K](\phi_o')_{A\times(p_4, G=0)}}{\cfrac{p_3, x = 1 \wedge E = 0 \vdash (\phi_0')_{A\times(p_4, G=0)}}{\Pi_{1,1}} \qquad \cfrac{p_1, x = 1 \wedge D = 1 \vdash (\phi_0')_{A\times(p_4, G=0)}}{\Pi_{1,2}}}$$

where $p_4 = C :: rec\, z.G :: a_6.p_2$ and $p_3 = B :: rec\, z.E :: a_5.p_1$.

There are two cases: either both processes perform an action ($\Pi_{1,1}$) or only process p_1 performs an action ($\Pi_{1,2}$). The possibility that p_2 performs a (non-idle) action while p_1 is idle is not considered, since transition t_1 triggers transition t_2.

Proof $\Pi_{1,1}$

$$
\cfrac{\cfrac{\cfrac{p_3, x=1 \wedge E=0 \vdash (\phi_0')_{A \times (p_4, G=0)}}{I :: t_1 \bigtriangledown A :: p_3 \times p_4, x=1 \wedge A=1 \wedge G=0 \wedge E=0 \vdash \phi_0}\ \text{shift}}{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash \phi_0}\ \text{thin}}{\Pi_3}
$$

Proof Π_3

$$
\cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash \phi_0}{\cfrac{\cdots}{\cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash [-]\phi_1}{\cfrac{\cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash [\epsilon]\phi_1}{\cfrac{I :: t_1, \omega_1 - 1 \vdash [\epsilon]\phi_1}{\cfrac{I :: t_2, H=0 \wedge x=2 \vdash \phi_1}{\cfrac{I :: t_2, H=0 \wedge x=2 \vdash \neg A \wedge x \leq 2}{}}}}\ \bigtriangledown[]\epsilon 2 \qquad \cfrac{I :: t_2, \omega_1 - 1 \vdash [\bar{\epsilon}]\phi_1}{}}{}}}}
$$

where $\phi_1 = \nu X.\{(p, \omega_1), (I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1)\}.((\neg A \wedge x \leq 2) \vee [-]X)$. (We have omitted part of the proof which is analogous to the one of $p, \omega_0 \vdash \phi$.)

Rule $\bigtriangledown[]\epsilon_2$ is applied instead of $\bigtriangledown[]\epsilon_1$ since the maximal delay of the higher-order process expired ($A = 2$). Therefore, only the high-level process has to be checked in the idle case. On the contrary no idle action is possible since both processes have to agree on idle actions and the high-level process is forced to proceed. Finally, proof $\Pi_{1,2}$ is analogous to $P_{1,1}$ and proof Π_2 is analogous to Π_1.

Let us consider an invariance property $\psi \equiv \nu X.(A \supset \phi \wedge [-]X)$ expressing the property that always location A is entered for at most two units of time.

$$
\cfrac{p, \omega_0 \vdash \psi}{\cfrac{\cfrac{p, \omega_0 \vdash \neg A \vee \phi}{\cfrac{p, \omega_0 \vdash \phi}{}} \qquad \cfrac{p, \omega_0 \vdash [-]\psi_0}{\cfrac{\{\Pi_1, \Pi_2\}}{\cfrac{P_1, \omega_1 - 1 \vdash \psi_0}{\Delta_1} \quad \cfrac{P_2, \omega_1 - 1 \vdash \psi_0}{\Delta_2} \quad \cfrac{P_3, \omega_1 - 1 \vdash \psi_0}{\Delta_3}}}}{}}
$$

where $\psi_0 \equiv \nu X\{(p, \omega_0)\}.(A \supset \phi \wedge [-]X)$ and $P_1 = I :: t_1 \bigtriangledown A :: p_3 \times p_4$, $P_2 = I :: t_1 \bigtriangledown A :: p_1 \times p_4$ and $P_3 = I :: t_1 \bigtriangledown A :: p_1 \times p_2$.

The notation $\{\Pi_1, \Pi_2\}$ is used to say that the same proofs arise in this case. Proofs Δ_1, Δ_2 and Δ_3 are quite similar.

Proof Δ_1

$$
\cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash \psi_0}{\cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash \neg A \vee \phi}{\cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash \phi}{\Delta_{1,1}}} \qquad \cfrac{I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash [-]\psi_1}{\Delta_{1,2}}}
$$

where $\psi_1 \equiv \nu X\{(p, \omega_0), (I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1)\}.(A \supset \phi \wedge [-]X)$.
Proof $\Delta_{1,1}$ is analogous to the one of $p, \omega_0 \vdash \phi$.

Proof $\Delta_{1,2}$

$$\begin{array}{c}
I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1 \vdash [-]\psi_1 \\ \hline
I :: t_2, H = 0 \vdash \psi_1 \\ \hline
I :: t_2, 0 \leq H \leq 2 \vdash \psi_1 \\ \hline
\begin{array}{cc}
\begin{array}{c} I :: t_2, 0 \leq H \leq 2 \vdash \neg A \vee \phi \\ \hline I :: t_2, 0 \leq H \leq 2 \vdash \neg A \\ \hline \end{array} &
\begin{array}{c} I :: t_2, 0 \leq H \leq 2 \vdash [-]\psi_2 \\ \hline p, \omega_0 \vdash \psi_2 \\ \hline \end{array}\ \nu_1
\end{array}
\end{array}$$

where $\psi_2 \equiv \nu X\{(p, \omega_0), (I :: t_2, 0 \leq H \leq 2), (I :: t_1 \bigtriangledown A :: p_3 \times p_4, \omega_1 - 1)\}.(A \supset \phi \wedge [-])$.

Note that as in proof Π_3 the maximal delay of the high-level component has been reached. Therefore, the process $A ::; p_3 \times p_4$ does not has to be considered in the decomposition of the refined process. Rule ν_1 concludes the proof since the initial timed process has already been checked.

11 Conclusions and Related Works

The first contribution of the paper is the definition of a timed process algebra which abstractly models timed (discrete) statecharts with transitions constrained within a minimal and maximal delay. Other process-algebraic versions of statecharts have been defined [47, 46] by introducing an operator of refinement similar to $\bigtriangledown$. The corresponding LTS semantics may seem quite similar to ours since LTS labels contain causality information. However, the LTS semantics of [47, 46] is not equivalent to the classical Pnueli and Shalev statecharts semantics. Causality, consistency and synchrony of steps require actually both the causal relation between input and outputs in LTS labels and the explicit representation both of idle and non-idle steps (see [30, 32]).

The compositional semantics achieved through $\mathcal{TSP}$ is the basis to define a compositional proof system for checking quantitative temporal properties of timed statecharts. The proof system is based on the approach of [9, 10, 8] combining local model checking with tagged fix-points [49] and compositional reasoning by means of a quotienting technique. The idea is that of avoiding the construction of LTS transitions for compound processes by removing components of the process, while transforming the formula into a specialized formula (extended assertion) representing the residual process. The method has been extended to automatic compositional verification of real-time properties for parallel timed graphs in [29] by combining quotienting with the region technique of [2].

Especially process refinement and synchronous parallel composition of $\mathcal{TSP}$ require special care as shown in the proof system of [33] for the un-timed version of $\mathcal{TSP}$. In the timed framework the main problem is that the underlying transition system is infinite. We have shown that by considering proofs statements parametrized to clock constraints both the method of tagged fix-point for local model checking and the compositional rules can be suitably adapted. Since clock constraints allow to implicitly represent set of timed states (even infinite), the proof system is sound in general and it is complete for a class of processes inlcluding the ones corresponding to timed statecharts.

The obtained proof system is quite complex and it is clear that very simple proofs only can be done by hand. The formal framework has been formalized in the proof assistant *Coq* [14]. The tool provides the user with mechanized assistance during the construction of proofs. It is worth mentioning that much of the complexity is due to the non-modular statecharts semantics. Compositional verification is a very difficult task for statecharts and other more elegant approaches [17, 6, 7, 13, 23, 50] seem to be not easily applicable. The well-known assume-guarantee paradigm [17, 6, 7] for instance does not seem to be adequate, since the necessary information about causality should be expressed in the formulas modelling the environment. The same difficulties are present in the work of [24] where an axiomatic compositional proof system for checking correctness properties of statecharts is defined. Properties are formalized in a first-order logic that allows to express classical liveness and safety properties. In order to achieve compositional verification the causal ordering between events has to be explicitly represented in formulas.

Automatic (non-compositional) verification of timed statecharts can be realized by applying well-known techniques of real-time model checking [2, 22, 21]. A translation of the considered version of timed statecharts in timed graphs is shown in [32]. The same approach is applied to a variant with delays of the synchronous language Argos in [37, 26].

One of the goals of a future research is to extend the proposed approach to a dense time framework. The proposed model of timed statecharts is not adequate for modelling applications working within an environment that may produce signals at arbitrary close instants of time. Dense timed versions of statecharts have been proposed in [27, 40, 39]. However, it is not clear if the synchrony hypothesis is still reasonable in the dense framework, since it would imply the system reaction time to be less than an arbitrary small interval of time. In contrast, dense time seems to arise from the asynchronous execution of concurrent processes.

Note: This paper is a complete and revised version of [31].

Acknowledgements I would like to thank Andrea Maggiolo-Schettini and Simone Tini for their helpful comments and suggestions. Furthermore, I would like to thank the anonymous referees for their careful reading of the paper.

References

[1] R. Alur. *Techniques for Automatic Verification of Real-time Systems.* PhD thesis, University of Stanford, 1991.

[2] R. Alur, C. Courcoubetis, and D.Dill. Model-Checking for Real-Time Systems. *Information and Computation*, 104(1):2–34, 1993.

[3] R. Alur and T.A. Henzinger. A really temporal logic. In *Proceeding of the 30th Annual Symposium on Foundations of Computer Science*, pages 164–169. IEEE Computer Society Press, 1989.

[4] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressivness. In *Proc. Fifth IEEE Symp. on Logic In Computer Science*, pages 390–401. IEEE Computer Society Press, 1990.

[5] R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In *Proceedings of REX workshop 'Real-time Theory in Practice'*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer-Verlag, Berlin, 1992.

[6] R. Alur and T.A. Henzinger. Reactive Modules. In *Proc. 11th IEEE Symp. on Logic In Computer Science*, 1996.

[7] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer ans S.K. Rajamani, and S. Tasiran. MOCHA: Modularity in Model Checking. In *Proceedings of CAV 98*, volume 1427 of *Lecture Notes in Computer Science*, pages 521–525. Springer-Verlag, Berlin, 1998.

[8] H.R. Andersen. *Verification of Temporal Properties of Concurrent Systems.* PhD thesis, University of Aarhus, Computer Science Department, 1993.

[9] H.R. Andersen, C. Stirling, and G. Winskel. A Compositional Proof System for the Modal μ-Calculus. In *Proc. Ninth IEEE Symp. on Logic In Computer Science*, pages 144–153, 1994.

[10] H.R. Andersen and G. Winskel. Compositional Checking of Satisfaction. *Formal Methods in System Design*, 1(4):144–153, 1992.

[11] G. Berry and G. Gonthier. The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation. *Science of Computer Programming*, 19:87–152, 1992.

[12] P. Caspi, N. Halbwachs, D. Pilaud, and J. Plaice. Lustre, a Declarative Language for Programming Synchronous Systems. In *Proccedings 14-th Symposium of Principles of Programming Languages*, pages 178–188, 1987.

[13] E. Chang, Z. Manna, and A. Pnueli. Compositional Verification of Real-time Systems. In *Proc. Ninth IEEE Symp. on Logic In Computer Science*, pages 458–465, 1994.

[14] C. Cornes, J. Courant, J. Fillatre, G. Huet, P. Manoury, C. Munoz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saibi, and B. Werner. The Coq Proof Assistant Reference Manual-Version 5.10. Technical report, INRIA, Rocquenfort, 1995.

[15] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.

[16] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *First Annual Workshop on Computer-Aided Verification*, 1989.

[17] O. Grumberg and D. E. Long. Model Checking and Modular Verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.

[18] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.

[19] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. Trakhtenbrot. Statemate: A Working Environment for the Development of Complex Reactive Systems. *IEEE Transations on Software Engineering*, 16:403–414, 1990.

[20] D. Harel, A. Pnueli, J. P. Schmidt, and R. Sherman. On the Formal Semantics of Statecharts. In *Proc. Second IEEE Symp. on Logic In Computer Science*, pages 54–64. IEEE Computer Society Press, 1987.

[21] T.A. Henzinger, O. Kupferman, and M.Y. Vardi. A Space-Efficient On-the-fly Algorithm for Real-Time Model Checking. In *Proceedings of CONCUR 96*, volume 1119 of *Lecture Notes in Computer Science*, pages 514–529, 1996.

[22] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model Checking for Real-Time Systems. *Information and Computation*, 111:193–244, 1994.

[23] J. Hooman and J. Widom. A Temporal-Logic Based Compositional Proof System for Real-Time Message Passing. In E. Odijk, M. Rem, and J.C. Syre, editors, *PARLE89: Parallel Architectures and Languages Europe*, volume 366 of *Lecture Notes in Computer Science*, pages 424–441. Springer-Verlag, Berlin, 1989.

[24] J.J.M. Hooman, R. Ramesh, and W.P. de Roever. A Compositional Axiomatization of Statecharts. *Theoretical Computer Science*, 101:289–335, 1992.

[25] C. Huizing and R. Gerth. Semantics of Reactive Systems in Abstract Time. In *Proceedings of REX workshop 'Real-time Theory in Practice'*, volume 600 of *Lecture Notes in Computer Science*, pages 291–314. Springer-Verlag, Berlin, 1992.

[26] M. Jourdan, F. Maraninchi, and A. Oliveiro. Verifying quantitative real-time properties of synchronous programs. In *Proceedings of CAV 93*, volume 697 of *Lecture Notes in Computer Science*, pages 347–358. Springer-Verlag, Berlin, 1993.

[27] Y. Kesten and A. Pnueli. Timed and Hybrid Statecharts and their Textual Representation. In *Formal Techniques in Real-time and Fault Tolerant Systems*, volume 571 of *Lecture Notes in Computer Science*, pages 591–620. Springer-Verlag, Berlin, 1992.

[28] D. Kozen. Results on the Propositional mu-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[29] F. Laroussinie and K.G. Larsen. Compositional Model Checking of Real Time Systems. In *Proceedings of CONCUR 93*, volume 962 of *Lecture Notes in Computer Science*, pages 27–41. Springer-Verlag, Berlin, 1995.

[30] F. Levi. A Process Language for Statecharts. In *Proc. of the LOMAPS Workshop on "Analysis and Verification of Multiple-Agent Languages"*, volume 1192 of *Lecture Notes in Computer Science*, pages 388–403, 1996.

[31] F. Levi. Compositional Verification of Timed Statecharts. In Dov Gabbay H. Barringer, Michael Fischer and Graham Gough, editors, *Proceedings of the Second International Conference on Temporal Logic (ICTL 97)*. Applied Logic Series, Kluwer Academic Publishers, 1997. to appear.

[32] F. Levi. *Verification of Temporal and Real-Time Properties of Statecharts*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 1997.

[33] F. Levi. A Compositional μ-Calculus Proof System for Statecharts Processes. *Theoretical Computer Science*, 216:271–310, 1999.

[34] A. Maggiolo-Schettini and M. Merro. Priorities, interrupts and preemption in Statecharts. In *Proc. of the Workshop Analysis and Verification of Multiple-agent Languages*, 1996. to appear in LNCS.

[35] A. Maggiolo-Schettini, A. Peron, and S. Tini. Equivalences of Statecharts. In *Proc. of CONCUR 96*, volume 1119 of *Lecture Notes in Computer Science*, pages 687–702. Springer-Verlag, Berlin, 1996.

[36] A. Maggiolo-Schettini and S. Tini. Projectability in Statecharts. In *Proc. MFCS'98 Workshop on Concurrency*. Electronic Notes in Computer Science 18, 1998.

[37] F. Maraninchi. Operational and Compositional Semantics of Synchronous Automaton Composition. In *Proc. of CONCUR 92*, volume 630 of *Lecture Notes in Computer Science*, pages 550–564, 1992.

[38] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[39] A. Peron. *Synchronous and Asynchronous Models for Statecharts*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 1993.

[40] A. Peron and A. Maggiolo-Schettini. Transitions as Interrupts: A New Semantics for Timed Statecharts. In *Proceedings of International Symposium on Theoretical Aspects of Computer Software*, number 789 in Lecture Notes in Computer Science, pages 806–821. Springer-Verlag, Berlin, 1994.

[41] A. Pnueli. The Temporal Logic of Concurrent Programs. In *Proc. of the 18th annual IEEE Symposium on Foundations of Computer Science*, pages 45–57, 1977.

[42] A. Pnueli and M. Shalev. What is in a Step. In *Theoretical Aspects of Computer Science*, volume 526 of *Lecture Notes in Computer Science*, pages 244–464. Springer-Verlag, Berlin, 1991.

[43] V. Pratt. A decidable mu-calculus. In *Proceedings of 22nd. FOCS*, pages 421–427, 1981.

[44] C. Stirling and D. Walker. Local Model Checking in the Modal mu-Calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[45] A.C. Uselton and S.A. Smolka. State Refinement in Process Algebra. In *Proceedings of the North American Process Algebra Workshop*, 1993.

[46] A.C. Uselton and S.A. Smolka. A Compositional Semantics for Statecharts using Labelled Transition Systems. In *Proc. of CONCUR 94*, volume 836 of *Lecture Notes in Computer Science*, pages 2–17, 1994.

[47] A.C. Uselton and S.A. Smolka. A Process Algebraic Semantics for Statecharts. In *Proceedings of IFIP Working Conference on Programming Concepts, Methods and Calculi*, 1994.

[48] M. von der Beeck. A Comparison of Statecharts Variants. In *Formal Techniques in Real-Time and Fault-Torelant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 128–148. Springer-Verlag, Berlin, 1994.

[49] G. Winskel. A Note on Model Checking the Modal Nu-calculus. *Theoretical Computer Science*, 83:157–187, 1991.

[50] P. Zhou, J. Hooman, and R. Kuiper. Compositional Verification of Real-Time Systems with Explicit Clock Temporal Logic. *Formal Aspects of Computing*, 8(3):294–323, 1996.

A Proofs

We exploit the following properties concerning the correspondence between assignments and constraints.

Proposition 45 Let C be a set of clocks and $C_1, C_2 \subseteq C$, $C_1 \cap C_2 = \emptyset$. For each assignment π over C and for each $\omega \in \Omega(C)$:

1. if $\pi \models \omega$ then $proc_{C_1}(\pi) \models proc_{C_1}(\omega)$. Moreover, if ω is satisfiable, then for each π' over C_1 such that $\pi' \models proc_{C_1}(\omega)$, there exists $\pi \models \omega$ and $\pi' = proc_{C_1}(\pi)$.
2. if $\pi \models \omega$, then $reset(\pi, C_1, C_2) \models reset(\omega, C_1\ , C_2)$. Moreover, if ω is satisfiable for each π' such that $\pi' \models reset(\omega, C_1, C_2)$, then there exists $\pi \models \omega$ and $reset(\pi, C_1, C_2,) = \pi'$.
3. $\pi \models \omega + \delta$ iff $\pi + \delta \models \omega$ and $\pi \models \omega - \delta$ iff $\pi - \delta \models \omega$ for each $\delta \in N$.

Proposition 46 Let C be a set of clocks. Let $C_1 \cap C_2 = \emptyset$, $C_1 \subseteq C$ and $C_{i,1}, C_{i,2}$, such that $C_{j,i} \cup C_{k,i} = C_i$, for each $i, j, k \in \{1, 2\}$ with $j \neq k$ and $C_{1,i} \cap C_{2,i} = \emptyset$. Let $\omega \in \Omega(C)$ such that, for each $x + n\mathcal{R}y + m \in \omega$, either $x, y \in C_{1,1}$ or $x, y \in C_{2,1}$. We have that $reset(\omega, C_1, C_2) \equiv reset(\omega, C_{1,1}, C_{1,2}) \wedge reset(\omega, C_{2,1}, C_{2,2})$.

Proof (Lemma 38)

1. Assume $P \subseteq \psi(\mu V.(\psi(V) \setminus P))$. Consider the function $\psi'(V) \equiv \psi(V) \setminus P$. We have $\psi'(\mu V.\psi(V)) = \psi(\mu V.\psi(V)) \setminus P = \mu V.\psi(V) \setminus P$. Therefore, $\psi'(\mu V.\psi(V)) \subseteq \mu V.\psi(V)$, i.e. $\mu V.\psi(V)$ is a prefixed point of ψ'. Since $\mu V.(\psi(V) \setminus P)$ is the least fixed point of this function, $\mu V.(\psi(V) \setminus P) \subseteq \mu V.\psi(V)$. Since ψ is monotonic, $\psi(\mu V.(\psi(V) \setminus P)) \subseteq \psi(\mu V.\psi(V)) = \mu V.\psi(V)$. Since $P \subseteq \psi(\mu V.(\psi(V) \setminus P))$, then $P \subseteq \mu V.\psi(V)$.

2. Assume $P \subseteq \psi(\nu V.(\psi(V) \cup P))$. Since $\nu V.(\psi(V) \cup P)$ is a fixed point of the function $\psi'(V) \equiv \psi(V) \cup P$, we have $\nu V.(\psi(V) \cup P) = P \cup \psi(\nu V.(\psi(V) \cup P))$ and since $P \subseteq \psi(\nu V.(\psi(V) \cup P))$, $\nu V.(\psi(V) \cup P) = \psi(\nu V.(\psi(V) \cup P))$. Hence, $\nu V.(\psi(V) \cup P)$ is a fixed point of ψ, i.e. a postfixed point. Therefore $\nu V.(\psi(V) \cup P) \subseteq \nu V.\psi(V)$, as $\nu V.\psi(V)$ is the greatest fixed point and by monotonicity $\psi(\nu V.(\psi(V) \cup P)) \subseteq \psi(\nu V.\psi(V))$. Since $P \subseteq \psi(\nu V.(\psi(V) \cup P))$, then $P \subseteq \psi(\nu V.\psi(V)) = \nu V.\psi(V)$. Assume $P \subseteq \nu V.\psi(V)$. Then $P \cup \nu V.\psi(V) = \nu V.\psi(V) = \psi(\nu V.\psi(V))$. Hence, $\nu V.\psi(V)$ is a postfixed point of $\psi'(V) = \psi(V) \cup P$. Since $\nu V.(\psi(V) \cup P)$ is the greatest fixed point of this function, then $\nu V.\psi(V) \subseteq \nu V.(\psi(V) \cup P)$. Since ψ is monotonic, $\nu V.\psi(V) = \psi(\nu V.\psi(V)) \subseteq \psi(\nu V.(\psi(V) \cup P))$. Hence, $P \subseteq \nu V.\psi(V)$ implies $P \subseteq \psi(\nu V.(\psi(V) \cup P))$.
3. Since $P \subseteq \mu V.\psi(V) = \psi^k(\emptyset)$ then there exists $\{k_1, \ldots, k_n\}$ such that $k_i \leq k$ and $k_i < k_{i+1}$, for each $i \in \{1, n\}$, and $\psi^{k_i}(\emptyset) \cap P = P'_i$ and for each $k' < k_i$, $\psi^{k'}(\emptyset) \cap P \subset P'_i$. Since ψ is monotonic, $P'_{i-1} \subset P'_i$.
   Let $P_i = (P'_i \setminus P'_{i-1})$. We prove that $P_i \subseteq \psi(\mu V.(\psi(V) \setminus P_i))$ for each $i \in \{1, n\}$. Let $\psi_i(V) = \psi(V) \setminus P_i$. Since $P_i \notin \psi^{k'}(\emptyset)$, for each $i \in \{1, n\}$ and for each $k' < k_i$, then $\psi^{k'}(\emptyset) = \psi_i^{k'}(\emptyset)$ for each $k' < k_i$. Hence, $\psi^{k_i-1}(\emptyset) = \psi_i^{k_i-1}(\emptyset)$. Since $P_i \subseteq \psi(\psi^{k_i-1}(\emptyset)) = \psi^{k_i}(\emptyset)$ then $P_i \subseteq \psi(\psi_i^{k_i-1}(\emptyset))$.
   Since $\psi_i^{k_i-1}(\emptyset) \subseteq \mu V.(\psi(V) \setminus P_i)$, by monotonicity $\psi(\psi_i^{k_i-1}(\emptyset)) \subseteq \psi(\mu V.(\psi(V) \setminus P_i))$. Hence, $P_i \subseteq \psi(\psi_i^{k_i-1}(\emptyset))$ implies $P_i \subseteq \psi(\mu V.(\psi(V) \setminus P_i))$.

Let p be a closed process and A be a closed formula. We denote by $\mathcal{R}_{(p,A)}$ the set of $(p, A) - regions$.

Before showing the proof of lemma 41 we introduce some basic concepts.

Definition 47 Let $p_1, p_2 \in \mathcal{R}_p$. For each $\alpha \in LTS(\Pi)$ we define $trans(p_1, \alpha, p_2) \in 2^{\Omega(loc(p_1))}$ as follows:

1. $trans(\ell_1 :: \ell_2 :: [m, M]a.t, \mathcal{C}(a) : \epsilon, \ell_1 :: t) = \{m \leq \ell_2 \leq M\}$;
2. $trans(\ell_1 :: \ell_2 :: [m, M]a.t, \sigma : \bar{\epsilon}, \ell_1 :: \ell_2 :: [m, M]a.t) = \{\ell_2 < M\}$ where $\sigma \in \#(a) \vee (\#(a) = \emptyset \wedge \sigma = \sigma_0)$;
3. $trans(\ell_1 :: \ell_2 :: b_1 + b_2, \alpha, p') = \cup_{i \in \{1,2\}} trans(\ell_1 :: \ell_2 :: b_i, \alpha, p')$ if $type(\alpha) = \epsilon$;
4. $trans(\ell_1 :: \ell_2 :: b_1 + b_2, \alpha, \ell_1 :: \ell_2 :: b_1 + b_2) = \wedge_{i \in \{1,2\}} trans(\ell_1 :: \ell_2 :: b_i, \alpha, \ell_1 :: \ell_2 :: b_i)$ if $type(\alpha) = \bar{\epsilon}$;
5. $trans(\ell :: t, \alpha, p') \in trans(\ell :: t \bigtriangledown p, \alpha, p')$ if $type(\alpha) = \epsilon$;
6. $trans(p, \alpha, p') \wedge delay(\ell :: t) + 1 \in trans(\ell :: t \bigtriangledown p, \alpha, \ell :: t \bigtriangledown p')$ if $type(\alpha) = \epsilon$;
7. $trans(\ell :: t \bigtriangledown p, \alpha, \ell :: t \bigtriangledown p) = trans(\ell :: t, \alpha, \ell :: t) \wedge trans(p, \alpha, p)$ if $type(\alpha) = \bar{\epsilon}$;
8. $trans(\ell :: rec\ x.t, \alpha, p') = trans(\ell :: t[rec\ x.t/x], \alpha, p')$;
9. $trans(\ell :: p_1 \times p_2, \alpha\ell :: p'_1 \times p'_2) = \wedge_{i \in \{1,2\}} trans(p_i, \alpha_i, p'_i)$ and $\alpha \in \alpha_1 \otimes \alpha_2$.

Intuitively, $trans(p_1, \alpha, p_2)$ gives necessary and sufficient conditions on the clocks of p_1 for the existence of a transition $(p_1, \pi) \overset{\alpha}{\mapsto} (p_2, \pi')$.

Proposition 48 Let $p_1, p_2 \in \mathcal{R}_p$ and let $\alpha \in LTS(\Pi)$. For each assignment π over $loc(p_1)$ there exists $(p_1, \pi) \overset{\alpha}{\mapsto} (p_2, \pi')$ iff there exists $\omega \in trans(p_1, \alpha, p_2)$ such that $\pi \models \omega$ and $\pi' = reset(\pi + 1, loc(p_1) \cap loc(p_2), loc(p_1) \setminus loc(p_2))$.

Proposition 48 can trivially be adapted to extended actions $\mathcal{E}(\Pi)$.

Proof (Lemma 41) It is sufficient to show that for a formula A, that is fixpoint-quantifier free and with free formula variables $X_1, \ldots, X_n$, if $\rho(X_i) \in 2^{\mathcal{R}_{(p,A)}}$, for each $i \in \{1, n\}$, then $[\![A]\!]_\rho \in 2^{\mathcal{R}_{(p,A)}}$.

The proof is by induction on the structure of A.

- $A = X$ is a logical variable. Then $[\![X]\!]_\rho = \{(p, \pi) \mid (p, \pi) \in \rho(X)\}$. Since $\rho(X) \in 2^{\mathcal{R}_{(p,A)}}$, then $[\![A]\!]_\rho \in 2^{\mathcal{R}_{(p,A)}}$.
- $A = P$ is a propositional symbol. Then $[\![A]\!]_\rho = \{(p, \pi) \mid p \in \gamma(P)\}$. Hence, $[\![A]\!]_\rho = \{(p, \pi) \mid p \in \gamma(P), \pi \models true\} \in 2^{\mathcal{R}_{(p,A)}}$.
- $A = \neg P$ where P is a propositional symbol. Then $[\![A]\!]_\rho = \{(p, \pi) \mid p \in S \setminus \gamma(P)\}$. Since the negation of a (p, A)-constraint is a (p, A)-constraint, then $[\![A]\!]_\rho \in 2^{\mathcal{R}_{(p,A)}}$.
- $A = \omega \in \Omega(C)$. By definition ω is a (p, A)-constraint and $[\![A]\!]_\rho = \{(p, \pi) \mid \pi \models \omega \wedge delay(p)\}$. Since for each process p, $\omega \wedge delay(p)$ is a (p, A)-constraint, then $[\![A]\!]_\rho \in 2^{\mathcal{R}_{(p,A)}}$.
- $A = A_1 \wedge A_2$. By definition $[\![A]\!]_\rho = [\![A_1]\!]_\rho \cap [\![A_2]\!]_\rho$. Since by induction hypothesis, for each $i \in \{1, 2\}$, there exist (p, A)-constraints $\omega_{p',i}$ such that $[\![A_i]\!]_\rho = \{(p', \pi) \mid \pi \models \omega_{p',i}\}$. Therefore, $[\![A_1]\!]_\rho \cap [\![A_2]\!]_\rho = \{(p', \pi) \mid \pi \models \omega_{p',1} \wedge \omega_{p',2}\}$. Since $\omega_{p',1} \wedge \omega_{p',2}$ is a (p, A)-constraint, then $[\![A_1]\!]_\rho \cap [\![A_2]\!]_\rho$ is a set of (p, A)-regions.
- $A = A_1 \vee A_2$. Analogous to the previous case.
- $A = x.A_1$. By definition $[\![x.A_1]\!]_\rho = \{(p', \pi) \mid (p', \pi[x := 0]) \in [\![A_1]\!]_\rho\}$. By induction hypothesis, $[\![A_1]\!]_\rho \in 2^{\mathcal{R}_{(p,A_1)}}$. Hence, for each process p' there exists a (p, A)-constraint $\omega_{p'} \in \Omega(clock(p') \cup C)$ such that $(p', \pi) \in [\![A_1]\!]_\rho$ iff $\pi \models \omega_{p'}$. We have that $\pi[x := 0] \models \omega_{p'}$ iff $\pi \models \omega_{p'}[x := 0]$. Therefore, $(p', \pi) \in [\![A]\!]_\rho$ iff $\pi \models \omega_{p'}[x := 0]$ and $\omega_{p'}[x := 0] \in \Omega(C \cup loc(p'))$ is a (p, A)-constraint. Hence, $[\![A]\!]_\rho \in 2^{\mathcal{R}_{p,A}}$.
- $A = < \alpha > A_1$. We show only the case $\alpha \in LTS(\Pi)$. By definition $[\![A]\!]_\rho = [\![< \alpha >]\!]([\![A_1]\!]_\rho) = \{(p_1, \pi) \mid (p_1, \pi) \overset{\alpha}{\mapsto} (p_2, \pi'), (p_2, \pi') \in [\![A_1]\!]_\rho\}$. By induction hypothesis, $[\![A_1]\!]_\rho \in 2^{\mathcal{R}_{(p,A_1)}}$. For each $p_2 \in \mathcal{R}_p$, there exists $\omega_{p_2} \in \omega(loc(p_2) \cup C)$, such that ω_{p_2} is a (p, A_1)-constraint and $\pi' \models \omega_{p_2}$ iff $(p_2, \pi') \in [\![A_1]\!]_\rho$. For processess p_i, p_j, let $\bar{\omega}_{i,j} = \vee_{\omega \in trans(p_i, \alpha, p_j)} \omega$. By proposition 48 $(p_i, \pi) \overset{\alpha'}{\mapsto} (p_j, \pi')$ iff $\pi \models \bar{\omega}_{i,j}$ and $\pi' = reset(\pi + 1, loc(p_i) \cap loc(p_j), loc(p_j) \setminus loc(p_i))$. Let $\omega_{i,j} \equiv \bar{\omega}_{i,j} \wedge (\omega_{p_j} + 1)[new(p_j) := 0]$ where $new(p_j) = loc(p_j) \setminus loc(p_i)$. We show that, for each process p_i, $(p_i, \pi) \in [\![< \alpha >]\!]([\![A_1]\!]_\rho)$ iff $\pi \models \bigvee_{j \in \{1,n\}} \omega_{i,j}$.
  1. $\Rightarrow$ Suppose $(p_i, \pi) \in [\![< \alpha >]\!]([\![A_1]\!]_\rho)$, i.e. there exists $(p_i, \pi) \overset{\alpha}{\mapsto} (p_j, \pi')$, with $(p_j, \pi') \in [\![A_1]\!]_\rho$. By proposition 48 $\pi \models \bar{\omega}_{i,j}$ and $\pi' = reset(\pi + 1, loc(p_j) \cap loc(p_i), new(p_j))$. Moreover, $\pi' \models \omega_{p_j}$. By proposition 45, $reset(\pi + 1, loc(p_i) \cap loc(p_j), new(p_j)) \models \omega_{p_j}$ implies $\pi \models (\omega_{p_j} + 1)[new(p_j) := 0]$. Thus, $\pi \models \omega_{i,j}$.
  2. $\Leftarrow$ Suppose $\pi \models \bigvee_{j \in \{1,n\}} \omega_{i,j}$. By proposition 48 since $\pi \models \bar{\omega}_{i,j}$, then there exists $(p_i, \pi) \overset{\alpha}{\mapsto} (p_j, \pi')$, where $\pi' = reset(\pi + 1, loc(p_i) \cap loc(p_j), new(p_j))$. By proposition 45, $\pi' \models \omega_{p_j}$. Therefore, $(p_j, \pi') \in [\![A_1]\!]_\rho$.
- $A = [\alpha]A_1$. Note that $[\![A]\!]_\rho = TS \setminus [\![< \alpha >]\!](TS \setminus [\![A_1]\!]_\rho)$. By induction hypothesis, $[\![A_1]\!]_\rho \in 2^{\mathcal{R}_{(p,A_1)}}$. Since the negation of (p, A)-constraint is a (p, A)-constraint and due to the results concerning $<>$, $[\![A]\!]_\rho \in 2^{\mathcal{R}_{(p,A)}}$.

Proof of lemma 43 is by induction on the structure of processes by taking a measure defined as follows:

1. $m(\ell_1 :: \ell_2 :: nil) = m(\ell_1 :: \ell_2 :: [m, M]a, .t) = 0;$
2. $m(\ell_1 :: \ell_2 :: b_1 + b_2) = max(m(\ell_1 :: \ell_2 :: b_1), m(\ell_1 :: \ell_2 :: b_2)) + 1;$

3. $m(\ell :: t \bigtriangledown p) = max(m(\ell : t), m(p)) + 1$;
4. $m(\ell :: p_1 \times p_2) = 1 + m(p_1) + 2m(p_2)$;
5. $m(\ell :: rec\ x.t) = m(\ell :: t[rec\ x.t/x])$.

Proposition 49 Let $p, \omega \models < \alpha > A$ such that $\alpha \in LTS(\Pi)$. There exist $\{\omega_i\}_{i\in\{1,n\}}$ such that $\omega \equiv \bigvee_{i\in\{1,n\}} \omega_i$, and for each $i \in \{1, n\}$

1. there exists $\omega_i' \in trans(p, \alpha, p')$ such that $\models \forall(\omega_i \supset \omega_i')$.
2. $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$.

Proof By proposition 48.

Proposition 49 can easily be extended to actions $\mathcal{E}(\Pi)$.

Proof (Lemma 43) We show only the relevant cases for $< \alpha > A$. If $p, \omega \models < \alpha > A$, then for each $\pi \models \omega$ there exists $(p, \pi) \overset{\alpha'}{\mapsto}_\alpha (p', \pi')$ such that $(p', \pi') \models A$. By proposition 49 there exist $\omega \equiv \bigvee_{i\in\{1,n\}} \omega_i$, such that for each $i \in \{1, n\}$ there exists $\omega' \in trans(p, \alpha', p')$ such that

1. by proposition 48 for all $\pi' \models \omega$ $(p, \pi) \overset{\alpha'}{\mapsto}_\alpha (p', \pi')$ and $\pi' = reset(\pi + 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p))$.
2. $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$.

Since a proof of $p, \omega \vdash < \alpha > A$ can be obtained by rule $disj$ from $\{p, \omega_i \vdash < \alpha >\}_{i\in\{1,n\}}$ we can assume without loss of generality $n = 1$. Therefore, let $p, \omega \models < \alpha > A$ and let $\omega' \in trans(p, \alpha', p')$ be the constraint that satisfy the previous properties.

- Assume $p = \ell_1 :: \ell_2 :: nil$.

By definition $\alpha = \sigma_0 : \bar{\epsilon}$ and $p' = p$. Therefore, $\omega - 1 \equiv reset(\omega - 1, loc(p), \emptyset)$ and $p, \omega - 1 \models A$. We have that $p, \omega \vdash < \alpha > A$ is derivable from $p, \omega - 1 \vdash A$, by rule $<> nil$.

- Assume $p = \ell_1 :: \ell_2 :: [m, M]a.t$.

There are two cases, either $\alpha = \mathcal{C}(a) : \epsilon$ and $p' = \ell_1 :: t$ or $\alpha = \sigma : \bar{\epsilon}$, for $\sigma \in \#(a)$, if $\#(a) \neq \emptyset$ or $\sigma_0 = \sigma$, and $p' = p$.

In the first case, $reset(\omega - 1, loc(p) \cap loc(p'), loc(p' \setminus loc(p)) \equiv reset(\omega - 1, C, loc(t))$ so that $\ell_1 :: t, reset(\omega - 1, C, loc(t)) \models A$. Moreover, $\models \forall(\omega \supset \omega'$ with $\omega' \in trans(p, \alpha, p')$ guarantees by construction $\models \forall(\omega \supset m \leq \ell_2 \leq M)$. Therefore, $p, \omega \vdash < \alpha > A$ is derivable from $\ell_1 :: t, reset(\omega - 1, C, loc(t)) \vdash A$, by rule $<> \epsilon$.

In the second case, $\omega_2 \equiv reset(\omega - 1, loc(p), \emptyset) \equiv \omega - 1$. Therefore, $p, \omega - 1 \models A$. Moreover, $\models \forall(\omega \supset \omega')$ guarantees $\models \forall(\omega \supset \ell_2 < M)$, and by proposition 45 $\models \forall(\omega - 1 \supset \ell_2 \leq M)$. Therefore, $p, \omega \vdash < \alpha > A$ is derivable from $p, \omega - 1 \vdash A$, by rule $<> \bar{\epsilon}$.

- $p = \ell_1 :: \ell_2 :: b_1 + b_2$. There are two cases: either $type(\alpha) = \epsilon$ or $type(\alpha) = \bar{\epsilon}$.
  - If $type(\alpha) = \epsilon$, then $\forall(\omega \supset \omega')$ with $\omega' \in trans(p, \alpha, p')$ iff $\omega' \in trans(\ell_1 :: \ell_2 :: b_i, \alpha, p')$ for some $i \in \{1, 2\}$. By proposition 48 and by $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$, $\ell_1 :: \ell_2 :: b_i, \omega \models < \alpha > A$. Since $m(\ell_1 :: \ell_2 :: b_i) < m(\ell_1 :: \ell_2 :: b_1 + b_2)$, by induction hypothesis there exist $\{p_j \in \mathcal{S}_{\ell_1::\ell_2::b_i}\}_{j\in\{1,n\}}$ and $\{p_j' \in \mathcal{S}_{\ell_1::\ell_2::b_i}\}_{j\in\{1,k\}}$ such that
    1. $\{p_j, \omega_j' \models A\}_{j\in\{1,n\}}$ and $\{p_j', \omega_j'' \models true\}_{j\in\{1,k\}}$;
    2. $\ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > A$ can be obtained from $\{p_j, \omega_j' \vdash A\}_{j\in\{1,n\}}$ and $\{p_j', \omega_j'' \vdash true\}_{j\in\{1,k\}}$.

By rule $+ <> \epsilon$, $\ell_1 :: \ell_2 :: b_1 + b_2, \omega \vdash < \alpha > A$ can be obtained from $\ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > A$.

– If $type(\alpha) = \bar{\epsilon}$, $\omega' \in trans(\ell_1 :: \ell_2 :: b_1 + b_2, \alpha, p')$ iff $p' = p$ and $\omega' \in trans(\ell_1 :: \ell_2 :: b_i, \alpha, \ell_1 :: \ell_2 :: b_i)$, for each $i \in \{1,2\}$. Since $reset(\omega - 1, loc(p), \emptyset) \equiv \omega - 1$ and $p', reset(\omega - 1, loc(p), \emptyset) \models A$, then $p, \omega - 1 \models A$. Since $m(\ell_1 :: \ell_2 :: b_i) < m(\ell_1 :: \ell_2 :: b_1 + b_2)$, by induction hypothesis there exist $\{p_{i,j} \in \mathcal{S}_{\ell_1::\ell_2::b_i}\}_{j \in \{1,n_i\} i \in \{1,2\}}$, such that

1. $\{p_{i,j}, \omega'_{i,j} \models true\}_{j \in \{1,n_i\}, i \in \{1,2\}}$;
2. $\ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > true$ can be obtained from $\{p_{i,j}, \omega'_{i,j} \vdash A\}_{j \in \{1,n_i\}}$, for each $i \in \{1,2\}$.

By rule $+ <> \bar{\epsilon}$, $p, \omega \vdash < \alpha > A$ can be obtained from $p, \omega - 1 \vdash A$ and $\{\ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > true\}_{i \in \{1,2\}}$. Moreover, $p_{i,j} \in \mathcal{S}_{\ell_1::\ell_2::b_1+b_2}$.

- $p = \ell :: t \bigtriangledown p_1$. There are two cases: either $type(\alpha) = \epsilon$ or $type(\alpha) = \bar{\epsilon}$.

– If $type(\alpha) = \epsilon$, there exists a constraint $\omega' \in trans(\ell :: t \bigtriangledown p, \alpha, p')$ such that $\forall(\omega \supset \omega')$ iff one of the following cases holds:

1. $\omega' \in trans(\ell :: t, \alpha, p')$. By proposition 48 for each $\pi_1 \models \omega_1$ with $\omega_1 \equiv pro_{C \cup loc(ell::t)}(\omega)$, there exists $(\ell :: t, \pi_1) \overset{\alpha}{\mapsto} (p', \pi'_1)$ and $\pi'_1 = reset(\pi_1 + 1, loc(\ell :: t) \cap loc(p'), loc(p') \setminus loc(\ell :: t))$. Since the process has distinct locations $reset(\pi_1 + 1, loc(\ell :: t) \cap loc(p'), loc(p') \setminus loc(\ell :: t)) = reset(\pi, loc(p) \cap loc(p'), loc(p') \setminus loc(p))$ where $\pi = \pi_1 \cup \pi_2$. Therefore, for each $\pi_1 \models \omega_1$, $p', reset(\pi_1 + 1, loc(\ell :: t) \cap loc(p'), loc(p') \setminus loc(\ell :: t)) \models A$ and $\ell :: t, \omega_1 \models < \alpha > A$. Since $m(\ell :: t) < m(\ell :: t \bigtriangledown p)$, by induction hypothesis there exist $\{p_i \in \mathcal{S}_{\ell::t}\}_{i \in \{1,n\}}$ and $\{p'_j \in \mathcal{S}_{\ell::t}\}_{j \in \{1,k\}}$ such that
   (a) $\{p_i, \omega'_i \models A\}_{i \in \{1,n\}}$ and $\{p'_j, \omega''_j \models true\}_{j \in \{1,k\}}$;
   (b) $\ell :: t, \omega_1 \vdash < \alpha > A$ can be obtained from $\{p_i, \omega'_i \vdash A\}_{i \in \{1,n\}}$ and $\{p'_j, \omega''_j \vdash true\}_{j \in \{1,k\}}$.

   By rule $\bigtriangledown <> \epsilon_1$ $\ell :: t \bigtriangledown p, \omega \vdash < \alpha > A$ from $\ell :: t, \omega_1 \vdash < \alpha > A$, by rule $\bigtriangledown <> \epsilon_1$. Moreover, $p_i, p'_j \in \mathcal{S}_{\ell::t \bigtriangledown p_1}$.
2. $\omega' \in trans(p_1, \alpha, p'_1) \wedge delay(\ell :: t) + 1$. By proposition 48 for each $\pi_1 \models \omega_1$ with $\omega_1 \equiv pro_{loc(p_1) \cup C}(\omega)$ $(p_1, \pi_1) \overset{\alpha}{\mapsto} (p'_1, \pi'_1)$ and $\pi'_1 \models reset(\omega_1, loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1))$. Since the process has all distinct locations $reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \equiv reset(\omega_1, loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1)) \wedge pro_{loc(\ell::t)}(\omega) - 1$ by proposition 46. Therefore, $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$ implies $\ell :: t \bigtriangledown p_1, reset(\omega_1, loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1)) \wedge pro_{loc(\ell::t)}(\omega) - 1 \models A$, i.e. $p'_1, reset(\omega_1 , loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1)) \models (A)_{(\ell::t, pro_{loc(\ell::t)}(\omega) - 1) \bigtriangledown}$.
   Therefore, $p_1, \omega_1 \models < \alpha > (A)_{(\ell::t, pro_{loc(\ell::t)}(\omega) - 1) \bigtriangledown}$. Since $m(p_1) < m(\ell :: t \bigtriangledown p_1)$, by induction hypothesis there exist $\{p_{1,i} \in \mathcal{S}_{p_1}\}_{i \in \{1,n\}}$ and $\{p'_{1,j} \in \mathcal{S}_{p_1}\}_{j \in \{1,k\}}$ such that
   (a) $\{p_{1,i}, \omega'_i \models (A)_{(\ell::t, pro_{loc(\ell::t)}(\omega) - 1) \bigtriangledown}\}_{i \in \{1,n\}}$, $\{p'_{1,j}, \omega''_j \models true\}_{j \in \{1,k\}}$;
   (b) $p_1, \omega_p \vdash < \alpha > (A)_{(\ell::t, pro_{loc(\ell::t)}(\omega) - 1)}$ can be obtained from $\{p_{1,i}, \omega'_i \vdash (A)_{(\ell::t, pro_{loc(\ell::t)}(\omega) - 1) \bigtriangledown}\}_{i \in \{1,n\}}$ and $\{p'_{1,j}, \omega''_j \vdash true\}_{j \in \{1,k\}}$.

   Since $\models \forall(\omega \supset delay(\ell :: t) + 1)$, then $\models \forall(pro_{loc(\ell::t)}(\omega) - 1 \supset delay(\ell :: t))$ then $\ell :: t \bigtriangledown p_1, \omega \vdash < \alpha > A$ can be obtained from $p_1, \omega_1 \vdash < \alpha > (A)_{(\ell::t, pro_{loc(\ell::t)}(\omega) - 1) \bigtriangledown}$, by rule $\bigtriangledown <> \epsilon_2$.

– If $type(\alpha) = \bar{\epsilon}$ the proof is analogous to the corresponding case for choice.

- $p = \ell :: p_1 \times p_2$.

Suppose $\alpha \in LTS(\Pi)$. By rules $\mathcal{E} <>$ a proof for $p, \omega \vdash < \alpha > A$ can be trivially obtained from one of $p, \omega \vdash < \alpha' > A$ for some proper $\alpha' \in \mathcal{E}(\alpha)$. Since $p, \omega \models < \alpha > A$ implies $p, \omega \models < \alpha' > A$ in the following we omit the details concerning the structure of extended types when they are not relevant.

By definition $\omega' \in trans(p, \alpha, p')$ iff $\omega' \equiv \omega_1' \wedge \omega_2'$ with $\omega_i' \in trans(p_i, \alpha_i, p_i')$ where $p' = \ell :: p_1' \times p_2'$ and $\alpha \in \alpha_1 \otimes \alpha_2$.

The proof is by cases on p_2.

- Suppose $p_2 = \ell_1 :: \ell_2 :: nil$. We have $p_2' = p_2$ and $\alpha_2 = \sigma_0 : \bar{\epsilon}$. By proposition 46, $reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \equiv \omega_1 \wedge \omega_2$ where $\omega_1 \equiv reset(pro_{C \cup loc(p_1)}(\omega) - 1, loc(p_1) \cap loc(p_1'), loc(p_1') \setminus loc(p_1))$ and $\omega_2 \equiv pro_{loc(\ell_1 :: \ell_2 :: nil)}(\omega) - 1$. Since $\models \forall(\omega \supset \omega'$ then $\models \forall(pro_{C \cup loc(p_1)}(\omega) \supset \omega_1'$ with $\omega_1' \in trans(p_1, \alpha_1, p_1')$. By proposition 48 for each $\pi_1 \models pro_{C \cup loc(p_1)}(\omega)$, $(p_1, \pi_1) \overset{\alpha_1}{\mapsto} (p_1', \pi_1')$ where $\pi_1' \models \omega_1$. Moreover, $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$ implies $p_1', \omega_1 \models (A)_{\ell \times (p_2, \omega_2)}$ so that $p_1, pro_{C \cup loc(p_1)}(\omega) \models < \alpha_1 > (A)_{\ell \times (p_2, \omega_2)}$.
  Since $m(p_1) < m(\ell :: p_1 \times p_2)$, by induction hypothesis, there exist $\{p_{1,i} \in \mathcal{S}_{p_1}\}_{i \in \{1,n\}}$ and $\{p_{1,j}' \in \mathcal{S}_{p_1}\}_{j \in \{1,k\}}$ such that
  1. $\{p_{1,i}, \omega_{1,i}' \models (A)_{\ell \times (p_2, \omega_2)}\}_{i \in \{1,n\}}$ and $\{p_{1,j}', \omega_{1,j}'' \models true\}_{j \in \{1,k\}}$;
  2. $p_1, pro_{C \cup loc(p_1)}(\omega) \vdash < \alpha_1 > (A)_{\ell \times (p_2, \omega_2)}$ can be obtained from $\{p_{1,i}, \omega_{1,i}' \vdash (A)_{\ell \times (p_2, \omega_2)}\}_{i \in \{1,n\}}$ and $\{p_{1,j}', \omega_{1,j}'' \vdash true\}_{j \in \{1,k\}}$.

  We obtain $\ell :: p_1 \times p_2, \omega \vdash < \alpha > A$ from $p_1, pro_{loc(p_1) \cup C}(\omega) \vdash < \alpha_1 > (A)_{\ell \times (p_2, \omega_2)}$ by applying rule $\times <> nil$, since $\alpha \in \alpha_1 \otimes \alpha_2$. Since $p_{1,i}, \omega_{1,i}' \models (A)_{\ell \times (p_2, \omega_2)}$, then $\ell :: p_{1,i} \times p_2, \omega_{1,i}' \wedge \omega_2 \models A$. Moreover, premises $\{p_{1,i}, \omega_{1,i}' \vdash (A)_{\ell \times (p_2, \omega_2)}\}_{i \in \{1,n\}}$ are obtained from $\{\ell :: p_{1,i} \times p_2, \omega_{1,i}' \wedge \omega_2 \vdash A\}_{i \in \{1,n\}}$, by shift rule. It is obvious that $p_{1,j}', \ell :: p_{1,i} \times \ell_1 :: t \in \mathcal{S}_{\ell :: p_1 \times p_2}$.
- Suppose $p_2 = \ell_1 :: \ell_2 :: [m, M]a.t$. There are two cases: either $p_2' = p_2$ and $\alpha_2 = \sigma : \bar{\epsilon}$ for $\sigma \in \#(a)$ or $\sigma = \sigma_0$ and $\#(a) = \emptyset$, or $p_2' = \ell_1 :: t$ and $\alpha_2 = \mathcal{C}(a) : \epsilon$.
  - For $\alpha_2 = \sigma : \bar{\epsilon}$ the proof is analogous to the one for nil. Note only that the side condition of rule $\times <>_2$ is guaranteed by $\models \forall(\omega \supset \omega')$ where $\omega' \in trans(p, \alpha, p')$.
  - Suppose $\alpha_2 = \mathcal{C}(a) : \epsilon$. By proposition 46, $reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \equiv \omega_1 \wedge \omega_2$ where $\omega_1 \equiv reset(pro_{C \cup loc(p_1)}(\omega) - 1, loc(p_1) \cap loc(p_1'), loc(p_1') \setminus loc(p_1))$ and $\omega_2 \equiv reset(pro_{loc(\ell_1 :: t)}\ (\omega) - 1, \emptyset, \{loc(t)\})$. Since $\models \forall(\omega \supset \omega')$ then by definition $\models \forall(pro_{C \cup loc(p_1)}(\omega) \supset \omega_1'$ with $\omega_1' \in trans(p_1, \alpha_1, p_1')$. By proposition 48 for each $\pi_1 \models pro_{C \cup loc(p_1)}(\omega)$, $(p_1, \pi_1) \overset{\alpha_1}{\mapsto} (p_1', \pi_1')$ where $\pi_1' \models \omega_1$. Since $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$, then $p_1', \omega_1 \models (A)_{\ell \times (\ell_1 :: t, \omega_2)}$ and $p_1, pro_{C \cup loc(p_1)}(\omega) \models < \alpha_1 > (A)_{\ell \times (\ell_1 :: t, \omega_2)}$.
    Since $m(p_1) < m(\ell :: p_1 \times p_2)$, by induction hypothesis, there exist $\{p_{1,i} \in \mathcal{S}_{p_1}\}_{i \in \{1,n\}}$ and $\{p_{1,j}' \in \mathcal{S}_{p_1}\}_{j \in \{1,k\}}$ such that
    1. $\{p_{1,i}, \omega_{1,i}' \models (A)_{\ell \times (\ell_1 :: t, \omega_3)}\}_{i \in \{1,n\}}$ and $\{p_{1,j}', \omega_{1,j}'' \models true\}_{j \in \{1,k\}}$;
    2. $p_1, pro_{C \cup loc(p_1)}(\omega) \vdash < \alpha_1 > (A)_{\ell \times (\ell_1 :: t, \omega_2)}$ can be obtained from $\{p_{1,i}, \omega_{1,i}' \vdash (A)_{\ell \times (\ell_1 :: t, \omega_3)}\}_{i \in \{1,n\}}$ and $\{p_{1,j}', \omega_{1,j}'' \vdash true\}_{j \in \{1,k\}}$.

    Since $\models \forall(\omega \supset \omega'$ and $\omega' \equiv \omega_1' \wedge \omega_2'$ then $\models \forall(\omega \supset \omega_2'$. Moreover, $\omega_2' \in trans(p_2, \alpha_2, p_2')$ implies $\models \forall(\omega \supset m \leq \ell_2 \leq M)$. Since $\alpha \in \alpha_1 \otimes \alpha_2$ we obtain $\ell :: p_1 \times p_2, \omega \vdash < \alpha > A$ from $p_1, pro_{loc(p_1) \cup C}(\omega) \vdash < \alpha_1 > (A)_{\ell \times (\ell_2 :: t, \omega_2)}$, by rule $\tau \times <>_1$. Since $p_{1,i}, \omega_{1,i}' \models (A)_{\ell \times (\ell_1 :: t, \omega_2)}$, then $\ell :: p_{1,i} \times \ell_1 :: t, \omega_{1,i}' \wedge \omega_3 \models A$. Moreover, premises $\{p_{1,i}, \omega_{1,i}' \vdash (A)_{\ell \times (\ell_1 :: t, \omega_2)}\}_{i \in \{1,n\}}$ are obtained from $\{\ell :: p_{1,i} \times \ell_1 :: t, \omega_{1,i}' \wedge \omega_3 \vdash A\}_{i \in \{1,n\}}$, by shift rule. It is obvious that $p_{1,j}', \ell :: p_{1,i} \times \ell_1 :: t \in \mathcal{S}_{\ell :: p_1 \times p_2}$.

- Suppose $p_2 = \ell_1 :: \ell_2 :: b_1 + b_2$. There are two cases: either $type(proj_2(\alpha)) = \epsilon$ or $type(proj_2(\alpha)) = \bar{\epsilon}$.

  - $type(proj_2(\alpha)) = \epsilon$. Since $\omega' \equiv \omega'_1 \wedge \omega'_2$, with $\omega'_i \in trans(p_i, \alpha_i, p'_i)$ and $p_2 = \ell_1 :: \ell_2 :: b_1 + b_2$, $\omega'_2 \in trans(p_2, \alpha_2, p'_2)$ iff $\omega'_2 \in trans(\ell_1 :: \ell_2 :: b_i, \alpha_2, p'_2)$ for some $i \in \{1,2\}$. By proposition 48 for each $\pi_2 \models pro_{loc(p_2)}(\omega)$, there exists $(\ell_1 :: \ell_2 :: b_i, \pi_2) \overset{\alpha_2}{\mapsto} (p'_2, \pi'_2)$ where $p'_2 = reset(\pi_2 + 1, loc(p'_2) \cap loc(p_2), loc(p'_2) \setminus loc(p_2))$. By rule $choice_1$, for each $\pi \models \omega$ with $\pi = pi_1 \cup \pi_2$, $(\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \pi) \overset{\alpha}{\mapsto} (\ell :: p'_1 \times p'_2, \pi'_1 \cup \pi'_2)$ and $\pi'_1 = reset(\pi_1 + 1, loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1))$. Since the process has distinct locations $\pi'_1 \cup \pi'_2 \models reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p))$ so that $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$ implies $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega \models < \alpha > A$.

    Since $m(\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i) < m(\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2)$, by induction hypothesis there exist $\{\ell :: p_{1,j} \times p_{2,j} \in \mathcal{S}_{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i}\}_{j \in \{1,n\}}$ and $\{\ell :: p'_{1,j} \times p'_{2,j} \in \mathcal{S}_{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i}\}_{j \in \{1,k\}}$ such that

    1. $\{\ell :: p_{1,j} \times p_{2,j}, \omega'_{1,j} \models A\}_{j \in \{1,n\}}$ and $\{\ell :: p'_{1,j} \times p'_{2,j}, \omega''_{1,j} \models true\}_{j \in \{1,k\}}$;
    2. $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > A$ can be obtained from $\{\ell :: p_{1,j} \times p_{2,j}, \omega'_{1,j} \vdash A\}_{j \in \{1,n\}}$ and $\{\ell :: p'_{1,j} \times p'_{2,j}, \omega''_{1,j} \vdash true\}_{j \in \{1,k\}}$.

    We obtain $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_1 + b_2, \omega \vdash < \alpha > A$ from $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > A$, by rule $\times <> +\epsilon$. Moreover, $\ell :: p_{1,j} \times p_{2,j}, \ell :: p'_{1,j} \times p'_{2,j} \in \mathcal{S}_{\ell :: p_1 \times p_2}$.

  - $type(proj_2(\alpha)) = \bar{\epsilon}$. Since $\omega' \equiv \omega'_1 \wedge \omega'_2$, with $\omega'_i \in trans(p_i, \alpha_i, p'_i)$ and $p_2 = \ell_1 :: \ell_2 :: b_1 + b_2$, $\omega'_2 \equiv \omega'_{2,1} \wedge \omega'_{2,2}$ where $\omega'_{2,i} \in trans(\ell_1 :: \ell_2 :: b_i, \alpha_2, \ell_1 :: \ell_2 :: b_i)$ for each $i \in \{1,2\}$. By proposition 48 for each $\pi_2 \models pro_{loc(p_2)}(\omega)$, there exists $(\ell_1 :: \ell_2 :: b_i, \pi_2) \overset{\alpha_2}{\mapsto} (\ell_1 :: \ell_2 :: b_i, \pi_2 + 1)$. By rule $\times$, for each $i \in \{1,2\}$ and for each $\pi \models \omega$ with $\pi = pi_1 \cup \pi_2$, $(\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \pi) \overset{\alpha}{\mapsto} (\ell :: p'_1 \times \ell_1 :: \ell_2 :: b_i, \pi'_1 \cup \pi_2 + 1)$ and $\pi'_1 = reset(\pi_1 + 1, loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1))$. Since the process has distinct locations $\pi'_1 \cup \pi_2 + 1 \models reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p))$ so that $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models (A)_{\times + b_j}$ and $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega \models < \alpha > (A)_{\times + b_j}$.

    Since $m(\ell :: p_1 \times \ell_1 : \ell_2 :: b_i) < m(\ell :: p_1 \times \ell_1 : \ell_2 :: b_1 + b_2)$, by induction hypothesis there exist $\{\ell :: p_{1,j,i} \times p_{2,j,i} \in \mathcal{S}_{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i}\}_{j \in \{1,n_i\}, i \in \{1,2\}}$, $\{\ell :: p'_{1,j,i} \times p'_{2,j,i} \in \mathcal{S}_{\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i}\}_{j \in \{1,k_i\}, i \in \{1,2\}}$ such that

    1. $\{\ell :: p_{1,j,i} \times p_{2,j,i}, \omega'_{1,j,i} \models (A)_{\times + b_k}\}_{j \in \{1,n_i\}, i \in \{1,2\}, k \neq i}$ and $\{\ell :: p'_{1,j,i} \times p'_{2,j,i}, \omega''_{1,j,i} \models true\}_{j \in \{1,k_i\}, i \in \{1,2\}}$;
    2. $\ell :: p_1 \times \ell_1 :: \ell_2 :: b_i, \omega \vdash < \alpha > A$ can be obtained from $\{\ell :: p_{1,j,i} \times p_{2,j,i}, \omega'_{1,j,i} \vdash (A)_{\times + b_k}\}_{j \in \{1,n_i\}}$ and $\{\ell :: p'_{1,j,i} \times p'_{2,j,i}, \omega''_{1,j,i} \vdash true\}_{j \in \{1,k_i\}}$, for each $i \in \{1,2\}$.

    We obtain $\ell :: p_1 \times \ell_1 : \ell_2 :: b_1 + b_2, \omega \vdash < \alpha > A$ from $\{\ell :: p_1 \times \ell_1 : \ell_2 :: b_i, \omega \vdash < \alpha > (A)_{\times + b_j}\}_{i,j \in \{1,2\}, i \neq j}$, by rule $\times <> +\bar{\epsilon}$. Moreover, $\ell :: p'_{1,j,i} \times p'_{2,j,i} \in \mathcal{S}_{\ell :: p_1 \times p_2}$ for each $j \in \{1, k_i\}$ and premises $\{\ell :: p_{1,j,i} \times p_{2,j,i}, \omega'_{1,j,i} \vdash (A)_{\times + b_k}\}_{j \in \{1,n_i\}}$ are obtained by shift rule from $\{\ell :: p_{1,j,i} \times p''_{2,j,i}, \omega'_{1,j,i} \vdash A\}_{j \in \{1,n_i\}}$, where $\ell :: p_{1,j,i} \times p''_{2,j,i} \in \mathcal{S}_{\ell :: p_1 \times p_2}$.

- Suppose $p_2 = \ell_1 :: t \bigtriangledown p_3$. There are two cases: either $type(proj_2(\alpha)) = \epsilon$ or $type(proj_2(\alpha)) = \bar{\epsilon}$.

  - $type(proj_2(\alpha)) = \epsilon$. By definition $\omega'_2 \in trans(p_2, \alpha_2, p'_2)$ iff either $\omega'_2 \in trans(\ell_1 :: t, \alpha_2, p'_2)$ or $\omega'_2 \in trans(p_3, \alpha_2, p'_3)$ $\wedge delay(\ell_1 :: t) + 1$ and $p'_2 = \ell_1 :: t \bigtriangledown p'_3$. The first case is analogous to the non-idle case of choice.

In the second case, by proposition 48 for each $\pi \models \omega$ with $\pi = \pi_1 \cup p_{2,1} \cup \pi_{2,2}$, $(p_3, \pi_{2,2}) \overset{\alpha_2}{\mapsto} (p'_3, \pi'_{2,2})$, where $\pi'_{2,2} = reset(\pi_{2,2}+1, loc(p_3) \cap loc(p'_3), loc(p'_3) \setminus loc(p_3))$. By rule $\times$, $(\ell :: p_1 \times p_3, \pi_1 \cup \pi_{2,2}) \overset{\alpha}{\mapsto} (\ell :: p'_1 \times p'_3, \pi'_1 \cup \pi'_{2,2})$ with $\pi'_1 = reset(\pi_1 + 1, loc(p_1) \cap loc(p'_1), loc(p'_1) \setminus loc(p_1))$. Moreover, by proposition 45 $\pi'_1 \cup \pi'_{2,2} \models \omega_3$ with $\omega_3 \equiv reset(pro_{loc(p_1) \cup loc(p_3) \cup C} - 1, loc(\ell :: p_1 \times p_3) \cap loc(\ell :: p'_1 \times p'_3), loc(\ell :: p'_1 \times p'_3) \setminus loc(\ell :: p'_1 \times p'_3))$. By proposition 46 $reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \equiv \omega_3 \wedge \omega_2$ with $\omega_2 \equiv pro_{loc(\ell_1::t)}(\omega - 1)$. Therefore, $p', reset(\omega - 1, loc(p) \cap loc(p'), loc(p') \setminus loc(p)) \models A$ implies $\ell :: p'_1 \times p'_3, \omega_3 \models (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}$.

Since $m(\ell :: p_1 \times p_3) < m(\ell :: p_1 \times \ell_1 :: t \bigtriangledown p_3)$, by induction hypothesis there exist $\{\ell :: p_{1,i} \times p_{3,i} \in \mathcal{S}_{\ell::p_1 \times p_3}\}_{i \in \{1,n\}}$ and $\{\ell :: p'_{1,j} \times p'_{3,j} \in \mathcal{S}_{\ell::p_1 \times p_3}\}_{j \in \{1,k\}}$ such that

1. $\{\ell :: p_{1,i} \times p_{3,i}, \omega_i \models (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}\}_{i \in \{1,n\}}$ and $\{\ell :: p'_{1,j} \times p'_{3,j}, \omega'_j \models true\}_{j \in \{1,k\}}$;
2. $\ell :: p_1 \times p_3, \omega_3 \vdash < \alpha > (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}$ can be obtained from $\{\ell :: p_{1,i} \times p_{3,i}, \omega_i \vdash (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}\}_{i \in \{1,n\}}$ and $\{\ell :: p'_{1,j} \times p'_{3,j}, \omega'_j \vdash true\}_{j \in \{1,k\}}$.

Since $\models \forall(\omega \supset delay(\ell_1 :: t) + 1$ then $\forall(\omega_2 \supset delay(\ell_1 :: t))$ by proposition 45. then $\models \forall(\omega_3 \supset delay(\ell_1 :: t))$. We obtain $\ell :: p_1 \times \ell_1 :: t \bigtriangledown p_3, \omega \vdash < \alpha > A$ from $\ell :: p_1 \times p_3, \omega_3 \vdash < \alpha > (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}$, by rule $\times <> \bigtriangledown\epsilon_2$.

Since $\ell :: p_{1,i} \times p_{3,i}, \omega_i \models (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}$, then $\ell :: p_{1,i} \times \ell_1 :: t \bigtriangledown p_{3,i}, \omega_i \wedge \omega_2 \models A$. Moreover, premises $\{\ell :: p_{1,i} \times p_{3,i}, \omega_i \vdash (A)_{\times(\ell_1::t,\omega_2)\bigtriangledown}\}_{i \in \{1,n\}}$ are obtained by shift rule from $\{\ell :: p_{1,i} \times \ell_1 :: t \bigtriangledown p_{3,i}, \omega_i \wedge \omega_2 \vdash A\}_{i \in \{1,n\}}$. It is obvious that $\ell :: p_{1,i} \times \ell_1 :: t \bigtriangledown p_{3,i}, \ell :: p'_{1,j} \times p'_{3,j} \in \mathcal{S}_{\ell::p_1 \times p_2}$.

- $type(proj_2(\alpha)) = \bar{\epsilon}$. It is analogous to the corresponding one for choice.

- $p_2 = \ell_1 :: rec\ x.t$ Straightforward.
- $p_2 = \ell_1 :: p_3 \times p_4$. By definition $\omega' \in trans(p, \alpha, p')$ iff $\omega' \equiv \omega'_1 \wedge \omega'_2$ and $\omega'_i \in trans(p_i, \alpha_i, p'_i)$ for each $i \in \{1, 2\}$ and $\alpha \in \alpha_1 \otimes \alpha_2$. Moreover, $\omega'_2 \equiv \omega'_{2,3} \wedge \omega'_{2,4}$ such that $\omega'_{2,3} \in trans(p_3, \alpha_3, p'_3)$ and $\omega'_{2,4} \in trans(p_4, \alpha_4, p'_4)$ with $\alpha_2 \in \alpha_3 \otimes \alpha_4$. The proof can be trivially obtained by noting that for the associative property of $\times$ $\omega'_1 \wedge \omega'_{2,3} \in trans(\ell_1 :: (p_1 \times p_3))$ and $\omega'_1 \wedge \omega_{2,3'} \wedge \omega'_{2,4} \in trans(\ell :: \ell_1 :: (p_1 \times p_3) \times p_4)$ and that by definition $m(\ell :: \ell_1 :: (p_1 \times p_3) \times p_4) < m(\ell :: p_1 \times p_2)$.

Proof (Theorem 44) If $p, \omega \models A$ there are two cases. Either $\omega \wedge delay(p)$ is not satisfiable or it is satisfiable. In the first case by rule *empty*, $p, \omega \vdash A$. In the second case by rule *disj* we can assume $\models \forall(\omega \supset delay(p))$. The proof is by induction on $\prec$, that is well-founded for finite-state timed graphs.

1. The cases of $A \in \{P, \neg P, \omega, A_1 \vee A_2, A_1 \wedge A_2\}$ are trivial.
2. $A \equiv x.A_1$. By definition $p, \omega \models x.A_1$ iff, for each $\pi \models \omega$, $(s, \pi[x := 0]) \models A_1$. Since ω is satisfiable, by proposition 45 for each $\pi' \models reset(\omega, (C \cup loc(p)) \setminus \{x\}, x)$, there exists $\pi \models \omega$, such that $\pi' = pro_{(C \cup loc(p)) \setminus \{x\}}(\pi) \cup \{x := 0\}$. Since $x \in C$, then $\pi[x := 0] = \pi'$. Hence, $(s, \pi') \models A_1$, for each $\pi' \models reset(\omega, (C \cup loc(p)) \setminus \{x\}, x)$, i.e. $s, reset(\omega, (C \cup loc(p)) \setminus \{x\}, x) \models A_1$. Since $A_1 \prec A$, by induction hypothesis, $p, reset(\omega, (C \cup loc(p)) \setminus \{x\}, x) \vdash A_1$ and by rule *reset*, $p, \omega \vdash A$.
3. $A \equiv < \alpha > A_1$. By lemma 43 there exists $\{p_i, \omega_i \models A_1\}_{i \in \{1,n\}}$ and $\{p'_j, \omega'_j \models true\}_{j \in \{1,k\}}$, such that $p, \omega \vdash < \alpha > A_1$ is derivable from $\{p_i, \omega_i \vdash A_1\}_{i \in \{1,n\}}$ and $\{p'_j, \omega'_j \vdash true\}_{j \in \{1,k\}}$. By induction hypothesis proofs $\{p_i, \omega_i \vdash A_1\}_{i \in \{1,n\}}$ and $\{p'_j, \omega'_j \vdash true\}_{j \in \{1,k\}}$ indeed exist.
4. $A = [\alpha]A_1$. Similar to the previous case.

5. $A = \nu X\{U\}.B$. If $[\![(p,\omega)]\!] \subseteq [\![U]\!]$, then by ν_1, $p,\omega \vdash \nu X\{U\}.B$. Otherwise, by lemma 41, $[\![\nu X\{U\}.B]\!]$ is a (p,A)-region, namely there exists a (p,A)-constraint $\omega' \in \Omega(loc(p) \cup C)$, such that $\models \forall(\omega \supset \omega')$ and $p,\omega' \models A$. By rule *thin* $p,\omega \vdash A$ is derivable from $p,\omega' \vdash A$. By lemma 38, $(p,\omega') \in [\![B[\nu X\{U \cup (p,\omega')\}.B/X] \cup U]\!]$. There are two cases:

   (a) If $[\![(p,\omega')]\!] \cap [\![U]\!] = \emptyset$, then $p,\omega' \models B[\nu X\{U \cup (p,\omega')\}.B/X]$. Moreover, since $B[\nu X\{U \cup (p,\omega')\}.B/X] \prec A$, by induction hypothesis $p,\omega' \vdash B[\nu X\{U \cup (p,\omega')\}.B/X]$. By rule ν_2 $p,\omega' \vdash A$.

   (b) If $[\![(p,\omega')]\!] \cap [\![U]\!] \neq \emptyset$, then there exists $\{(p,\omega_i) \in U\}_{i \in \{1,n\}}$, such that $\omega' \wedge \omega_i$ is satisfiable, for each $i \in \{1,n\}$. Consider $\omega'_1 \equiv \omega' \wedge (\bigvee_{i \in \{1,n\}} \omega_i)$ and $\omega'_2 \equiv \omega' \wedge \neg(\bigvee_{i \in \{1,n\}} \omega_i)$. It is obvious that $\omega' \equiv \omega'_1 \vee \omega'_2$. Hence, $p,\omega' \vdash A$ can be obtained by rule *disj* from $p,\omega'_1 \vdash A$ and $p,\omega'_2 \vdash A$. Moreover, $\models \forall(\omega'_1 \supset \bigvee_{i \in \{1,n\}} \omega_i)$. Hence, by rule ν_1, $p,\omega'_1 \vdash A$. On the other hand, $[\![(p,\omega'_2)]\!] \cap [\![U]\!] = \emptyset$. Therefore, $p,\omega'_2 \models B[\nu X\{U \cup (p,\omega'_2)\}.B/X]$ and, since $B[\nu X\{U \cup (p,\omega'_2)\}.B/X] \prec A$, by induction hypothesis $p,\omega'_2 \vdash B[\nu X\{U \cup \{(p,\omega'_2)\}.B/X]$. By applying rule ν_2 $p,\omega'_2 \vdash A$.

6. $A = \mu X\{U\}.B$. By lemma 38 and by lemma 41, there exist (p,A)-constraints $\omega_1,\ldots,\omega_k$, such that $\omega \equiv \bigvee_{i \in \{1,k\}} \omega_i$ and $p,\omega_i \models B[\mu X\{U \cup (p,\omega_i)\}.B/X]$. Moreover, by the semantics of tagged least fixpoints $[\![(p,\omega_i)]\!] \cap [\![U]\!] = \emptyset$. Since $B[\mu X\{U \cup (p,\omega_i)\}.B/X] \prec A$ by induction hypothesis, $p,\omega_i \vdash B[\mu X\{U \cup (p,\omega_i)\}.B/X]$, for each $i \in \{1,k\}$. By rules *disj* and rule μ, we have $p,\omega \vdash A$.