INTERPRETATIONS OF AUTOMATA

Tim A.C. Willemse

Eindhoven University of Technology,

Department of Mathematics and Computer Science

5600 MB Eindhoven, The Netherlands

T.A.C.Willemse@tue.nl

Abstract

Timed and hybrid automata are models designed for describing real-time or hybrid systems. Both models are wide-spread and are widely accepted. Process algebras are often used to study fundamental concepts of processes and their behaviours. In this paper, we discuss translations from timed automata and hybrid automata to process terms in the process algebra $\mu\mathrm{CRL}_t^\bullet$. These translations provide directions for extending timed process algebras to cope with hybrid systems.

1 Introduction

Timed and hybrid automata (see [3, 17, 13]) are widely accepted and used formal models for describing timed and mixed discrete-continuous systems. These models are in essence purely semantical models. Various classes of automata have been identified that allow for automated answering of questions about their behaviour. The results thereof have led to the development of various tools (see e.g. HyTech [14], Uppaal [20, 21]) that efficiently can check properties of a system.

Process algebras have been introduced to study the theory and the fundamental concepts of processes, non-determinism, concurrency and behavioural equivalences. Moreover, process algebras are widely used for studying and performing algebraic verifications. Many variants of process algebras have been developed, e.g. CCS [23], CSP [18] and ACP [5]. Without a means of expressing timing dependencies, their expressiveness is limited. However, many different timed extensions of process algebras have been proposed.

The process algebra used in this paper is $\mu\mathrm{CRL}_t^\bullet$, which is a minor extension of $\mu\mathrm{CRL}_t$ [11]. Data is an integral part of $\mu\mathrm{CRL}_t$, allowing to express both timing and data dependencies in a single formalism.

In this paper we provide and investigate the translation of arbitrary timed and hybrid automata to process terms in $\mu\mathrm{CRL}_t$. The need for such a translation is to provide an intuition on the relation between the theories of process algebra and timed and hybrid automata. Moreover, this translation may provide a sound basis for extending existing tool-support for μCRL to cope with time in an efficient manner. Although $\mu\mathrm{CRL}_t$ was not developed and intended to be used in the context of hybrid systems, several case studies have already been conducted in this area using $\mu\mathrm{CRL}_t$ (see e.g. [12, 26]). Until now, this has been done on an ad-hoc basis. Therefore, an investigation of the concepts of the well-established model of a hybrid automaton provides a solid basis for further investigations into process algebras and their extensions that deal with continuous processes. This has revealed an unexpected difference between $\mu\mathrm{CRL}_t$ and the model of a hybrid automaton. Equivalence in terms of bisimulation defined for the model of a hybrid automaton is more discriminative than the standard notion of bisimulation of two timed process terms. However, classes of hybrid automata can be identified for which both notions of bisimulation are equally discriminative.

The timed and hybrid automata discussed in this paper are based on [3, 17] and [13]. However, we will not use their standard semantics, but a slight variation thereof. The reason for doing so is explained in section 3. This alteration immediately introduces the need to also discuss a variant of μCRL$_t$, i.e. μCRL$_t^\bullet$. However, the results obtained in this paper straightforwardly carry over to the standard semantics.

Providing a translation of timed or hybrid automata to process terms is not straightforward. The operational semantics of timed and hybrid automata is defined in terms of two-phase transition systems, employing a notion of relative time. At the other end of the spectrum, μCRL$_t$ is based on a time-stamped transition system, employing a notion of absolute time. In this paper, we have chosen to reformulate the standard semantics of timed and hybrid automata and our variations thereof to equivalent absolute time, time-stamped transition systems. This is motivated by the large number of rules for the operational semantics of μCRL$_t$, in contrast to the small number of rules for the semantics of timed and hybrid automata.

This paper is outlined as follows. Section 2 introduces the basic notions of the timed transition systems that are used in this paper. In section 3, both the model of a timed automaton and a hybrid automaton are introduced. We provide both a two-phase semantics of these models and a time-stamped semantics and relate them. In section 4, we give a short overview of the process algebra μCRL$_t^\bullet$. Section 5 subsequently shows how we can interpret arbitrary timed and hybrid automata in this process algebra, and we show that these interpretations are sound. A discussion on the obtained results can be found in section 6.

2 Preliminaries

Section 2.1 introduces the notion of *labelled transition systems* and defines a suitable equivalence relation on labelled transition systems. Subsequently, we discuss two different extensions of labelled transition systems with a notion of time in section 2.2.

2.1 Labelled Transition Systems

The conceptual model for the behaviour of processes used in this paper is that of *labelled transition systems*. A labelled transition system can be seen as a graph, connecting nodes via edges. This graph denotes the dynamic behaviour of a system, e.g. its causal relations and its moments of choice. The edges of a labelled transition system are decorated with elements from a label-set, representing the execution of an action. The nodes represent the states of a given system. A labelled transition system is defined as follows:

Definition 2.1 (*Labelled Transition System*).
A labelled transition system X is a tuple $\langle S, S^0, \Sigma, \longrightarrow\rangle$, where

- S is a (possibly infinite) set of states,
- $S^0 \subseteq S$ is the set of initial states,
- Σ is the set of labels,
- $\longrightarrow \subseteq S \times \Sigma \times S$ is the transition function.

Instead of writing $(s, \sigma, s') \in \longrightarrow$, we will write $s \xrightarrow{\sigma} s'$.

Many notions of equivalence on labelled transition systems have been defined; here, we use the notion of *strong bisimulation*. Strong bisimulation formalises the notion of two adversaries countering each other's every move, and hence provides a very intuitive notion of equivalence.

Definition 2.2 (*Strong Bisimulation*).
Let $X = \langle S, S^0, \Sigma, \longrightarrow \rangle$ and $Y = \langle L, L^0, \Gamma, \longrightarrow \rangle$ be two labelled transition systems. A bisimulation relation $R \subseteq S \times L$ is a binary relation, such that the initial states are related:

- for all $l^0 \in L^0$, there exists an $s^0 \in S^0$, such that $l^0 \ R \ s^0$.
- for all $s^0 \in S^0$, there exists an $l^0 \in L^0$, such that $l^0 \ R \ s^0$,

Moreover, both systems should be able to mimic each others behaviours:

for each $s \in S$ and $l \in L$, if $sR\, l$ and

- $s \xrightarrow{\sigma} s'$ then there exists an $l' \in L$, such that $l \xrightarrow{\sigma} l'$ and $s' R\, l'$,
- $l \xrightarrow{\gamma} l'$ then there exists an $s' \in S$, such that $s \xrightarrow{\gamma} s'$ and $s' R\, l'$.

In case such a relation R exists, we write $R : \ X \ \underline{\leftrightarrow} \ Y$, and we say X and Y are strongly bisimilar.

In practice, labelled transition systems are not used to model complex systems. However, due to their intuitiveness, the model is often used as the underlying model for high-level formalisms.

2.2 Timed Transition Systems

Labelled transition systems are intuitive models when time does not play a crucial role in a system's behaviour. However, the need for models of real-time systems is steadily increasing. Hence, there is an apparent need for incorporating a notion of time in the model of a labelled transition systems. This extension can be done in more than one way. In this paper, we will discuss both the models of *two-phase* transition systems and *time-stamped* transition systems.

The model of two-phase transition systems is probably one of the most accepted extensions of labelled transition systems. The passage of time is modelled as an explicit transition. Often, the label set is extended with a set representing time. The labels in the transition relation then either represent the passage of time, or the execution of an action. However, here we introduce a special labelled transition relation to denote the passage of time. This is done to make a clear separation between the execution of actions and the passage of time.

Definition 2.3 (*Two-Phase Transition System*).
A two-phase transition system X is a tuple $\langle S, S^0, \Sigma, \mathbb{T}, \longrightarrow, \longmapsto \rangle$, where

- S is a set of states,
- $S^0 \subseteq S$ is the set of initial states,
- Σ is a set of labels, representing actions,
- $\mathbb{T}$ is a set representing the time domain,
- $\longrightarrow \subseteq S \times \Sigma \times S$ is the set of action transitions,
- $\longmapsto \subseteq S \times \mathbb{T} \times S$ is the set of time transitions.

Instead of writing $(s, \sigma, s') \in \longrightarrow$, we write $s \xrightarrow{\sigma} s'$, and instead of writing $(s, t, s') \in \longmapsto$, we write $s \stackrel{t}{\longmapsto} s'$.

The interpretation of a transition $s \overset{t}{\longmapsto} s'$ is dependent on the notion of time we work in. If we use *relative* time, i.e. when we measure time from the start of the current state, the transition $s \overset{t}{\longmapsto} s'$ denotes the passage of t time-units, thereby changing the system's state from s to s'. When we use *absolute* time, i.e. when we measure time from the start of the system, the transition $s \overset{t}{\longmapsto} s'$ denotes a change of state s to s' at time t. Using absolute time, the current time is often encoded in the state of the system. This is not needed when we use relative time.

Equality on the level of two-phase transition systems is defined by means of *timed strong bisimulation*. Timed strong bisimulation is a straightforward extension of the notion of strong bisimulation, in which two adversaries not only counter each other's every action, but they also should counter each other's every time-step.

Definition 2.4 (*Timed Strong Bisimulation*).
Let $X = \langle S, S^0, \Sigma, \mathbb{T}, \longrightarrow, \longmapsto\rangle$ and $Y = \langle L, L^0, \Gamma, \mathbb{T}, \longrightarrow, \longmapsto\rangle$ be two two-phase transition systems. A bisimulation relation $R \subseteq S \times L$ is a binary relation, such that the initial states are related:

- For each state $s^0 \in S^0$, there should exist a state $l^0 \in L^0$ such that $s^0 R l^0$,
- For each state $l^0 \in L^0$, there should exist a state $s^0 \in S^0$ such that $s^0 R l^0$.

Moreover, both systems should be able to mimic each other's behaviours:

for each $s \in S$ and $l \in L$, if $s\ R\ l$ and

- If $s \overset{\sigma}{\longrightarrow} s'$, then there exists $l' \in L$, such that $l \overset{\sigma}{\longrightarrow} l'$ and $s'\ R\ l'$,
- If $l \overset{\sigma}{\longrightarrow} l'$, then there exists $s \in S$, such that $s \overset{\sigma}{\longrightarrow} s'$ and $s'\ R\ l'$,
- If $s \overset{t}{\longmapsto} s'$, then there exists $l' \in L$, such that $l \overset{t}{\longmapsto} l'$ and $s'\ R\ l'$,
- If $l \overset{t}{\longmapsto} l'$, then there exists $s \in S$, such that $s \overset{t}{\longmapsto} s'$ and $s'\ R\ l'$.

In case such a relation R exists, we write $R:\ X\ \underline{\leftrightarrow}_t^t\ Y$, and we say X and Y are timed strongly bisimilar.

Opposed to considering the passage of time as a special kind of action for which one can define transitions, the model of time-stamped transition systems models passage of time as a property of a state. This means that for all states in the system, it must be expressed until which time the system can stay in this state. This has two consequences; instead of taking a time-transition to a new state, the state will remain the same until an action is executed. Second, it must be explicitly mentioned *when* the action is executed. A time-stamped transition systems is defined as follows:

Definition 2.5 (*Time-stamped Transition System*).
A time-stamped transition system X is a tuple $\langle S, S^0, \Sigma, \mathbb{T}, \longrightarrow, \mathcal{U}\rangle$, where

- S is a set of states,
- S^0 is the set of initial states,
- Σ is the set of labels, representing actions,
- $\mathbb{T}$ is a set representing the time domain,
- $\longrightarrow \subseteq S \times (\Sigma \times \mathbb{T}) \times S$ is the transition relation,

- $\mathcal{U} \subseteq S \times \mathbb{T}$ is the delay predicate.

Instead of writing $(s, (\sigma, t), s') \in \longrightarrow$, we write $s \xrightarrow{\sigma}_t s'$. Instead of writing $(s, t) \in \mathcal{U}$ we write $\mathcal{U}_t(s)$.

The interpretation of the passage of time and the time-stamps on the action-transitions is again dependent on the setting of time we work in. When we work using relative time, the predicate $\mathcal{U}_t(s)$ expresses that the system can idle for t time-units in state s, where time is measured entering in state s. Whenever we write $s \xrightarrow{\sigma}_t s'$, this must be interpreted that when the system is in state s, it can execute an action σ at exactly t time-units from entering state s and move to state s'. These interpretations change slightly when we work using absolute time. The predicate $\mathcal{U}_t(s)$ then expresses that the system can idle at least until time t (measured from the start of the system) when the system is in state s. The transition $s \xrightarrow{\sigma}_t s'$, then expresses that when the system is in state s, it can execute an action σ at absolute time t and subsequently move to state s'. Using absolute time, the notion of the *current* time is often encoded in the state.

The notion of equality on time-stamped transition systems we will work with is called *timed bisimulation*. A formal definition of timed bisimulation can be found below.

Definition 2.6 (*Timed Bisimulation*).
Let $X = \langle S, S^0, \Sigma, \mathbb{T}, \longrightarrow, \mathcal{U} \rangle$ and $X = \langle L, L^0, \Gamma, \mathbb{T}, \longrightarrow, \mathcal{U} \rangle$ be two timed transition systems. A bisimulation relation $R \subseteq S \times L$ is a binary (symmetric) relation, such that the initial states are related:

- For each state $s^0 \in S^0$, there exists a state $l^0 \in L^0$ such that $s^0 R l^0$,
- For each state $l^0 \in L^0$, there exists a state $s^0 \in S^0$ such that $s^0 R l^0$.

Moreover, both systems should be able to mimic each other's behaviours:

for each $t \in \mathbb{T}$, $s \in S$ and $l \in L$, if $sR\,l$ and

- If $s \xrightarrow{\sigma}_t s'$ then there exists an $l' \in L$, such that $l \xrightarrow{\sigma}_t l'$ and $s' R\, l'$,
- If $l \xrightarrow{\gamma}_t l'$ then there exists an $s \in S$, such that $s \xrightarrow{\gamma}_t s'$ and $s' R\, l'$,
- $\mathcal{U}_t(s)$ iff $\mathcal{U}_t(l)$.

In case such a relation R exists, we write $R : X \;\underline{\leftrightarrow}_t\; Y$, and we say X and Y are timed bisimilar.

In the subsequent sections, we use both the two-phase transition system model and the model of time-stamped transition systems when providing a semantics to a high-level formalism. From now on, we will assume a fixed set $\mathbb{T}$ as an abstract representation of the set of non-negative real numbers ($\mathbb{R}_{\geq 0}$) with $\mathbf{0}$ as its least element.

3 Timed and Hybrid Automata

Time-stamped transition systems and two-phase transition systems are very intuitive, yet they are hardly applicable when modelling (real) real-time systems. However, as a basis for interpreting high-level formalisms, these models are often employed.

In this section, the focus is on *Timed Automata* [3, 17] and *Hybrid Automata* [13]. Since their introduction, these formalisms have become rather popular. This popularity is partly explained by the availability of tool-support for performing automatic verifications and validations on models.

In section 3.1, discusses the standard two-phase semantics of a timed automaton, and a minor adjustment thereof. In this paper we use this latter semantics. The reason for doing so is explained in section 3.1. Furthermore, we relate our two-phase semantics to an equivalent time-stamped semantics for timed automata.

Section 3.2 introduces hybrid automata and its two-phase semantics. For the reasons discussed above, we do not consider its standard semantics but a slight variation thereof. Moreover, we provide a time-stamped semantics of a hybrid automaton, and show how this semantics is related to the two-phase semantics discussed in this paper.

The reason for giving both a two-phase semantics and a time-stamped semantics for timed and hybrid automata is to allow for a smooth translation from timed automata and hybrid automata to $\mu\text{CRL}_t^\bullet$. The translation is discussed in section 5.

It must be noted that all results in this and subsequent sections remain valid when we consider the standard two-phase semantics of timed and hybrid automata, instead of the variation discussed in this paper.

3.1 Timed Automata

The formalism of timed automata was first introduced by Alur and Dill in [3]. Its definition provides a simple and general way for annotating state-transition graphs with timing constraints using finitely many real-valued *clock variables*. Various extensions of this formalism have been defined, most notably an extension with invariants (see e.g. Timed Safety Automata [17]). In this paper, we will discuss timed automata and hybrid automata extended with a more generalised notion of invariants.

In a timed automaton, timing constraints are typically modelled using a notion of *clocks*. The timing constraints are restricted to predicates over these clocks, and hence are called *clock constraints*.

Definition 3.1 (*Clocks, Valuations and Clock Constraints*).
A clock is a variable that can assume values in the set of non-negative real numbers, represented by the set $\mathbb{T}$. Henceforth, we will assume a (finite) set of clocks $\mathcal{C}$.

A clock constraint is a conjunction of atomic constraints; each of these atomic constrains compares clock values with time constants. Clock constraints are thus timing constraints on one or more clocks. Formally, the set $\Phi(\mathcal{C})$ of *clock constraints* φ is defined by the grammar

$$\varphi := c \sim v \mid (c_1 - c_2) \sim v \mid (c_1 + c_2) \sim v \mid \varphi_1 \wedge \varphi_2$$

Here, c_1, c_2 are clocks in the set $\mathcal{C}$, v is a constant in $\mathbb{T}$ and $\sim \in \{<, \leq, >, \geq\}$.

A valuation, or interpretation of clocks assigns a non-negative real value to the clocks, i.e. a valuation is a mapping $\vartheta : \mathcal{C} \to \mathbb{T}$, such that for every clock $c \in \mathcal{C}$ $\vartheta(c) \in \mathbb{T}$. The set of all valuations for clocks is denoted $\mathbb{V}$.

Satisfaction of a clock constraint in a valuation ϑ is defined inductively by a set of rules (where again $\sim \in \{<, \leq, >, \geq\}$) (see Table 1). If, for a given valuation ϑ a clock constraint φ is satisfied, we write $\vartheta \models \varphi$. When a clock constraint φ is not satisfied, we write $\vartheta \not\models \varphi$.

$$\frac{\vartheta(c) \sim v}{\vartheta \models c \sim v} \qquad \frac{(\vartheta(c_1) - \vartheta(c_2)) \sim v}{\vartheta \models (c_1 - c_2) \sim v} \qquad \frac{(\vartheta(c_1) + \vartheta(c_2)) \sim v}{\vartheta \models (c_1 + c_2) \sim v} \qquad \frac{\vartheta \models \varphi_1 \quad \vartheta \models \varphi_2}{\vartheta \models \varphi_1 \wedge \varphi_2}$$

Table 1: *Deduction Rules for Clock Constraints*

We allow ourselves to write $\vartheta, \chi \models \varphi$ when we have both $\vartheta \models \varphi$ and $\chi \models \varphi$. Whenever we want to reset a subset λ of the set of all clocks $\mathcal{C}$, we use the notation $\vartheta[\lambda := \mathbf{0}]$, meaning that for all $c \in \lambda$ we have $\vartheta[\lambda := \mathbf{0}](c) = \mathbf{0}$ and for all $c \in \mathcal{C} \setminus \lambda$ we have $\vartheta[\lambda := \mathbf{0}](c) = \vartheta(c)$.

Often, the constants to which clocks are compared are non-negative rationals. Restricting comparison of clocks with non-negative rationals only, allows for decidability results (see e.g. [3]). If decidability is not an issue, no restrictions are necessary. In this paper, we assume clocks can be compared with any element in the set of non-negative real numbers. A desirable property of clock constraints is the *interpolation* property.

Property 3.1 (Interpolation).
Let ϑ be a valuation for the clocks in $\mathcal{C}$ and let φ be a clock constraint in $\Phi(\mathcal{C})$. Suppose we have $\vartheta \models \varphi$ and we have, for some $d \geq \mathbf{0}$, $\vartheta + d \models \varphi$. Then, the interpolation property says that we also have $(\vartheta + d') \models \varphi$ for any $\mathbf{0} \leq d' \leq d$.

Remark 3.1. here, we consider constants also as functions; hence, the addition in $\vartheta + d$ is the addition as defined on functions, i.e. for all $c \in \mathcal{C}$ we have $(\vartheta + d)(c) = \vartheta(c) + d$.

A formal definition of a timed automaton is provided by Def. 3.2.

Definition 3.2 (*Timed Automaton*).
A timed automaton X is a tuple $\langle L, L^0, \Sigma, \mathcal{C}, \iota, E\rangle$,where

- L is a finite set of *locations*,
- $L^0 \subseteq L$ is a set of *initial* locations,
- Σ is a finite set of labels,
- $\mathcal{C}$ is a finite set of clocks,
- $\iota : L \to \Phi(\mathcal{C})$ is a mapping that labels each location l in L with some clock constraint in $\Phi(\mathcal{C})$,
- $E \subseteq L \times \Sigma \times \Phi(\mathcal{C}) \times 2^{\mathcal{C}} \times L$ is a set of *switches*.

We write $l \xrightarrow{\sigma,\varphi,\lambda} l'$ for $(l, \sigma, \varphi, \lambda, l') \in E$.

Conceptually, a timed automaton is a graph, where a node is called a *location* and each edge is called a *switch*. Like two-phase transition systems or time-stamped transition systems, edges are decorated with additional information. The intuition behind a switch $l \xrightarrow{\sigma,\varphi,\lambda} l'$ is that whenever the clock constraint φ is valid, the action σ is enabled, and upon execution of the action the clocks in the set λ are simultaneously reset to zero. The clock constraint φ is often referred to as the *jump condition*.

As already mentioned in the beginning of this section, various definitions of timed automata exist (e.g. [3, 17, 25]). The definition of timed automata presented in this section is closely related to Timed Safety Automata [17]. A difference is that we do not restrict our clocks to be *past closed*. Past closedness is the property that if a clock constraint φ is valid at a certain point in time, it is also valid for all points prior to that time.

A formal interpretation, or semantics, of timed automata is presented in Def. 3.3. There, a translation of a timed automaton to a two-phase transition system is given. Instead of the standard relative time interpretation, an absolute time semantics is provided.

In order to uniquely represent the states of the two-phase transition systems, both the location of the timed automaton and the valuation for the clocks are encoded in the state. A third parameter is necessary to register the global time; this clock can be thought of as a clock which is reset only at the start of the system. Note that this third parameter can be omitted in case of a relative time interpretation.

Definition 3.3 (*Two-Phase Semantics of Timed Automata*).
Let $X = \langle L, L^0, \Sigma, \mathcal{C}, \iota, E\rangle$ be a timed automaton. The semantics of X is given by the two-phase labelled transition system $X^s = \langle L \times \mathbb{V} \times \mathbb{T}, L^0 \times \{\vartheta_{\mathbf{0}}\} \times \{\mathbf{0}\}, \Sigma, \mathbb{T}, \longrightarrow, \longmapsto\rangle$, where $\vartheta_{\mathbf{0}}(c) = \mathbf{0}$ for all $c \in \mathcal{C}$. The relations $\longrightarrow$ and $\longmapsto$ are defined as the least relations satisfying the rules of Table 2.

$$\frac{\vartheta, \vartheta + (u - t) \models \iota(l) \quad t \leq u}{(l, \vartheta, t) \overset{u}{\longmapsto} (l, \vartheta + (u - t), u)}$$

$$\frac{l \xrightarrow{\sigma,\varphi,\lambda} l' \quad \vartheta \models \varphi \quad \vartheta \models \iota(l) \quad \vartheta' = \vartheta[\lambda := 0] \quad \vartheta' \models \iota(l')}{(l, \vartheta, t) \xrightarrow{\sigma} (l', \vartheta', t)}$$

Table 2: *Rules for the Two-Phase Semantics*

The state-space of X^s consists of the set of states that can be reached from the initial states.

Remark 3.2. The interpretation given here allows for multiple events in zero time. Not all interpretations allow for so-called urgent actions. Most notably, the original definition of Alur and Dill [3] requires the passage of time between two subsequent action transitions.

In this paper, we deviate slightly from the standard semantics for timed automata as given above. This is done for two reasons. First, the notion of an invariant (e.g. as used in [6, 19]) is better served this way. Invariants are intended as properties of a system that are invariably true during state-changes, and moreover, invariants are assumed to hold initially. Whenever an invariant is violated, this means the system has not been specified consistently. The standard semantics of timed automata uses invariants to "remove" erroneous behaviours in a specification, i.e. inconsistencies in specifications can thus go by unnoticed (see e.g. example 3.1). Second, the alternative semantics supports a more state-oriented way of reasoning about timed and hybrid automata. This is useful when modelling complex systems.

The alternative semantics differs from the standard semantics in the omission of a "look-ahead" premise (i.e. the premise $\vartheta' \models \iota(l')$ for action transitions). In the standard semantics, this premise is needed to ensure no transitions are taken to states that violate their invariant upon arrival. Omitting this premise introduces the notion of a *deadlocked* state, i.e. a state in which no time can pass, nor actions can be executed. This alternative semantics, employed in this paper is formalised by Def. 3.4.

Definition 3.4 (*Alternative Two-Phase Semantics of Timed Automata*).
Let $X = \langle L, L^0, \Sigma, \mathcal{C}, \iota, E\rangle$ be a timed automaton. The semantics of X is given by the two-phase labelled transition system $X^s = \langle L \times \mathbb{V} \times \mathbb{T}, L^0 \times \{\vartheta_{\mathbf{0}}\} \times \{\mathbf{0}\}, \Sigma, \mathbb{T}, \longrightarrow, \longmapsto\rangle$, where $\vartheta_{\mathbf{0}}(c) = \mathbf{0}$ for all $c \in \mathcal{C}$. The relations $\longrightarrow$ and $\longmapsto$ are defined as the least relations satisfying the rules depicted in Table 3.

The state-space of X^s consists of the set of states that can be reached from the initial states.

Remark 3.3. Whenever we refer to the two-phase semantics of a timed automaton X as defined according to Table 3, we will write X^s.

The interpretation of a timed automaton changes accordingly. This difference in interpretations is exemplified in Example 3.1.

$$\frac{\vartheta, \vartheta + (u - t) \models \iota(l) \quad t \leq u}{(l, \vartheta, t) \stackrel{u}{\longmapsto} (l, \vartheta + (u - t), u)}$$

$$\frac{l \xrightarrow{\sigma,\varphi,\lambda} l' \quad \vartheta \models \varphi \quad \vartheta \models \iota(l) \quad \vartheta' = \vartheta[\lambda := 0]}{(l, \vartheta, t) \xrightarrow{\sigma} (l', \vartheta', t)}$$

Table 3: *Rules for the Alternative Two-Phase Semantics*

Example 3.1. Let X be a three-state timed automaton with two clocks (see Fig. 1). Transitions can be made within 5 seconds starting from state R to either state Q or state S. The system arrives at state S, by performing a shutdown. Whenever the global clock Y is at most 50 seconds, there is still a possibility of rebooting the system, while after 50 seconds, the system cannot be restarted and forever "hangs".

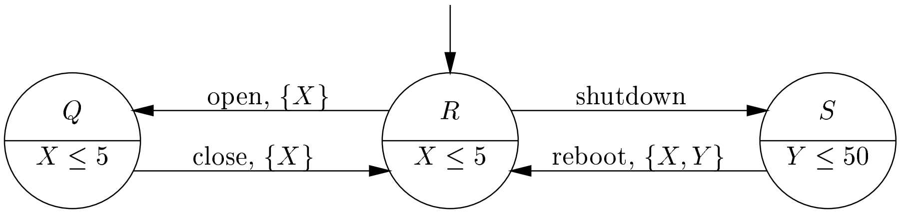

Figure 1: *Limited-lifespan system*

In the above example, the standard semantics induces a transition system that causes the system not to shut-down, once the system's internal clock has been running for more than 50 seconds. Using the standard semantics, one can prove the formula $\Box(\text{shutdown} \Rightarrow (\Diamond\text{reboot}))$. Hence, this system must be interpreted as a system that can always be rebooted.

The semantics employed in this paper still allows a system to shut-down, but not restart once the system's internal clock has been running for more than 50 seconds. Hence, the temporal formula $\Box(\text{shutdown} \Rightarrow (\Diamond\text{reboot}))$ no longer holds.

In order to be able to make a comparison between different interpretations of automata, a notion of equivalence is needed. In this paper, we discriminate timed automata using either timed bisimulation or timed strong bisimulation, depending on the interpretation. For the two-phase transition system interpretation, this is formalised in Def. 3.5.

Bisimulation is not the only means of discriminating between timed automata. Other notions of equivalence are also possible [8], e.g. isomorphism, symbolic bisimulation, trace equivalence, etc. In this paper we will restrict ourselves to the notion of bisimulation. This is done since bisimulation is the most discriminating notion of equivalence that differentiates two timed automata on the basis of their behaviour instead of on their syntax.

Definition 3.5 (*Two-phase Equivalence of Timed Automata*).
Let X_1 and X_2 be two timed automata, and let X_1^s and X_2^s be the interpretations of X_1 and X_2 in terms of two-phase transition systems. We will consider X_1 and X_2 two-phase equivalent, denoted by $X \equiv^\iota X_2$ iff there exists a relation R, such that $R : X_1^s \underline{\leftrightarrow}{}^{\iota}_{t} X_2^s$.

A consequence of this definition is that whenever two timed automata are two-phase equivalent, time-stamps of related states are equivalent. This is formalised in lemma 3.1.

Lemma 3.1 (*Synchronisation on Time-stamps*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \iota_2, E_2\rangle$ be two arbitrary timed automata, such that $X_1 \equiv^\iota X_2$. Let R be such that $R : X_1^s \leftrightarrow_t^\iota X_2^s$. Then, for each state (l, ϑ, t) of X_1^s and (s, χ, u) of X_2^s:

$$(l, \vartheta, t)\ R\ (s, \chi, u) \Rightarrow u = t$$

Proof. The proof is by induction on the rules of the transition system. □

In this paper, we relate both timed and hybrid automata on the semantical level with $\mu\mathrm{CRL}_t^\bullet$; hence, we need to provide a semantics for all formalisms, all formulated in terms of one semantical model. Since timed and hybrid automata are based on two-phase transition systems and $\mu\mathrm{CRL}_t^\bullet$ is based on a time-stamped transition system, an alternative semantics for either timed and hybrid automata or $\mu\mathrm{CRL}_t^\bullet$ must be provided. The easiest way is to formulate an equivalent time-stamped semantics of a timed automaton. Alternatively, the semantics of $\mu\mathrm{CRL}_t^\bullet$ can be rewritten to a two-phase semantics. This, however, is considered to be quite a laborious task, since the semantics of $\mu\mathrm{CRL}_t^\bullet$ is rather extensive.

Definition 3.6 (*Time-Stamped Semantics of Timed Automata*).
Let $X = \langle L, L^0, \Sigma, \mathcal{C}, \iota, E\rangle$ be a timed automaton. The semantics of X is given by the time-stamped transition system $X^{\overline{s}} = \langle L \times \mathbb{V} \times \mathbb{T}, L^0 \times \{\vartheta_{\mathbf{0}}\} \times \{\mathbf{0}\}, \Sigma, \mathbb{T}, \longrightarrow, \mathcal{U}\rangle$, where $\vartheta_{\mathbf{0}}(c) = \mathbf{0}$, for all $c \in \mathcal{C}$. The relations $\longrightarrow$ and $\mathcal{U}$ are defined as the least relations satisfying the rules of Table 4.

$$\frac{\mathcal{U}_u(l,\vartheta,t) \quad u' \leq u}{\mathcal{U}_{u'}(l,\vartheta,t)} \qquad \frac{\vartheta, \vartheta + (u-t) \models \iota(l) \quad t \leq u}{\mathcal{U}_u(l,\vartheta,t)}$$

$$\frac{l \xrightarrow{\sigma,\varphi,\lambda} l' \quad \vartheta, \vartheta + (u-t) \models \iota(l) \quad \vartheta + (u-t) \models \varphi \quad \vartheta' = (\vartheta + (u-t))[\lambda := 0] \quad t \leq u}{(l,\vartheta,t) \xrightarrow{\sigma}_u (l',\vartheta',u)}$$

Table 4: *Rules for the Time-Stamped Transition System Semantics*

The state-space of $X^{\overline{s}}$ consists of the set of states that can be reached from the initial states.

Remark 3.4. Whenever we refer to the time-stamped semantics of a timed automaton X, we will write $X^{\overline{s}}$.

When we need to prove properties over the delay predicate, we need to make case distinctions to which of the two delay predicate rules hold. Lemma 3.2, however, states that we only need to consider one of both rules to prove a property.

Lemma 3.2 (*Upper Bound*).
Let X be a timed automaton and let (l, ϑ, t) be a state in $X^{\overline{s}}$. Then the following property holds:

$$\mathcal{U}_t(l,\vartheta,t) \Leftrightarrow (\forall_{u<t}\ \mathcal{U}_u(l,\vartheta,t))$$

Proof. The implication from left to right follows immediately from the rules of the delay predicate, so we need only to prove the reverse implication. Assume $\forall_{u<t}\ \mathcal{U}_u(l,\vartheta,t)$ and suppose $\neg\mathcal{U}_t(l,\vartheta,t)$. Given $\neg\mathcal{U}_t(l,\vartheta,t)$, we know that $\vartheta \not\models \iota(l)$. Hence, it cannot be the case that $\mathcal{U}_{t'}(l,\vartheta,t)$ for some $t' \geq t$. Therefore, it cannot be the case that $\mathcal{U}_{\mathbf{0}}(l,\vartheta,t)$. Hence, our assumption must be wrong and thus $\mathcal{U}_t(l,\vartheta,t)$ holds. □

The notion of when two timed automata are equivalent is dependent on the underlying transition system. A suitable equivalence between timed automata, based on a time-stamped semantics, is defined in Def. 3.7.

Definition 3.7 (*Time-stamped Equivalence of Timed Automata*).
Let X_1 and X_2 be two timed automata, and $X_1^{\overline{s}}$ and $X_2^{\overline{s}}$ the interpretations of X_1 and X_2 in terms of time-stamped transition systems. The timed automata X_1 and X_2 are considered time-stamped equivalent, denoted by $X_1 \equiv X_2$ iff there exists a relation R, such that $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$.

This notion of time-stamped equivalence induces the property that whenever two states of a system are bisimilar, the time-stamps of these states are equivalent. This is expressed in Lemma 3.3.

Lemma 3.3 (*Synchronisation on Time-stamps*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \iota_2, E_2\rangle$ be two arbitrary timed automata, such that $X_1 \equiv X_2$. Let R be such that $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$. Then, for each state (l, ϑ, t) of $X_1^{\overline{s}}$ and (s, χ, u) of $X_2^{\overline{s}}$,

$$(l, \vartheta, t)\ R\ (s, \chi, u) \Rightarrow u = t$$

Proof. The proof is by induction on the rules of the transition system. □

Thus far, we have introduced three different interpretations of the behaviour of a timed automaton. Two of these (Def. 3.3 and 3.4) are based on a two-phase transition system while the other is based on a time-stamped transition system. However, the two kinds of equivalences that have been defined turn out to be equally discriminative, as is formalised in Theorem 3.1. This means that it does not matter which underlying model for defining the semantics is used. Note that equivalence based on the standard semantics is different, as is already exemplified by Example 3.1.

Theorem 3.1 (*Corresponding Interpretations*).
Let X_1 and X_2 be two timed automata. We have the following equivalence:

$$X_1 \ \equiv^{\iota} X_2 \ \Leftrightarrow \ X_1 \equiv X_2$$

Proof. If $R : X_1^s \ \underline{\leftrightarrow}_t^{\iota} X_2^s$, then it is not hard to check that also $R : X_1^{\overline{s}} \ \underline{\leftrightarrow}_t X_2^{\overline{s}}$.
If $R : X_1^{\overline{s}} \ \underline{\leftrightarrow}_t X_2^{\overline{s}}$, then also $R' : X_1^s \ \underline{\leftrightarrow}_t^{\iota} X_2^s$, where
$R' = R \cup \{\langle (l, \vartheta + d, t + d), (s, \chi + d, r + d)\rangle | (l, \vartheta, t) R (s, \chi, r) \wedge \mathcal{U}_{t+d}(l, \vartheta, t) \wedge \mathbf{0} \leq d\}$
It is easy to check that R' is a bisimulation relation. □

Timed automata are very useful for modelling real-time behaviour of simple components. However, large systems, composed of several components are still very hard to model.
Various ways exist to overcome this problem. The standard way is to introduce auxiliary operators defining some form of composition (e.g. alternative composition or sequential composition). Since timed automata are often used in the realm of communicating or *synchronising* systems, there is an apparent need for expressing parallelism.
Parallel composition is considered to be an auxiliary operator, i.e. every occurrence of parallelism in a model can be removed. The rules for removing parallelism are presented in Def. 3.8.

Definition 3.8 (*Synchronisation*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be timed automata, such that $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. The parallel composition of X_1 and X_2, denoted by $X_1 \|_s X_2$ is the timed automaton $\langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, \mathcal{C}_1 \cup \mathcal{C}_2, \iota, E\rangle$, where $\iota(l, s) = \iota_1(l) \wedge \iota_2(s)$ and the set of transitions E is defined as the least relation satisfying the rules of Table 5.

$$\frac{l \xrightarrow{\sigma,\varphi,\lambda} l' \quad s \xrightarrow{\sigma,\psi,\kappa} s'}{(l,s) \xrightarrow{\sigma,\varphi\wedge\psi,\lambda\cup\kappa} (l',s')} \qquad \frac{l \xrightarrow{\sigma,\varphi,\lambda} l' \quad \sigma \in \Sigma_1 \setminus \Sigma_2}{(l,s) \xrightarrow{\sigma,\varphi,\lambda} (l',s)} \qquad \frac{s \xrightarrow{\sigma,\psi,\kappa} s' \quad \sigma \in \Sigma_2 \setminus \Sigma_1}{(l,s) \xrightarrow{\sigma,\psi,\kappa} (l,s')}$$

Table 5: *Rules for Synchronisation*

The parallel operator models *blocking synchronisation on shared labels*, i.e. for all shared actions σ, a component can execute action σ only if the other component(s) can also do so. The parallel operator in some sense combines the encapsulation of actions with the execution of actions. The semantics of the parallel operator are given only in terms of time-stamped transition systems.

Definition 3.9 (*Semantics of Synchronisation*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be timed automata. The semantics of $X_1\|_s X_2$ is given by the time-stamped transition system $(X_1\|_s X_2)^{\overline{s}} = \langle (L_1 \times \mathbb{V} \times \mathbb{T}) \times (L_2 \times \mathbb{V} \times \mathbb{T}), (L_1^0 \times \{\vartheta_\mathbf{0}\} \times \{\mathbf{0}\}) \times (L_2^0 \times \{\chi_\mathbf{0}\} \times \{\mathbf{0}\}), \Sigma_1 \cup \Sigma_2, \mathbb{T}, \longrightarrow, \mathcal{U}\rangle$, where $\vartheta_\mathbf{0}(c_1) = \mathbf{0}$ and $\chi_\mathbf{0}(c_2) = \mathbf{0}$ for all $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$. For all states $(l,\vartheta,t) \in X_1^{\overline{s}}$ and $(s,\chi,u) \in X_2^{\overline{s}}$, the set of transitions $\longrightarrow$ and the predicate $\mathcal{U}$ are defined as the least relations satisfying the rules of Table 6.

$$\frac{\mathcal{U}_r(l,\vartheta,t) \quad \mathcal{U}_r(s,\chi,u)}{\mathcal{U}_r((l,\vartheta,t)\|_s(s,\chi,u))}$$

$$\frac{(l,\vartheta,t) \xrightarrow{\sigma}_r (l',\vartheta',r) \quad \mathcal{U}_r(s,\chi,u) \quad \sigma \in \Sigma_1 \setminus \Sigma_2 \quad u \uparrow t \le r}{(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_r (l',\vartheta',r)\|_s(s,\chi,u)}$$

$$\frac{(s,\chi,u) \xrightarrow{\sigma}_r (s',\chi',r) \quad \mathcal{U}_r(l,\vartheta,t) \quad \sigma \in \Sigma_2 \setminus \Sigma_1 \quad u \uparrow t \le r}{(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_r (l,\vartheta,t)\|_s(s',\chi',r)}$$

$$\frac{(l,\vartheta,t) \xrightarrow{\sigma}_r (l',\vartheta',r) \quad (s,\chi,u) \xrightarrow{\sigma}_r (s',\chi',r)}{(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_r (l',\vartheta',r)\|_s(s',\chi',r)}$$

Table 6: *Rules for the Synchronisation Operator*

Remark 3.5. Note that we write $u \uparrow t \le r$ when we mean $u \le r$ and $t \le r$.

The rules of Table 5 define how parallel composition can be eliminated using the transition relations of two timed automata. The semantics of synchronisation is defined in Def. 3.9 in terms of time-stamped transition systems. Hence, the rules defined by Def. 3.8 should be sound and complete with respect to the rules defined by Def. 3.9. This is expressed by Theorem 3.2.

Theorem 3.2 (*Soundness and Completeness of Synchronisation*).
Let X_1 and X_2 be timed automata. Let $X_1^{\overline{s}}$ be the interpretation of X_1 and $X_2^{\overline{s}}$ be the interpretation of X_2 and $(X_1\|_s X_2)^{\overline{s}}$ be the interpretation of $X_1\|_s X_2$. Then, there exists a bisimulation R, such that $R : (X_1\|_s X_2)^{\overline{s}} \;\underline{\leftrightarrow}_t\; X_1^{\overline{s}} \|_s X_2^{\overline{s}}$.

Proof. Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be timed automata. Then, for each $l \in L_1, s \in L_2, \vartheta, \chi \in \mathbb{V}, t, u \in \mathbb{T}$, the relation R is defined as the set

$$\begin{array}{rcl} R & = & \{\langle (l,\vartheta,t)\|_s(s,\chi,t)\ ,\ ((l,s),\vartheta \cup \chi, t)\rangle\ \} \\ & \cup & \{\langle (l,\vartheta,t)\|_s(s,\chi,u)\ ,\ ((l,s),\vartheta \cup (\chi + t - u), t)\rangle\ | u \leq t \wedge \chi \models \iota_2(s)\} \\ & \cup & \{\langle (l,\vartheta,t)\|_s(s,\chi,u)\ ,\ ((l,s),(\vartheta + u - t) \cup \chi, t)\rangle\ | t \leq u \wedge \vartheta \models \iota_1(l)\} \end{array}$$

It is easy to check that R is a timed bisimulation relation such that $R : (X_1\|_s X_2)^{\overline{s}} \ \underline{\leftrightarrow}_t\ X_1^{\overline{s}} \ \|_s\ X_2^{\overline{s}}$.
□

3.2 Hybrid Automata

Hybrid Automata have first been proposed in [2]. A hybrid automaton is a formal model for a mixed discrete-continuous system, and can be seen as a generalisation of a timed automaton, in which clocks are no longer constrained to progress at a fixed rate. Typical examples of mixed discrete-continuous systems are digital controllers of analog plants.

In continuous systems, *continuous variables* are used to model the essential dynamics of the system. By means of constraints on these continuous variables, e.g. inequalities, the systems behaviour is modelled. This way of modelling the dynamics of the continuous part of a system has been adopted by the model of hybrid automata. Constraints over continuous variables are henceforth referred to as *variable constraints*. Moreover, constraints over the first derivatives of continuous variables are called *flow constraints*.

Definition 3.10 (*Variable Constraints, Flow Constraints and Reset Constraints*).
A continuous variable is a variable that can assume any value in the set of real numbers and is thought to change continuously in time. Henceforth, we will assume a (finite) set of continuous variables $\mathcal{V}$.

A snapshot valuation of continuous variables assigns a real value to continuous variables, i.e. a valuation is a mapping $\vartheta : \mathcal{V} \to \mathbb{R}$, such that for every variable $v \in \mathcal{V}$, $\vartheta(v) \in \mathbb{R}$. The set of all snapshot valuations is denoted by $\mathbb{V}$. A valuation of continuous variables is an infinite set of snapshot valuations, described by a (partial) mapping $f : \mathbb{T} \to \mathcal{V} \to \mathbb{R}$.

A variable constraint is a predicate on a set of continuous variables. Variable constraints compare continuous variables with values from the set of real numbers $\mathbb{R}$ or with other continuous variables. For a set $\mathcal{V}$ of continuous variables, the set $\Phi(\mathcal{V})$ is the set of variable constraints.

A valuation f satisfies a variable constraint φ, denoted by $f \models \varphi$, iff for all $\varepsilon \in \mathsf{dom}(f)$ and all continuous variables $v \in \mathcal{V}$, the closed predicate $\varphi[f(\varepsilon)(v)/v]$ holds on the real-line. Whenever f does not satisfy a variable constraint φ, we write $f \not\models \varphi$.

A flow constraint is a predicate on a set of continuous variables and their derivatives. For a set $\mathcal{V}$ of continuous variables and their first derivatives $\dot{\mathcal{V}}$, the set $\Theta(\mathcal{V} \cup \dot{\mathcal{V}})$ is the set of flow constraints over the set $\mathcal{V} \cup \dot{\mathcal{V}}$. We will use the predicate $\mathsf{diff}(f)$ to denote that the function f is differentiable on its domain $\mathsf{dom}(f)$.

The valuations f and $\dot{f}$ satisfy a flow constraint θ, denoted by $(f \cup \dot{f}) \models \theta$, iff $\mathsf{diff}(f)$ and for all $\varepsilon \in \mathsf{dom}(f)$ and all continuous variables $v \in \mathcal{V}$ and all first derivatives $\dot{v} \in \dot{\mathcal{V}}$ the closed predicate $\theta[f(\varepsilon)(v)/v][\dot{f}(\varepsilon)(v)/\dot{v}]$ holds on the real-line. Whenever f and $\dot{f}$ do not satisfy a flow constraint φ, we write $(f \cup \dot{f}) \not\models \varphi$.

A reset constraint is a predicate on a set of continuous variables and their updated counterparts. Let $\mathcal{V}'$ be the set of updated counterparts of the set $\mathcal{V}$, then $\Delta(\mathcal{V} \cup \mathcal{V}')$ is the set of reset constraints over the set $\mathcal{V} \cup \mathcal{V}'$.

The snapshot valuations ϑ and ϑ' satisfy a reset constraint ρ, denoted by $(\vartheta \cup \vartheta') \models \rho$, iff for all continuous variables $v \in \mathcal{V}$ and their updated counterparts $v' \in \mathcal{V}'$ the closed predicate $\rho[\vartheta(v)/v][\vartheta'(v)/v']$ holds on the real-line.

If no confusion is possible, we will omit brackets whenever possible, i.e. we will write $f \cup \dot{f} \models \theta$ instead of $(f \cup \dot{f}) \models \theta$. Henceforth, we assume the sets $\Phi(\mathcal{V})$, $\Theta(\mathcal{V} \cup \dot{\mathcal{V}})$ and $\Delta(\mathcal{V} \cup \mathcal{V}')$ are defined according to some rules.

A formal definition of a hybrid automaton is given in Def. 3.11.

Definition 3.11 (*Hybrid Automaton*).
A hybrid automaton X is a tuple $\langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$,where

- M is a finite set of *control modes*,
- $M^0 \subseteq M$ is a set of *initial* control modes,
- Σ is a finite set of labels,
- $\mathcal{V}$ is a finite set of continuous variables,
- $\iota : M \to \Phi(\mathcal{V})$ is a mapping that labels each location m in M with some variable constraint in $\Phi(\mathcal{V})$,
- $\theta : M \to \Theta(\mathcal{V} \cup \dot{\mathcal{V}})$ is the flow condition function,
- $\mathcal{I} : M^0 \to \Phi(\mathcal{V})$ is a mapping that assigns to every initial control mode a variable constraint in $\Phi(\mathcal{V})$ expressing the initial values a continuous variable is allowed to have,
- $E \subseteq M \times \Sigma \times \Phi(\mathcal{V}) \times \Delta(\mathcal{V} \cup \mathcal{V}') \times M$ is a set of *control switches*.

We will write $m \xrightarrow{\sigma,\varphi,\rho} m'$ instead of $(m, \sigma, \varphi, \rho, m') \in E$.

Like a timed automaton, a hybrid automaton can be represented as a directed graph. Each node in this graph is called a *control mode*, representing a state in which continuous processes are running, and each directed edge is called a *control switch*. Additional information is again attached to an edge. The intuition behind a transition $m \xrightarrow{\sigma,\varphi,\rho} m'$ is that whenever a variable constraint (also called the *jump condition*) φ is valid, the action σ is enabled. Upon execution of this action σ, the continuous variables are re-initialised according to the reset constraint ρ.

A formal interpretation of hybrid automata in terms of an absolute time, two-phase transition system is presented in Def. 3.12. This definition differs from the standard interpretation of a hybrid automaton only in the absence of a look-ahead premise in an action transition (cf. Def. 3.4 of our definition of a timed automaton). This has no consequences for any of the results discussed in this paper.

The states of the two-phase transition system that is obtained from a hybrid automaton are built from the information of the control-modes, snapshot valuations for the continuous variables and a reference to a global clock. This is comparable to the representation of states in the transition systems for a timed automaton.

Definition 3.12 (*Two-Phase Semantics of a Hybrid Automaton*).
Let $X = \langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$ be a hybrid automaton. The semantics of X is given by the two-phase transition system $X^s = \langle M \times \mathbb{V} \times \mathbb{T}, I, \Sigma, \mathbb{T}, \longrightarrow, \longmapsto\rangle$. The set I of initial states is defined as $I = \{(m, \vartheta, \mathbf{0}) \in M^0 \times \mathbb{V} \times \mathbb{T} \mid \vartheta \models \mathcal{I}(m)\}$, and the relations $\longrightarrow$ and $\longmapsto$ are defined as the least relations satisfying the rules of Table 7.

The operator $\rceil$ is used to denote the domain restriction of a function. For every set R and function f with domain $\mathsf{dom}(f)$, the function $f\rceil R$ is the function f restricted to the domain $\mathsf{dom}(f) \cap R$.

$$\frac{m \xrightarrow{\sigma,\varphi,\rho} m' \quad \vartheta \models \iota(m) \quad \vartheta \models \varphi \quad \vartheta \cup \vartheta' \models \rho}{(m, \vartheta, t) \xrightarrow{\sigma} (m', \vartheta', t)}$$

$$\frac{\begin{array}{c} f : [0, r-t] \to \mathbb{V} \quad f(\mathbf{0}) = \vartheta \quad f(r-t) = \vartheta' \\ \mathsf{diff}(f\rceil(0, r-t)) \quad f \models \iota(m) \quad (f \cup \dot{f})\rceil(0, r-t) \models \theta(m) \quad t \leq r \end{array}}{(m, \vartheta, t) \overset{r}{\longmapsto} (m, \vartheta', r)}$$

Table 7: *Rules for the Two-Phase Semantics*

The state-space of X^s consists of the set of states that can be reached from the initial states

Remark 3.6. *In the deduction rules, only the existence of a continuous, differentiable function is required, while the function itself is not part of the state of the resulting two-phase transition system. The function is often referred to as a witness, and needs not be unique for a transition.*

Analogously to timed automata, we will consider two hybrid automata equivalent if and only if their interpretations are again equivalent. This is formally expressed in Def. 3.13. Note that other ways to relate hybrid automata are also possible.

Definition 3.13 (*Two-phase Equivalence of Hybrid Automata*).
Let X_1 and X_2 be two hybrid automata, and let X_1^s and X_2^s be their two-phase interpretations of resp. X_1 and X_2. We will consider X_1 and X_2 two-phase equivalent, denoted by $X_1 \equiv^\iota X_2$ iff there exists a relation R, such that $R : X_1^s \underline{\leftrightarrow}_t^\iota X_2^s$.

A consequence of this equality is that for two two-phase equivalent hybrid automata, the time-stamps of bisimilar states are equivalent. This is formalised in Lemma 3.4.

Lemma 3.4 (*Synchronisation on Time-stamps*).
Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \iota_1, \theta_1, \mathcal{I}_1, E_1\rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \iota_2, \theta_2, \mathcal{I}_2, E_2\rangle$ be two arbitrary hybrid automata, such that $X_1 \equiv^\iota X_2$. Let R be such that $R : X_1^s \underline{\leftrightarrow}_t^\iota X_2^s$. Then, for each state (m, ϑ, t) of X_1^s and (n, χ, u) of X_2^s,

$$(l, \vartheta, t)\ R\ (n, \chi, u) \ \Rightarrow\ u = t$$

Proof. The proof is by induction on the operational rules. □

The two-phase transition system induced by a hybrid automaton links its states via action transitions and timed transitions. Timed transitions cause a change in the continuous variables, however, this change may not be smooth and uniform. Hence, in successive timed transitions, the growth of continuous variables may be different. This is exemplified in Example 3.2.

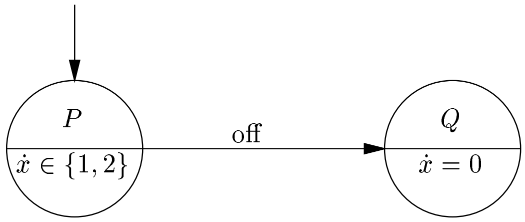

Figure 2: *Example of non-uniform progression*

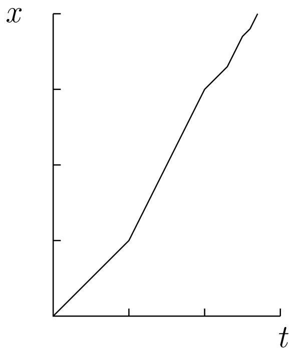

Figure 3: *Possible progression*

Example 3.2.
Let X be a hybrid automaton with two control modes, depicted in Fig. 2. A possible progression (trace) of variable x in control-mode P is depicted graphically in Fig. 3.
The trace of Fig. 3 illustrates how continuous variables can change in a non-uniform way. A reason for this non-uniform change is the non-determinacy that is allowed in flow constraints; hence, the witnesses for two successive timed transitions do not necessarily yield a new differentiable function.

Example 3.2 illustrates the need for a more complex premise when we consider an interpretation of a hybrid automaton in a time-stamped semantics. It turns out that weakening the premise $\mathsf{diff}(f)$ in Def. 3.12 is the solution. Instead of considering only differentiable functions, we are satisfied if the functions are differentiable on their domains except for on a small set of points. This is captured by the definition of *differentiable almost everywhere.*

Definition 3.14 (*Differentiable almost everywhere*).
A function f is said to be differentiable almost everywhere, denoted by $\mathsf{dae}(f)$, whenever the set of elements for which f is not differentiable has measure zero.

Essentially, we are only interested in a subset of all differentiable almost everywhere functions. Most notably, only the functions that are continuous on a closed interval are considered.

Definition 3.15 (*The Set of Continuous, Differentiable almost everywhere Functions*).
Let $\mathcal{D}_d^D$ be the set consisting of all continuous functions with domain $[0, d]$, $0 \leq d$ and range D, such that every function in $\mathcal{D}_d^D$ is differentiable almost everywhere, i.e.

$$\mathcal{D}_d^D \ := \ \{f : [0, d] \to D \mid \mathsf{cont}(f) \wedge \mathsf{dae}(f)\}$$

For any function $f \in \mathcal{D}_d^D$, let ω_f be the set of elements for which f is differentiable.

Note that every function $f \in \mathcal{D}_d^D$ can be characterised as a continuous function that consists of a sequence of smaller functions that are differentiable. The reverse also is true; for every interval for which a continuous sequence of piece-wise differentiable functions exists, the sequence can be transformed to a function that is also in the set $\mathcal{D}_d^D$ for some d and D.
Definition 3.16 presents an interpretation of a hybrid automaton in terms of a time-stamped transition system.

Definition 3.16 (*Time-stamped Semantics of a Hybrid Automaton*).

$\dfrac{\mathcal{U}_u(m,\vartheta,t) \quad u' \leq u}{\mathcal{U}_{u'}(m,\vartheta,t)}$	$\dfrac{f \in \mathcal{D}^{\mathbb{V}}_{u-t} \quad f(\mathbf{0}) = \vartheta \quad f \models \iota(m) \quad (f \cup \dot{f})]\omega_f \models \theta(m) \quad t \leq u}{\mathcal{U}_u(m,\vartheta,t)}$

$$\frac{\begin{array}{c} f \in \mathcal{D}^{\mathbb{V}}_{u-t} \quad f(\mathbf{0}) = \vartheta \quad f \models \iota(m) \quad (f \cup \dot{f})]\omega_f \models \theta(m) \\ m \xrightarrow{\sigma,\varphi,\rho} m' \qquad f(u-t) \models \varphi \quad f(u-t) \cup \vartheta' \models \rho \quad t \leq u \end{array}}{(m,\vartheta,t) \xrightarrow{\sigma}_u (m',\vartheta',u)}$$

Table 8: *Rules for the Timed Transition System*

Let $X = \langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$ be a hybrid automaton. The semantics of X is given by the time-stamped transition system $X^{\overline{s}} = \langle M \times \mathbb{V} \times \mathbb{T}, I, \Sigma, \mathbb{T}, \longrightarrow, \mathcal{U}\rangle$. The set I of initial states is defined as $I = \{(m,\vartheta,\mathbf{0}) \in M^0 \times \mathbb{V} \times \mathbb{T} \mid \vartheta \models \mathcal{I}(m)\}$, and the relations $\longrightarrow$ and $\mathcal{U}$ are defined as the least relations satisfying the rules of Table 8.

The state-space of $X^{\overline{s}}$ consists of the states that can be reached from the initial states

In this paper, we relate hybrid automata by means of time-stamped equivalence. Like time-stamped equivalence between two timed automata, time-stamped equivalence between two hybrid automata is defined by means of the existence of an bisimulation relation between their time-stamped interpretations. This is expressed in Def. 3.17.

Definition 3.17 (*Time-stamped Equivalence of Hybrid Automata*).
Let X_1 and X_2 be two hybrid automata, and let $X_1^{\overline{s}}$ and $X_2^{\overline{s}}$ be the time-stamped interpretations of resp. X_1 and X_2. We will consider X_1 and X_2 time-stamped equivalent, denoted by $X_1 \equiv X_2$ iff there exists a relation R, such that $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$.

When we consider two hybrid automata to be time-stamped equivalent, the states that are related via the equivalence relation carry the exact same time-stamp. This is formalised in Lemma 3.5.

Lemma 3.5 (*Synchronisation on Time-stamps*).
Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \iota_1\theta_1, \mathcal{I}_1, E_1\rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \iota_2, \theta_2, \mathcal{I}_2, E_2\rangle$ be two arbitrary hybrid automata, such that $X_1 \equiv X_2$. Let R be such that $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$. Then, for each state (m,ϑ,t) of $X_1^{\overline{s}}$ and (n,χ,u) of $X_2^{\overline{s}}$,

$$(l,\vartheta,t)\ R\ (n,\chi,u) \ \Rightarrow\ u = t$$

Proof. The proof is by induction on the rules of the semantics. □

When we need to prove properties over the delay predicate, we are forced to make case distinctions to which of the two delay predicate rules hold. The following Lemma states that we need to consider only one of both rules.

Lemma 3.6 (*Upper Bound*).
Let X be a hybrid automaton and let (m,ϑ,t) be a state in $X^{\overline{s}}$. Then the following property holds:

$$\mathcal{U}_t(m,\vartheta,t) \ \Leftrightarrow\ (\forall_{u<t}\, \mathcal{U}_u(m,\vartheta,t))$$

Proof. The proof is similar to the proof of Lemma 3.2. □

Thus far we have provided two different interpretations for a hybrid automaton. Unlike for a timed automaton, it does matter which interpretation is chosen. In fact, the two-phase semantics (which is close to the standard semantics) is more discriminative. Example 3.3 provides an intuition behind the reason for this difference. It can also be proved that no equally discriminative time-stamped interpretation for a hybrid automaton exists. However, there is a correspondence between both interpretations, expressing that if a system can perform an action at a specific point in time in a two-phase semantics, it can also perform this action at that point in time in the time-stamped semantics and vice versa.

Example 3.3.
Let X_1 and X_2 be two hybrid automata, depicted in Fig. 4. Both have merely one control mode and no control switches. Then $X_1 \equiv X_2$, but $X_1 \not\equiv^\iota X_2$.

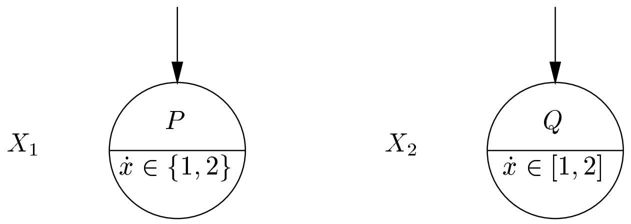

Figure 4: *Trivial Hybrid Automata*

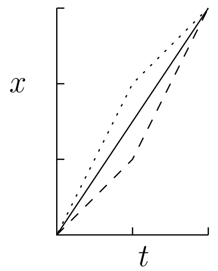

Figure 5: *Different Propagations*

The fact that $X_1 \equiv X_2$ can be easily seen. We will show why $X_1 \not\equiv^\iota X_2$. Let $(p, \mathbf{0}, \mathbf{0})$ be the initial state of X_1^s and let $(q, \mathbf{0}, \mathbf{0})$ be the initial state of X_2^s. Then system X_2^s can immediately take a time transition to state $(q, 3, 2)$, whereas system X_1^s can only reach this state via at least one intermediate state, e.g. state $(q, 1, 1)$ or state $(q, 2, 1)$ (see Fig. 5).

The difference in both semantics can be summarised as follows. For an arbitrary time-stamped transition system, time is reflexive-transitive closed under the predicate $\mathcal{U}$. For a two-phase transition system, this property needs to be enforced. Definition 3.18 defines rules such that an arbitrary two-phase transition system adheres to the time-reflexive-transitive closure property.

Note that the time-reflexive-transitive closure property in general does not hold for a two-phase transition system induced by a hybrid automaton. However, there are various classes of hybrid automata for which this property does hold. Linear hybrid automata [4] and rectangular automata [16] both fall into this category. Note that in [15], the time-reflexive-transitive closure is also used. There, it is called a *stutter-closed flow* relation. Even so, it is rather puzzling that this property is not part of the general framework of hybrid automata, as most formalisms regard this property as a basic requirement on a system's timed behaviour.

Definition 3.18 (*Time-reflexive-transitive Closure of a Two-phase Transition System*).
Let $X = \langle Q, Q^0, \Sigma, \mathbb{T}, \longrightarrow, \longmapsto \rangle$ be a two-phase transition system. The time-reflexive-transitive closure of X is denoted $[X]$, where $[X] = \langle Q, Q^0, \Sigma, \mathbb{T}, \longrightarrow, \models\!\!\Rightarrow \rangle$. The relation $\models\!\!\Rightarrow$ is defined as the least relation satisfying the rules in Table 9.

In this paper, we will not extend the two-phase semantics of a hybrid automaton such that it adheres to the time-reflexive-transitive closure property. Based on Def. 3.18 a slightly less discriminating definition of equality of hybrid automata is provided (see Def. 3.19).

$$
\frac{q \overset{t}{\longmapsto} q'}{q \overset{t}{\models\!\!\Longrightarrow} q'}
\qquad
\frac{q \overset{t}{\models\!\!\Longrightarrow} q'' \quad q'' \overset{u}{\models\!\!\Longrightarrow} q' \quad t \leq u}{q \overset{u}{\models\!\!\Longrightarrow} q'}
$$

Table 9: *Rules for the Time-reflexive-transitive Closure*

Definition 3.19 (*Weak Two-phase Equivalence of Hybrid Automata*).
Let X_1 and X_2 be two hybrid automata, and let X_1^s and X_2^s be the two-phase interpretations of resp. X_1 and X_2 We will consider X_1 and X_2 weak two-phase equivalent, denoted by $X_1 \equiv_w^{\iota} X_2$ iff there exists a relation R, such that $R : [X_1^s] \underline{\leftrightarrow}_t^{\iota} [X_2^s]$.

Note that when we restrict to the class of hybrid automata that adhere to the time-reflexive-transitive closure property, two-phase equivalence and weak two-phase equivalence are the same.

Weak two-phase equivalence turns out to coincide with the definition of time-stamped equivalence, i.e. both are equally discriminating. This is formalised in Theorem 3.3.

Theorem 3.3 (*Corresponding Interpretations*).
Let X_1 and X_2 be hybrid automata. The relation between both semantics is expressed by the following equation:

$$X_1 \equiv X_2 \ \Leftrightarrow \ X_1 \equiv_w^{\iota} \ X_2$$

Proof. Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \iota_1, \theta_1, \mathcal{I}_1, E_1 \rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \iota_2, \theta_2, \mathcal{I}_2, E_2 \rangle$ be two hybrid automata. If $R : [X_1^s] \underline{\leftrightarrow}_t^{\iota} [X_2^s]$, then it is not hard to check that also $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$.
Assume $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$; define R' as follows:

$$
\begin{aligned}
R' \quad = \quad & R \cup \{\langle (m, \vartheta', t'), (n, \chi', r') \rangle |\ m \in M_1 \wedge n \in M_2 \wedge \exists_{\vartheta, \chi \in \mathbb{V}, t, r \in \mathbb{T}} [\ (m, \vartheta, t)\ R\ (n, \chi, r) \wedge \\
& \exists_{f \in \mathcal{D}_{t'-t}^{\mathbb{V}}}\ ((f \cup \dot{f})] \omega_f \models \theta_1(m) \wedge f \models \iota_1(m) \wedge f(\mathbf{0}) = \vartheta \wedge f(t'-t) = \vartheta') \wedge \\
& \exists_{g \in \mathcal{D}_{r'-r}^{\mathbb{V}}}\ ((g \cup \dot{g})] \omega_g \models \theta_2(n) \wedge g \models \iota_2(n) \wedge g(\mathbf{0}) = \chi \wedge g(r'-r) = \chi')]\}
\end{aligned}
$$

It is not hard to check that R' is a bisimulation relation, such that $R' : [X_1^s] \underline{\leftrightarrow}_t^{\iota} [X_2^s]$ holds. $\square$

In the remainder of this paper we will only consider the time-stamped semantics of hybrid and timed automata. The extension of hybrid automata to synchronising hybrid automata will therefore only be given based on the time-stamped semantics for hybrid automata. A two-phase semantics for the synchronisation operator can be found in e.g. [13].

Typical examples of hybrid systems can be found in industry. Many of these systems can be characterised as digital controllers that are used to control analog processes. This means that in the area of hybrid systems, there is a desire to express parallelism in some form. As for timed automata, an auxiliary operator is defined that can be used to model blocking synchronisation, denoted by the operator $\|_s$. Definition 3.20 provides the rules describing the elimination of this operator on the level of hybrid automata.

It must be noted that there are other means of modelling parallelism and communication. Noteworthy is the approach in [22]. There, synchronisation can be both on shared actions and on shared continuous variables.

Definition 3.20 (*Synchronising Hybrid Automata*).

Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \mathcal{V}_1, \iota_1, \theta_1, \mathcal{I}_1, E_1 \rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \mathcal{V}_2, \iota_2, \theta_2, \mathcal{I}_2, E_2 \rangle$ be hybrid automata. Assume the sets of continuous variables are disjoint, i.e. $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$. The parallel composition of X_1 and X_2, denoted by $X_1 \|_s X_2$ is the hybrid automaton $\langle M_1 \times M_2, M_1^0 \times M_2^0, \Sigma_1 \cup \Sigma_2, \mathcal{V}_1 \cup \mathcal{V}_2, \iota, \theta, \mathcal{I}, E \rangle$, where $\iota(m,n) = \iota_1(m) \wedge \iota_2(n)$, $\theta(m,n) = \theta_1(m) \wedge \theta_2(n)$ and $\mathcal{I}(m,n) = \mathcal{I}_1(m) \wedge \mathcal{I}_2(n)$. The set of transitions E is defined as the least relation satisfying the rules of Table 10.

$$\frac{m \xrightarrow{\sigma,\varphi,\rho} m' \quad n \xrightarrow{\sigma,\psi,\pi} n'}{(m,n) \xrightarrow{\sigma,\varphi\cup\psi,\rho\wedge\pi} (m',n')} \qquad \frac{m \xrightarrow{\sigma,\varphi,\rho} m' \quad \sigma \in \Sigma_1 \setminus \Sigma_2}{(m,n) \xrightarrow{\sigma,\varphi,\rho} (m',n)} \qquad \frac{n \xrightarrow{\sigma,\psi,\pi} n' \quad \sigma \in \Sigma_2 \setminus \Sigma_1}{(m,n) \xrightarrow{\sigma,\psi,\pi} (m,n')}$$

Table 10: *Rules for the Synchronisation Operator*

Definition 3.21 defines a time-stamped semantics for the synchronisation operator. The rules for this operator are similar to the rules for the synchronisation operator for timed automata (cf. Def. 3.9).

Definition 3.21 (*Semantics of Synchronisation*).
Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \mathcal{V}_1, \theta_1, \iota_1, \mathcal{I}_1, E_1 \rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \mathcal{V}_2, \theta_2, \iota_2, \mathcal{I}_2, E_2 \rangle$ be two hybrid automata. The semantics of $X_1 \|_s X_2$ is given by the time-stamped transition system $X_1^{\overline{s}} \|_s X_2^{\overline{s}} = \langle (M_1 \times \mathbb{V} \times \mathbb{T}) \times (M_2 \times \mathbb{V} \times \mathbb{T}), I, \Sigma_1 \cup \Sigma_2, \mathbb{T}, \longrightarrow, \mathcal{U} \rangle$. Here, I is defined as the set $I = \{((m,n), \vartheta, \mathbf{0}) \in (M_1 \times M_2) \times \mathbb{V} \times \mathbb{T} \mid \vartheta \models (\mathcal{I}_1(m) \wedge \mathcal{I}_2(m))\}$. For all states $(m, \vartheta, t) \in X_1^{\overline{s}}$ and $(n, \chi, u) \in X_2^{\overline{s}}$, the set of transitions $\longrightarrow$ and the predicate $\mathcal{U}$ are defined as the least relations satisfying the rules in Table 11.

$$\frac{\mathcal{U}_r(m,\vartheta,t) \quad \mathcal{U}_r(n,\chi,u)}{\mathcal{U}_r((m,\vartheta,t)\|_s(n,\chi,u))}$$

$$\frac{(m,\vartheta,t) \xrightarrow{\sigma}_r (m',\vartheta',r) \quad \mathcal{U}_r(n,\chi,u) \quad \sigma \in \Sigma_1 \setminus \Sigma_2 \quad u \uparrow t \le r}{(m,\vartheta,t)\|_s(n,\chi,u) \xrightarrow{\sigma}_r (m',\vartheta',r)\|_s(n,\chi,u)}$$

$$\frac{(n,\chi,u) \xrightarrow{\sigma}_r (n',\chi',r) \quad \mathcal{U}_r(m,\vartheta,t) \quad \sigma \in \Sigma_2 \setminus \Sigma_1 \quad u \uparrow t \le r}{(m,\vartheta,t)\|_s(n,\chi,u) \xrightarrow{\sigma}_r (m,\vartheta,t)\|_s(n',\chi',r)}$$

$$\frac{(m,\vartheta,t) \xrightarrow{\sigma}_r (m',\vartheta',r) \quad (n,\chi,u) \xrightarrow{\sigma}_r (n',\chi',r)}{(m,\vartheta,t)\|_s(n,\chi,u) \xrightarrow{\sigma}_r (m',\vartheta',r)\|_s(n',\chi',r)}$$

Table 11: *Rules for the Synchronisation Operator*

The rules for the synchronisation operator are sound and complete with respect to the rules defined by Def. 3.20. This means that it does not matter whether parallelism is removed on the level of hybrid automata or on the level of the time-stamped transition systems. This is expressed in theorem 3.4.

Theorem 3.4 (*Soundness and Completeness of Synchronisation*).
Let X_1 and X_2 be two hybrid automata. Let $X_1^{\overline{s}}$ be the interpretation of X_1 and let $X_2^{\overline{s}}$ be the interpretation of X_2 and let $(X_1\|_s X_2)^{\overline{s}}$ be the interpretation of $X_1\|_s X_2$. Then, there exists a bisimulation relation R, such that $R : (X_1\|_s X_2)^{\overline{s}} \,\underline{\leftrightarrow}_t\, X_1^{\overline{s}} \|_s X_2^{\overline{s}}$.

Proof. Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \iota_1, \theta_1, \mathcal{I}_1, E_1 \rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \iota_2, \theta_2, \mathcal{I}_2, E_2 \rangle$ be hybrid automata. Then, for each $m \in M_1$, $n \in M_2$, $\vartheta, \chi \in \mathbb{V}$, $t, u \in \mathbb{T}$, the relation R is defined as follows:

$$\begin{array}{rcl} R & = & \{ \langle (m,\vartheta,t) \|_s (n,\chi,t) \ , \ ((m,n), \vartheta \cup \chi, t) \rangle \ \} \\ & \cup & \{ \langle (m,\vartheta,t) \|_s (n,\chi,u) \ , \ ((m,n), \vartheta \cup \chi', t) \rangle \ | u \leq t \wedge \\ & & \qquad \exists_{f \in \mathcal{D}^{\mathbb{V}}_{t-u}} (f \rceil \omega_f \models \theta_2(n) \wedge f \models \iota_2(n) \wedge f(\mathbf{0}) = \chi \wedge f(t-u) = \chi') \} \\ & \cup & \{ \langle (m,\vartheta,t) \|_s (n,\chi,u) \ , \ ((m,n), (\vartheta + u - t) \cup \chi, t) \rangle \ | t \leq u \wedge \\ & & \qquad \exists_{f \in \mathcal{D}^{\mathbb{V}}_{u-t}} (f \rceil \omega_f \models \theta_1(m) \wedge f \models \iota_1(m) \wedge f(\mathbf{0}) = \vartheta \wedge f(u-t) = \vartheta') \} \end{array}$$

Then, it is not hard to prove that R is indeed a bisimulation relation, such that
$R : (X_1 \parallel_s X_2)^{\overline{s}} \ \underline{\leftrightarrow}_t \ X_1^{\overline{s}} \parallel_s X_2^{\overline{s}}$. □

4 Process Algebra

The objects studied in process algebras are *processes*. Processes are abstract representations of real-life systems. In essence, processes describe all possible observable behaviours of the real-life system. The most elementary processes are *actions*. Actions are abstract representations of observable events that can occur in the real-life system. Combining processes allows for representing more complicated behaviour of systems.

The addition of time severely complicates the notion of a process. Processes are no longer considered to be entities with an unlimited life-span. Instead, the life-span of a process is restricted by hard timing constraints. In some sense, this allows for more accurate representations of real-life systems.

Process algebras come in various variants, such as CCS [23], CSP [18], LOTOS [7, 1] or ACP [5]. The process algebra employed in this paper is μCRL$_t$ [11], and a minor extension thereof, called μCRL$_t^\bullet$. In this paper, we will restrict ourselves to presenting the syntax of μCRL$_t$ and μCRL$_t^\bullet$, and its operational semantics. The axioms are included in appendix A.

4.1 Data Types

Unlike many other process algebras, the formalism μCRL$_t^\bullet$ combines a notion of data with descriptions of processes.

In the subsequent sections we will assume the time domain $\mathbb{T}$ is one of the data types of μCRL$_t^\bullet$. Only a few conceptual requirements are required for the time domain. The time domain should be totally ordered and contain a least element $\mathbf{0}$. No further requirements are posed on this data type. Moreover, we assume the existence of a set of Booleans ($\mathbb{B}$) with only two elements (t and f). Finally, the data type $\mathbb{R}$ represents the set of real numbers and the data type $\mathbb{F}$ represents the set of functions, including the set of constants (e.g. the set $\mathbb{R}$). For the subset of functions that are continuous and differentiable almost everywhere, we will write $\mathcal{D}_t^D$ for D of type $\mathbb{F}$ and t of type $\mathbb{T}$.

4.2 The Theory of pCRL$_t$ and pCRL$_t^\bullet$

The theory pCRL is built around a small number of intuitive basic operators and constants. It serves as the core of μCRL. The action constants model events that can be executed. The constant δ is used to model inaction, or *deadlock*. The most basic operators describe a notion of *alternative composition*, denoted by the infix operator $+$ and *sequential composition*, denoted by the infix operator $\cdot$. Alternative composition can be used to express a (non-deterministic) choice between two processes. The choice is resolved the moment one of the processes executes its first action. By means of sequential composition, one can model a sequence of processes that need to be executed. The special deadlock constant is the unit for alternative composition. Sequential composition is a generalisation of *action prefixing*, which is not present in our language. The sum operator ($\sum_{d:D}$) generalises alternative composition. Such a quantifier can be used to accurately model processes that depend on (possibly infinite) data sorts. In the context of data-dependent processes, a construct similar to the *if_then_else* selection is often required. This is denoted using the ternary operator $p \lhd b \rhd q$, meaning p if b is true, and q if b is false.

The novelty of $p\mathrm{CRL}_t$ over pCRL is the possibility to explicitly express timing dependencies of processes. This is modelled using the infix operator $^{\varsigma}$ (pronounced "at"). Intuitively, a process $p^{\varsigma}t$ models the process that must perform its first action at time t. Since the process p can be of arbitrary complexity, this means that the $^{\varsigma}$-operator propagates forward to the first action that occurs in p. The process $\delta^{\varsigma}t$ expresses a *timed deadlock*, whereas the special constant δ (which is equivalent to the process $\sum_t \delta^{\varsigma}t$) can now be understood to model *livelock*, i.e. the constant δ allows time to continue, but it can execute no actions, nor can it terminate. Note that in (untimed) pCRL the constant δ represents deadlock.

Standard $p\mathrm{CRL}_t$ does not contain a special constant modelling a deadlocked process, denoted by $\overset{\bullet}{\delta}$. However, this is easily mended, resulting in a slightly different version of $p\mathrm{CRL}_t$, viz. $p\mathrm{CRL}_t^{\bullet}$. The differences are minor, and even absent in the operational semantics. This is due to the fact that a deadlocked process can neither execute actions, nor idle (not even until "current" time).

Along with the $^{\varsigma}$-operator, an initialisation operator $\gg$ is included in the axiom-system and in the operational semantics. This operator can, however, be expressed using the operators already discussed, and hence can be considered to be an auxiliary operator. Formally, the syntax of $p\mathrm{CRL}_t^{\bullet}$ is discussed in Def. 4.1.

Definition 4.1 (*Syntax of* $p\mathrm{CRL}_t^{\bullet}$).
The signature of the theory $p\mathrm{CRL}_t^{\bullet}$ consists of a data signature and a process signature. The process signature consists of the sort symbol P and the following function declarations:

1. $a : s_1 \times \cdots \times s_n \to P$, action declarations for sort symbols s_i $(i = 0, \cdots, n)$ from the data signature,
2. $\delta :\to P$, modelling *livelock*,
3. $\overset{\bullet}{\delta} :\to P$, modelling the *deadlocked process*,
4. $+ : P \times P \to P$ is the alternative composition operator,
5. $\sum_v : P \to P$ is the alternative quantification,
6. $\cdot : P \times P \to P$ is the sequential composition operator,
7. $\lhd\ \rhd : P \times \mathbb{B} \times P \to P$ is the conditional operator,
8. $^{\varsigma} : P \times \mathbb{T} \to P$ is the at-operator,
9. $\gg : \mathbb{T} \times P \to P$ is the initialisation operator.

The binding strength of all operators is listed in decreasing strength:

$$^{\varsigma},\quad \cdot,\quad \gg,\quad \lhd\ \rhd,\quad \sum,\quad +$$

We write $a : s_1 \times \cdots \times s_n$ instead of $a : s_1 \times \cdots \times s_n \to P$. The set of all action declarations is denoted Act. Action terms are terms of the form $a(d_1, \ldots, d_n)$, where $a : s_1 \times \ldots \times s_n \in Act$, and d_i a data term of sort s_i (for $i = 0, \ldots, d_n$). The set of all action terms is denoted AT. We will write AT_δ instead of $AT \cup \{\delta\}$. Instead of writing $AT \cup \{\delta, \overset{\bullet}{\delta}\}$, we will write $AT_\delta^{\bullet}$.

Henceforth, we assume the existence of an infinite set of process variables V_P which is disjoint from the set of data variables V.

The objects in process algebra are *process terms* which are built from action terms, data terms, variables and process operators. The set of all process terms is denoted T_P. Process terms not containing process variables are called *process-closed terms.* Data variables, either bound or free can still occur in process-closed terms.

One of the main characteristics of a process algebra is the characterisation of operators by means of an axiom system. This provides a syntactical view on processes and process theory. The interested reader is referred to appendix A for an overview of the axioms. For a more elaborate overview of $p\mathrm{CRL}_t$ see [11, 10].

In this paper, the focus will be on the operational aspects of the language $p\mathrm{CRL}_t^{\bullet}$ and $\mu\mathrm{CRL}_t^{\bullet}$. The characteristics of the operators is described by means of an operational semantics in the style of Plotkin [24]. However, a semantics based on process terms is not straightforwardly obtained. Process terms in general are terms that can contain bound and free data variables. One way of dealing with this is to first define an interpretation for these process terms, thus obtaining true *processes.* The set of processes and actions is defined in Def. 4.2. The operational semantics can then be described using processes instead of process terms.

The interpretation of process terms hinges on the assumption of the existence of a data algebra with a universe $\mathcal{D}_s$ for each sort symbol s. Then, a valuation ν is a function from the set of variables V to the set of values $\cup_{s\in S}\mathcal{D}_s$, such that $\nu(v) \in \mathcal{D}_s$ if and only if $v \in V_s$. This latter requirement ensures that variables are mapped only to values of the right data algebra. As an abbreviation, we will write $\nu[v]\nu'$, where ν and ν' are valuations and $v \in V$ is a variable whenever for all $u \in V \setminus \{v\}$ we have $\nu(v) = \nu'(v)$. We will write $[\![t]\!]^{\nu}$ for the interpretation of a term t under valuation ν.

Definition 4.2 (*Actions and Processes*).
Given a $p\mathrm{CRL}_t^{\bullet}$ specification with a set of action declarations *Act*, the set of *actions* is defined by

$$\mathcal{A} = \{a(d_1,\ldots,d_n) | a : s_1 \times \cdots \times s_n \in Act, d_i \in \mathcal{D}_{s_i}\}$$

The set $\mathcal{A} \cup \{\delta\}$ is denoted by $\mathcal{A}_\delta$ and the set $\mathcal{A} \cup \{\delta, \overset{\bullet}{\delta}\}$ is denoted by $\mathcal{A}_{\delta}^{\bullet}$. The set $\mathcal{P} = \mathcal{P}^{\omega}$ is obtained by the following recursion:

$$\begin{array}{lcl} \mathcal{P}^0 & = & \mathcal{A}_{\delta}^{\bullet} \\ \mathcal{P}^{n+1} & = & \mathcal{P}^n \cup \{p\cdot q, \sum P', p\text{`}t, t \gg p | p,q \in \mathcal{P}^n, P' \neq \emptyset, P' \subseteq \mathcal{P}^n, t \in \mathcal{D}_{\mathbb{T}}\} \end{array}$$

Definition 4.3 (*Interpretation of Process-Closed Terms*).
Let $a : s_1 \times \cdots \times s_n \in Act$ and $d_i \in T_{s_i}$. Let $b : T_{\mathbb{B}}, t : T_{\mathbb{T}}$ and let $p, q \in T_P$ be process closed terms. The interpretation of process-closed terms under valuation ν is defined inductively by:

$$\begin{array}{lcl} [\![a(d_1,\cdots,d_n)]\!]^{\nu} & = & a([\![d_1]\!]^{\nu},\cdots,[\![d_n]\!]^{\nu}), \\ [\![\delta]\!]^{\nu} & = & \delta, \\ [\![\,\overset{\bullet}{\delta}\,]\!]^{\nu} & = & \overset{\bullet}{\delta}, \\ [\![p+q]\!]^{\nu} & = & \sum([\![p]\!]^{\nu},[\![q]\!]^{\nu}) \\ [\![p\cdot q]\!]^{\nu} & = & [\![p]\!]^{\nu}\cdot[\![q]\!]^{\nu}, \\ [\![p\text{`}t]\!]^{\nu} & = & [\![p]\!]^{\nu}\text{`}[\![t]\!]^{\nu} \\ [\![t \gg p]\!]^{\nu} & = & [\![t]\!]^{\nu} \gg [\![p]\!]^{\nu} \\ [\![p \lhd b \rhd q]\!]^{\nu} & = & \left\{ \begin{array}{ll} [\![p]\!]^{\nu} & \text{if } [\![b]\!]^{\nu} = [\![\mathrm{t}]\!]^{\nu} \\ [\![q]\!]^{\nu} & \text{if } [\![b]\!]^{\nu} = [\![\mathrm{f}]\!]^{\nu} \end{array} \right. \\ [\![\sum_v p]\!]^{\nu} & = & \sum\{[\![p]\!]^{\nu'} | \nu[v]\nu'\} \end{array}$$

The operational semantics of processes is defined in terms of the timed execution of actions. This is closely related to the framework of timed transition systems (see section 2). Here, we introduce *action relations* $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}}$ and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$. Instead of writing $\langle p, a, t\rangle \in \longrightarrow$ and $\langle p, a, t, p'\rangle \in \longrightarrow$, we write $p \xrightarrow{a}_t \surd$ and $p \xrightarrow{a}_t p'$. By $p \xrightarrow{a}_t \surd$, we express that process p can perform an action at time t and then successfully terminate at time t. By $p \xrightarrow{a}_t p'$ we express that process p evolves into process p' at time t by performing an action a. Furthermore, we are interested in the idling behaviour of processes, i.e. we need a notion of expressing until when a process can still idle. This is expressed by the binary delay relation $\mathcal{U} \subseteq \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$. By $\mathcal{U}_t(p)$, we express that process p can idle at least until time t. The operational semantics is formally defined in Def. 4.4.

Definition 4.4 (*Structured Operational Semantics of pCRL$_t^\bullet$*).
The action relations $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}}$ and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$ are defined by the transition rules in Table 12, whereas the delay relation $\mathcal{U} \subseteq \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$ is defined by the transition rules in Table 13.

$$a \xrightarrow{a}_t \surd \qquad \frac{p \xrightarrow{a}_t \surd}{\sum(\{p\} \cup P) \xrightarrow{a}_t \surd} \qquad \frac{p \xrightarrow{a}_t p'}{\sum(\{p\} \cup P) \xrightarrow{a}_t p'} \qquad \frac{p \xrightarrow{a}_t \surd}{p{}^{\backprime}t \xrightarrow{a}_t \surd} \qquad \frac{p \xrightarrow{a}_t p'}{p{}^{\backprime}t \xrightarrow{a}_t p'}$$

$$\frac{p \xrightarrow{a}_t \surd}{p \cdot q \xrightarrow{a}_t t \gg q} \qquad \frac{p \xrightarrow{a}_t p'}{p \cdot q \xrightarrow{a}_t p' \cdot q} \qquad \frac{p \xrightarrow{a}_t \surd \quad t' \leq t}{t' \gg p \xrightarrow{a}_t \surd} \qquad \frac{p \xrightarrow{a}_t p' \quad t' \leq t}{t' \gg p \xrightarrow{a}_t p'}$$

Table 12: *Transition rules for the pCRL$_t^\bullet$ operators, where $a \in \mathcal{A}, p, p', q \in \mathcal{P}, t, t' \in \mathcal{D}_{\mathbb{T}}$ and $P \subseteq \mathcal{P}$*

$$\mathcal{U}_t(a) \qquad \frac{\mathcal{U}_t(p)}{\mathcal{U}_t(p \cdot q)} \qquad \frac{\mathcal{U}_t(p)}{\mathcal{U}_t(\sum(\{p\} \cup P))} \qquad \frac{\mathcal{U}_t(p) \quad t \leq t'}{\mathcal{U}_t(p{}^{\backprime}t')} \qquad \frac{\mathcal{U}_t(p)}{\mathcal{U}_t(t' \gg p)} \qquad \frac{\mathcal{U}_0(p) \quad t \leq t'}{\mathcal{U}_t(t' \gg p)}$$

Table 13: *Transition rules for the delay relation, where $a \in \mathcal{A}_\delta, p, q \in \mathcal{P}, t, t' \in \mathcal{D}_{\mathbb{T}}$ and $P \subseteq \mathcal{P}$*

Remark 4.1. The execution of an action does not take time, i.e. it is very well possible to sequentially execute two subsequent actions at time t.

Remark 4.2. The deadlocked process does not appear in the axioms and the conclusion of the rules of Tables 12 and 13, since it can neither execute actions, nor idle. The livelock constant δ does not appear in the axioms and conclusions of the rules of Table 12 since it cannot execute actions, nor can it terminate. The livelock constant can, however,idle as is expressed in Table 13.

A suitable notion of equivalence relates both the executions of actions of processes and their idling behaviour. Process equivalence is often expressed by means of strong bisimulation. Here, we will adapt the notion of strong bisimulation to deal with the timing requirements. This notion of bisimulation is called *strong timed bisimulation*. A formal definition thereof can be found in Def. 4.5.

Definition 4.5 (*Strong Timed Bisimulation*).
A symmetric relation $R \subseteq \mathcal{P} \times \mathcal{P}$ is a strong timed bisimulation if we have for all $(p, q) \in R$ and $a \in \mathcal{A}$, $t \in \mathcal{D}_{\mathbb{T}}$ and $p' \in \mathcal{P}$:

1. if $p \xrightarrow{a}_t \surd$, then $q \xrightarrow{a}_t \surd$,
2. if $p \xrightarrow{a}_t p'$, then there is a q', such that $q \xrightarrow{a}_t q'$ and $(p', q') \in R$, and
3. if $\mathcal{U}_t(p)$, then $\mathcal{U}_t(q)$.

If a strong timed bisimulation relation R exists such that $(p, q) \in R$, we write $R : p \underline{\leftrightarrow}_t q$.

Strong timed bisimulation is used to justify the axioms. Results related to the soundness and completeness of $p\text{CRL}_t$ are discussed in [11]. With respect to the extension of $p\text{CRL}_t$ with the deadlocked process we have the following theorem:

Theorem 4.1 (*Soundness of pCRL$_t$*). The axiom system $p\text{CRL}_t^\bullet$ is sound with respect to strong timed bisimulation.

Proof. The theory $p\text{CRL}_t$ itself is sound (see [11]). Hence, for soundness we only need to consider the axioms describing the behaviour of the deadlocked process $\overset{\bullet}{\delta}$. It is easy to prove these axioms are sound. Hence, the axiom system $p\text{CRL}_t^\bullet$ is sound. □

4.3 The Theory of μCRL_t and $\mu\text{CRL}_t^\bullet$

The theories of $p\text{CRL}_t$ and $p\text{CRL}_t^\bullet$ are rather restrictive in the sense that they do not allow for specifying parallelism in a natural fashion. In this sense, the theories of $p\text{CRL}_t$ and $p\text{CRL}_t^\bullet$ are not interesting. However, concurrency is incorporated both in μCRL_t and $\mu\text{CRL}_t^\bullet$, based on an *interleaving semantics*.

To express two processes are executed in parallel, the binary operator $\|$ is used. The parallel execution of process $p\|q$ can choose to first perform an action of p, first perform an action of q, or choose to synchronise. Hence, parallelism can be described using two auxiliary operators, viz. a *left merge* operator ($\lfloor\!\lfloor$) and a *communication merge* operator ($|$). Intuitively, the process $p \lfloor\!\lfloor q$ behaves as $p\|q$, with the restriction that the first action is executed by p and p and q cannot synchronise on this action. Much like in $p\text{CRL}_t$ and $p\text{CRL}_t^\bullet$, an auxiliary operator $\ll$ is introduced. This operator is needed in giving an axiomatisation of the left merge operator. Intuitively, $p \ll q$ (pronounced p before q) denotes the process that behaves as process p until q can idle no more. In a sense, this operator represents a *time-out*. The process $p|q$ again behaves as the process $p\|q$, with the restriction that the processes p and q should synchronise (*communicate*) on their first action. Moreover, these actions should also synchronise on the time they are executed. The action resulting from a communication is defined by a binary, commutative and associative function γ, which is only defined on *action declarations*.

The theories of $p\text{CRL}_t$ and μCRL_t (or $p\text{CRL}_t^\bullet$ and $\mu\text{CRL}_t^\bullet$) are strongly related in the sense that parallel operators can be reformulated to a combination of alternative and sequential composition operators. This is expressed by the axiom system.

Whenever we wish to block actions of a process, we can *encapsulate* these actions, using the ∂_H operator. When we write $\partial_H(p)$, only those actions of p that do not occur in H can be executed. This operator is essential when we wish to force two processes to synchronise on certain actions.

In some contexts, *renaming* of actions can be useful. Using the renaming operator ρ_R, actions can be assigned new names according to the function R. This operator is absent in μCRL_t, however, it is present in μCRL and $\mu\text{CRL}_t^\bullet$. A formal discussion of the syntax is given in Def. 4.6.

Definition 4.6 (*Syntax of $\mu CRL_t^\bullet$*).

The signature of the theory $\mu\text{CRL}_t^\bullet$ consists of a data signature and a process signature. The process signature consists of the sort symbol P, the operator declarations of $p\text{CRL}_t^\bullet$ and the following additional function declarations:

1. $\| : P \times P \to P$, the parallel composition operator,
2. $\lfloor\!\lfloor : P \times P \to P$, the left-merge operator,
3. $| : P \times P \to P$, the communication merge operator,
4. $\ll : P \times P \to P$, the before, or *time-out* operator,
5. $\partial_H : \mathbb{P}(Act) \times P \to P$, the encapsulation operator,
6. $\rho_R : \mathbb{P}(Act \times Act) \times P \to P$, the renaming operator,
7. $\gamma : Act \times Act \to Act$ is the communication function.

The binding strength of all operators is listed in decreasing strength:

$$ˁ, \quad \partial_H, \quad \cdot, \quad \{\gg, \quad \ll\}, \quad \{\lhd\,\rhd, \quad \|, \quad ,\lfloor\!\lfloor, \quad |\}, \quad \sum, \quad +$$

The axioms characterising the operators of $\mu\text{CRL}_t^{\bullet}$ are listed in appendix A, and are provided as is in this paper. For a detailed account on the axioms of μCRL_t, please refer to [10, 11]. The addition of the deadlocked process to μCRL_t, yielding $\mu\text{CRL}_t^{\bullet}$ is straightforward. The focus of this section will be on the structured operational semantics of $\mu\text{CRL}_t^{\bullet}$. Again, the operational semantics is given in terms of processes rather than process terms. The set of processes defined in Def. 4.2 is extended with to the set defined in Def. 4.7.

Definition 4.7 (*Actions and Processes*).
Given a $\mu\text{CRL}_t^{\bullet}$ specification with a set of action declarations Act, the set of actions is defined by

$$\mathcal{A} = \{a(d_1, \cdots, d_n) | a : s_1 \times \cdots \times s_n \in Act, d_i \in \mathcal{D}_{s_i}\}$$

The set $\mathcal{A} \cup \{\delta\}$ is denoted by $\mathcal{A}_\delta$ and the set $\mathcal{A} \cup \{\delta, \overset{\bullet}{\delta}\}$ is denoted by $\mathcal{A}_\delta^{\bullet}$. The set $\mathcal{P} = \mathcal{P}^\omega$ is obtained by the following recursion:

$$\begin{array}{lcl} \mathcal{P}^0 & = & \mathcal{A}_\delta^{\bullet} \\ \mathcal{P}^{n+1} & = & \mathcal{P}^n \cup \{p \cdot q, \sum P', pˁt, t \gg p, p\|q, p\lfloor\!\lfloor q, p|q, p \ll q, \partial_H(p), \rho_R(p) \\ & & |p, q \in \mathcal{P}^n, P' \neq \emptyset, P' \subseteq \mathcal{P}^n, t \in \mathcal{D}_{\mathbb{T}}, H \subseteq Act, R \subseteq Act \times Act\} \end{array}$$

Definition 4.8 (*Interpretation of Process-Closed Terms*).
For process-closed terms $p, q \in T_P, H \subseteq Act, R \subseteq \mathbb{P}(Act \times Act)$ and valuation ν, the interpretation of process-closed terms under a valuation, given in Def. 4.3 is extended with the following clauses:

$$\begin{array}{lcl} [\![p \ll q]\!]^\nu & = & [\![p]\!]^\nu \ll [\![q]\!]^\nu, \\ [\![p\|q]\!]^\nu & = & [\![p]\!]^\nu \| [\![q]\!]^\nu, \\ [\![p\lfloor\!\lfloor q]\!]^\nu & = & [\![p]\!]^\nu \lfloor\!\lfloor [\![q]\!]^\nu, \\ [\![p|q]\!]^\nu & = & [\![p]\!]^\nu | [\![q]\!]^\nu, \\ [\![\partial_H p]\!]^\nu & = & \partial_H([\![p]\!]^\nu) \\ [\![\rho_R p]\!]^\nu & = & \rho_R([\![p]\!]^\nu) \end{array}$$

The operational semantics of processes is again defined in terms of the timed execution of actions. The set of action relations $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}}$ and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$ are extended to cope with the new operators of $\mu\text{CRL}_t^{\bullet}$. Formally, the operational semantics is defined in Def. 4.9.

Definition 4.9 (*Structured Operational Semantics of $\mu CRL_t^{\bullet}$*).
The action relations $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}}$ and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$ are defined by the transition rules in Tables 14 and 15, whereas the delay relation $\mathcal{U} \subseteq \mathcal{D}_{\mathbb{T}} \times \mathcal{P}$ is defined by the transition rules in Table 16.

$$\frac{p \xrightarrow{a}_t \surd \quad \mathcal{U}_t(q)}{p\|q \xrightarrow{a}_t t \gg q} \qquad \frac{q \xrightarrow{a}_t \surd \quad \mathcal{U}_t(p)}{p\|q \xrightarrow{a}_t t \gg p} \qquad \frac{p \xrightarrow{a}_t \surd \quad \mathcal{U}_t(q)}{p \mathbin{⫝̸} q \xrightarrow{a}_t t \gg q} \qquad \frac{p \xrightarrow{a}_t \surd \quad \mathcal{U}_t(q)}{p \ll q \xrightarrow{a}_t \surd}$$

$$\frac{p \xrightarrow{a}_t p' \quad \mathcal{U}_t(q)}{p\|q \xrightarrow{a}_t p'\|t \gg q} \qquad \frac{q \xrightarrow{a}_t q' \quad \mathcal{U}_t(p)}{p\|q \xrightarrow{a}_t t \gg p\|q'} \qquad \frac{p \xrightarrow{a}_t p' \quad \mathcal{U}_t(q)}{p \mathbin{⫝̸} q \xrightarrow{a}_t p'\|t \gg q} \qquad \frac{p \xrightarrow{a}_t p' \quad \mathcal{U}_t(q)}{p \ll q \xrightarrow{a}_t p'}$$

$$\frac{p \xrightarrow{\mathsf{a}(d)}_t \surd \quad q \xrightarrow{\mathsf{b}(d)}_t \surd \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p\|q \xrightarrow{\mathsf{c}(d)}_t \surd} \qquad \frac{p \xrightarrow{\mathsf{a}(d)}_t \surd \quad q \xrightarrow{\mathsf{b}(d)}_t q' \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p\|q \xrightarrow{\mathsf{c}(d)}_t q'}$$

$$\frac{p \xrightarrow{\mathsf{a}(d)}_t p' \quad q \xrightarrow{\mathsf{b}(d)}_t \surd \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p\|q \xrightarrow{\mathsf{c}(d)}_t p'} \qquad \frac{p \xrightarrow{\mathsf{a}(d)}_t p' \quad q \xrightarrow{\mathsf{b}(d)}_t q' \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p\|q \xrightarrow{\mathsf{c}(d)}_t p'\|q'}$$

$$\frac{p \xrightarrow{\mathsf{a}(d)}_t \surd \quad q \xrightarrow{\mathsf{b}(d)}_t \surd \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p|q \xrightarrow{\mathsf{c}(d)}_t \surd} \qquad \frac{p \xrightarrow{\mathsf{a}(d)}_t \surd \quad q \xrightarrow{\mathsf{b}(d)}_t q' \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p|q \xrightarrow{\mathsf{c}(d)}_t q'}$$

$$\frac{p \xrightarrow{\mathsf{a}(d)}_t p' \quad q \xrightarrow{\mathsf{b}(d)}_t \surd \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p|q \xrightarrow{\mathsf{c}(d)}_t p'} \qquad \frac{p \xrightarrow{\mathsf{a}(d)}_t p' \quad q \xrightarrow{\mathsf{b}(d)}_t q' \quad \gamma(\mathsf{a},\mathsf{b}) = \mathsf{c}}{p|q \xrightarrow{\mathsf{c}(d)}_t p'\|q'}$$

Table 14: *Transition rules for the parallel operators, where* $a \in \mathcal{A}, \mathsf{a},\mathsf{b},\mathsf{c} \in Act, d \in \mathcal{D}_s, p, q, p', q' \in \mathcal{P}$ *and* $t \in \mathcal{D}_{\mathbb{T}}$

$$\frac{p \xrightarrow{a(d)}_t \surd \quad a \notin H}{\partial_H(p) \xrightarrow{a(d)}_t \surd} \qquad \frac{p \xrightarrow{a(d)}_t p' \quad a \notin H}{\partial_H(p) \xrightarrow{a(d)}_t \partial_H(p')} \qquad \frac{p \xrightarrow{a(d)}_t \surd \quad a \in \mathrm{dom}(R)}{\rho_R(p) \xrightarrow{R(a)(d)}_t \surd}$$

$$\frac{p \xrightarrow{a(d)}_t p' \quad a \in \mathrm{dom}(R)}{\rho_R(p) \xrightarrow{R(a)(d)}_t \rho_R(p')} \qquad \frac{p \xrightarrow{a(d)}_t \surd \quad a \notin \mathrm{dom}(R)}{\rho_R(p) \xrightarrow{a(d)}_t \surd} \qquad \frac{p \xrightarrow{a(d)}_t p' \quad a \notin \mathrm{dom}(R)}{\rho_R(p) \xrightarrow{a(d)}_t \rho_R(p')}$$

Table 15: *Transition rules for encapsulation and renaming, where* $p, q \in \mathcal{P}, t \in \mathcal{D}_{\mathbb{T}}, H \subseteq Act, R \subseteq Act \times Act, \mathsf{a} \in Act$ *and* $d \in \mathcal{D}_s$

$$\frac{\mathcal{U}_t(p) \quad \mathcal{U}_t(q)}{\mathcal{U}_t(p \ll q)} \qquad \frac{\mathcal{U}_t(p) \quad \mathcal{U}_t(q)}{\mathcal{U}_t(p\|q)} \qquad \frac{\mathcal{U}_t(p) \quad \mathcal{U}_t(q)}{\mathcal{U}_t(p \mathbin{⫝̸} q)} \qquad \frac{\mathcal{U}_t(p) \quad \mathcal{U}_t(q)}{\mathcal{U}_t(p|q)} \qquad \frac{\mathcal{U}_t(p)}{\mathcal{U}_t(\partial_H(p))} \qquad \frac{\mathcal{U}_t(p)}{\mathcal{U}_t(\rho_R(p))}$$

Table 16: *Transition rules for the delay relation, where* $p, q \in \mathcal{P}, t \in \mathcal{D}_{\mathbb{T}}, H \subseteq Act$ *and* $R \subseteq Act \times Act$

The extension of μCRL$_t$ with the deadlocked process yields a sound system. This is expressed by the following Theorem:

Theorem 4.2 (*Soundness of $\mu CRL_t^\bullet$*). The axiom system μCRL$_t^\bullet$ is sound with respect to strong timed bisimulation.

Proof. The theory μCRL$_t$ is sound (see [11]). Hence, we only need to consider the axioms describing the behaviour of the deadlocked process $\overset{\bullet}{\delta}$. It is easy to prove these axioms are also sound. Hence, the axiom system μCRL$_t^\bullet$ is sound. □

In the subsequent sections, we will need several abbreviations and Lemma 4.1 expressing that the initialisation operator distributes over the parallel operator.

Definition 4.10 (*Abbreviations*).
Let x be a process term, $t \in \mathbb{T}$ and $\mathsf{b} \in \mathbb{B}$. Then the operators $::\!\rightarrow$ and $\gg_\bullet$ are defined as follows:

$$\begin{array}{lcl} \mathsf{b} \ ::\!\rightarrow x & = & x \triangleleft \mathsf{b} \triangleright \overset{\bullet}{\delta} \\ t \gg_\bullet x & = & \sum_{u:\mathbb{T}} \ [t \leq u] \ ::\!\rightarrow x^{\backprime} u \end{array}$$

Lemma 4.1 (*Distribution of initialisation*).
Let p and q be process-closed terms and $t \in \mathbb{T}$, then we have:

$$t \gg_\bullet (p \| q) \leftrightarrows_t (t \gg_\bullet p) \| (t \gg_\bullet q)$$

Lemma 4.2 (*Related initialisations*).
Let p be a process-closed term and $t, t' \in \mathbb{T}$, then we have:

$$\mathcal{U}_t(p) \wedge t' \leq t \Rightarrow t' \gg_\bullet p \leftrightarrows_t t' \gg p$$

5 Interpretations of Automata

The focus of this section is on the translation of timed automata and hybrid automata to process algebra. Essentially, this way one can analyse timed automata and hybrid automata based on their structure. Moreover, the general construction of such automata is emphasised, or highlighted. The hidden aspect of *deadlocked* parts of systems become clearly visible in the process algebraic counterpart of these automata. The gain is a better understanding of timed automata and hybrid automata.

5.1 Interpretations of timed automata

A timed automaton is in essence a high-level representation of a time-stamped transition system (or equally, a two-phase transition system). In that sense, the process algebraic interpretation of a timed automaton should yield an equivalent time-stamped transition system. The process algebraic interpretation of a timed automaton is based on the elementary building blocks of the timed automaton.

Definition 5.1 (*Abbreviations*).
Let $X = \langle L, L^0, \Sigma, \mathcal{C}, \iota, E \rangle$ be a timed automaton. Then, for all $l \in L$, the set E_l is defined as the set of edges starting in l:

$$E_l := \{ l \xrightarrow{\sigma, \varphi, \lambda} l' | l' \in L \wedge \sigma \in \Sigma \wedge \varphi \in \Phi(\mathcal{C}) \wedge \lambda \subseteq \mathcal{C} \}$$

For all edges $e \in E_l$, the following abbreviations are defined

$$
\begin{array}{lcl}
\sigma_e & \stackrel{\text{abbr.}}{=} & \pi_2(e) \\
\varphi_e & \stackrel{\text{abbr.}}{=} & \pi_3(e) \\
\lambda_e & \stackrel{\text{abbr.}}{=} & \pi_4(e) \\
l_e & \stackrel{\text{abbr.}}{=} & \pi_5(e)
\end{array}
$$

A formal definition of the process algebraic interpretation of a timed automaton is given in Def. 5.2.

Definition 5.2 (*Process Algebraic Interpretation of a Timed Automaton*).
Let $X = \langle L, L^0, \Sigma, \mathcal{C}, \iota, E\rangle$ be a timed automaton. Let $\vartheta_{\mathbf{0}}(c) = \mathbf{0}$ for all $c \in \mathcal{C}$. Let $[\![X]\!] : \mathcal{P}$ be the process algebraic interpretation of X, defined as:

$$
\begin{array}{lcl}
[\![X]\!] & = & \sum_{l:L^0}\ X_l(\mathbf{0}, \vartheta_{\mathbf{0}}) \\
 & & \\
X_l & = & \lambda t:\mathbb{T}.\lambda\vartheta:\mathbb{V}.\ t \gg_{\bullet} \sum_{u:\mathbb{T}}\ [\vartheta, (\vartheta + u - t) \models \iota(l)]\ :\!:\!\to \\
 & & \sum_{e:E_l, \vartheta':\mathbb{V}}\ (\sigma_e \cdot X_{l_e}(u, \vartheta') \triangleleft (\vartheta + u - t) \models \varphi_e\ \wedge\ \vartheta' = (\vartheta + u - t)[\lambda_e := 0] \triangleright \delta\ )\text{ʿ}u
\end{array}
$$

The translation of a timed automaton to a process algebraic expression is a task which can easily be automated. Note that a translation of the standard semantics of a timed automaton only requires an additional constraint $\vartheta' \models \iota(l_e)$ in the second condition of process X_l.

Example 5.1 provides an intuition on the appearance of a process algebraic interpretation of a timed automaton.

Example 5.1. Let X be the timed automaton illustrated in Fig. 6.

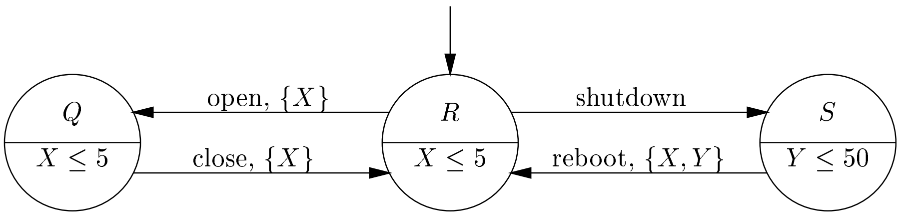

Figure 6: *Limited-lifespan System*

The process algebraic interpretation of X is the process $[\![X]\!]$, which, after some elementary calculus, is defined as the following process term:

$$
\begin{array}{lcl}
[\![X]\!] & = & X_R(\mathbf{0}, \mathbf{0}) \\
X_R(t, st:\mathbb{T}) & = & \sum_{u:\mathbb{T}} \text{open}\text{ʿ}u \cdot X_Q(u, st + u - t) \triangleleft u \leq t + 5 \triangleright \delta\text{ʿ}u \\
 & + & \sum_{u:\mathbb{T}} \text{shutdown}\text{ʿ}u \cdot X_S(u, st + u - t) \triangleleft u \leq t + 5 \triangleright \delta\text{ʿ}u \\
X_Q(t, st:\mathbb{T}) & = & \sum_{u:\mathbb{T}} \text{close}\text{ʿ}u \triangleleft u \leq t + 5 \triangleright \delta\text{ʿ}u \\
X_S(t, st:\mathbb{T}) & = & \sum_{u:\mathbb{T}} \text{reboot}\text{ʿ}u X_R(\mathbf{0}, \mathbf{0}) \triangleleft st \leq 50 \triangleright \stackrel{\bullet}{\delta}
\end{array}
$$

The process algebraic interpretation is defined in such a way that every location of a timed automaton has a dedicated process term, modelling its behaviour. Hence, one would expect to see a close relation between the behaviours a location can exhibit and the behaviours its dedicated process term exhibits. This relation is formalised by Lemmas 5.1 and 5.2.

Lemma 5.1 (*Coupled Behaviour*).
Let $X = \langle L, L^0, \Sigma, \iota, E\rangle$ be an arbitrary timed automaton. Then, for all $l \in L$, $\vartheta \in \mathbb{V}$, $t \in \mathbb{T}$:
$(l, \vartheta, t) \xrightarrow{\sigma}_{t'} (l', \vartheta', t')$ iff $X_l(t, \vartheta) \xrightarrow{\sigma}_{t'} X_{l'}(t', \vartheta')$

Proof. Assume $(l, \vartheta, t) \xrightarrow{\sigma}_{t'} (l', \vartheta', t')$. Hence, there must exist an edge $e' = l \xrightarrow{\sigma,\varphi,\lambda} l'$, such that $\vartheta, \vartheta + t' - t \models \iota(l)$ and $\vartheta + t' - t \models \varphi$ and $\vartheta' = (\vartheta + t' - t)[\lambda := 0]$ and $t \leq t'$ are true. This means there exists a valuation ν, such that $\nu(e) = e'$ and $\nu(u) = t'$. Hence, $X_l(t, \vartheta) \xrightarrow{\sigma}_{t'} X_{l'}(t', \vartheta')$.

Assume $X_l(t, \vartheta) \xrightarrow{\sigma}_{t'} X_{l'}(t', \vartheta')$. Hence, there must exist a valuation ν, such that $\nu(e) = l \xrightarrow{\sigma,\varphi,\lambda} l'$ and $\nu(u) = t'$ and $\vartheta, \vartheta + \nu(u) - t \models \iota(l)$ and $\vartheta + \nu(u) - t \models \varphi$ and $\nu(\vartheta') = (\vartheta + \nu(u) - t)[\lambda := 0]$ and $t \leq \nu(u)$. But then also $(l, \vartheta, t) \xrightarrow{\sigma}_{t'} (l', \vartheta', t')$. $\square$

The behaviour of a location is not entirely determined by the actions it prescribes. For instance, a location without outgoing edges, still portrays behaviour in the sense that it can let time pass. Hence, the passage of time, or the *idleness* of a location should be reflected by the dedicated process term of the location. This is expressed in Lemma 5.2.

Lemma 5.2 (*Coupled Idling Time*).
Let $X = \langle L, L^0, \Sigma, \iota, E\rangle$ be an arbitrary timed automaton. Then, for all $l \in L$, $\vartheta \in \mathbb{V}$, $t \in \mathbb{T}$: $\mathcal{U}_{t'}(l, \vartheta, t)$ iff $\mathcal{U}_{t'}(X_l(t, \vartheta))$.

Proof. Assume $\mathcal{U}_{t'}(l, \vartheta, t)$. By Lemma 3.2, we only have to consider the case in which $t \leq t'$. Hence, $\vartheta, \vartheta + t' - t \models \iota(l)$ must hold. But then, there must exist a valuation ν, such that $\nu(u) = t'$, which is a sufficient condition for proving $\mathcal{U}_{t'}(X_l(t, \vartheta))$.
Assume $\mathcal{U}_{t'}(X_l(t, \vartheta))$. Hence, there must be a valuation ν, such that $\vartheta, \vartheta + \nu(u) - t \models \iota(l)$ and $t' \leq \nu(u)$. Therefore, also $\mathcal{U}_{\nu(u)}(l, \vartheta, t)$, and hence also $\mathcal{U}_{t'}(l, \vartheta, t)$. $\square$

The Lemmas 5.1 and 5.2 express that there exists a strong link between a location and specific process terms that are part of the process algebraic interpretations of a timed automaton. This relation is formalised by Theorem 5.1.

Theorem 5.1 (*Soundness*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be timed automata. The process algebraic interpretation of Def. 5.2 is sound:

$$X_1 \equiv X_2 \Leftrightarrow [\![X_1]\!] = [\![X_2]\!]$$

Proof. If $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$, then also $R' : [\![X_1]\!] \underline{\leftrightarrow}_t [\![X_2]\!]$, where

$$R' = \{\langle X_{1\,l}(\vartheta, t), X_{2\,s}(\chi, u)\rangle | (l, \vartheta, t) R (s, \chi, u) \wedge \vartheta, \chi \in \mathbb{V} \wedge t, u \in \mathbb{T} \wedge l \in L_1 \wedge s \in L_2\}$$

If $R : [\![X_1]\!] \underline{\leftrightarrow}_t [\![X_2]\!]$, then also $R' : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$, where

$$R' = \{(l, \vartheta, t), (s, \chi, u)\rangle | X_{1\,l}(\vartheta, t)\ R\ X_{2\,s}(\chi, u) \wedge \vartheta, \chi \in \mathbb{V} \wedge u, t \in \mathbb{T} \wedge l \in L_1 \wedge s \in L_2\}$$

In both cases, the fact that R' is a bisimulation follows directly from Lemmas 5.1 and 5.2. $\square$

In section 3.1, parallel composition was defined both on the level of timed automata and on the semantical level. There, we showed both ways of removing parallelism from a model essentially are equivalent. For the process algebraic interpretation, there again are two ways of dealing with parallelism. One way is by using the rules for removing parallelism on the level of the timed automaton. This way, however, parallelism cannot be dealt with in a compositional fashion. The other option is to introduce a process algebraic construct translating parallelism on the level of timed automata to parallelism on the level of the process algebraic interpretations of timed automata. This is formalised in Def. 5.3.

Definition 5.3 (*Process Algebraic Interpretation of Synchronising Timed Automata*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be timed automata. Assume the existence of a set of valuations for the clocks in the set $\mathcal{C}_1 \cup \mathcal{C}_2$, denoted by $\mathbb{V}$. Let $[\![X_1\|_s X_2]\!] : \mathcal{P}$ be the process algebraic interpretation of $X_1\|_s X_2$, defined by:

$$[\![X_1\|_s X_2]\!] \quad = \quad \rho_R[\ \partial_{\Sigma_1\cap\Sigma_2}\ ([\![X_1]\!]\ \|\ [\![X_2]\!])]$$

Communication is defined by the binary function γ, given by $\gamma(\sigma,\sigma') = \overline{\sigma}$ if $\sigma = \sigma'$ and δ otherwise. Here, $\overline{\sigma} \notin \Sigma_1 \cup \Sigma_2$. Renaming is defined by the function $R : \overline{\Sigma}_1 \cap \overline{\Sigma}_2 \to (\Sigma_1 \cap \Sigma_2)$ as $R(\overline{\sigma}) = \sigma$ for each $\overline{\sigma} \in \overline{\Sigma}_1 \cap \overline{\Sigma}_2$.

The process algebraic interpretation of a synchronising timed automaton should exhibit behaviour that is in some sense equivalent to the behaviour of the synchronising timed automaton itself. This means that it should synchronise on shared labels in the same way synchronising timed automata synchronise on shared labels. Lemma 5.3 formalises this idea.

Lemma 5.3 (*Coupled Blocking Synchronisation*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be arbitrary timed automata. Then for all $l \in L_1, s \in L_2, \vartheta, \chi \in \mathbb{V}, u, t, r \in \mathbb{T}, \sigma \in \Sigma_1 \cap \Sigma_2$,
$(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_{t'} (l',\vartheta',t')\|_s(s',\chi',t')$ iff
$\rho_R\partial_{\Sigma_1\cap\Sigma_2}(t \uparrow u)\gg_\bullet(X_{1\,l}(t,\vartheta)\|X_{2\,s}(u,\chi)) \xrightarrow{\sigma}_{t'} \rho_R\partial_{\Sigma_1\cap\Sigma_2}(t' \uparrow t')\gg_\bullet(X_{1\,l'}(t',\vartheta')\|X_{2\,s'}(t',\chi'))$

Proof. We will only consider one direction of the lemma. The proof for the other direction is almost identical to this one. Assume $(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_{t'} (l',\vartheta',t')\|_s(s',\chi',t')$. Since $\sigma \in \Sigma_1 \cap \Sigma_2$, this can only be the case when $(l,\vartheta,t) \xrightarrow{\sigma}_{t'} (l',\vartheta',t')$ and $(s,\chi,u) \xrightarrow{\sigma}_{t'} (s',\chi',t')$ and $t \uparrow u \leq t'$. According to 5.1, we may now conclude that $X_{1\,l}(t,\vartheta) \xrightarrow{\sigma}_{t'} X_{1\,l'}(t',\vartheta')$ and $X_{2\,s}(u,\chi) \xrightarrow{\sigma}_{t'} X_{2\,s'}(t',\chi')$ and $t \uparrow u \leq t'$. Based on the operational semantics of our process algebra, we can now deduce $X_{1\,l}(t,\vartheta)\|X_{2\,s}(u,\chi) \xrightarrow{\sigma}_{t'} X_{1\,l'}(t',\vartheta')\|X_{2\,s'}(t',\chi')$ and $t \uparrow u \leq t'$ and hence, also the desired $\rho_R\partial_{\Sigma_1\cap\Sigma_2}(t\ \uparrow u)\gg_\bullet(X_{1\,l}(t,\vartheta)\|X_{2\,s}(u,\chi)) \xrightarrow{\sigma}_{t'} \rho_R\partial_{\Sigma_1\cap\Sigma_2}t'\gg_\bullet(X_{1\,l'}(t',\vartheta')\|X_{2\,s'}(t',\chi'))$. □

Lemma 5.3 expresses that a location of a synchronising timed automaton and the process term that is part of the process algebraic interpretation of the synchronising timed automaton agree on synchronising behaviour. However, this is only part of the behaviour that is reflected when composing systems via parallel composition. If one of the components of the synchronising timed automaton can execute an action without having to synchronise, this action should also be matched by the process term that is part of the process algebraic interpretation of the synchronising timed automaton. This is expressed in Lemma 5.4.

Lemma 5.4 (*Coupled Interleaving*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1\rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2\rangle$ be arbitrary timed automata. Then for all $l \in L_1, s \in L_2, \vartheta, \chi \in \mathbb{V}, u, t, r \in \mathbb{T}, \sigma \in \Sigma_1 \setminus \Sigma_2$,
$(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_{t'} (l',\vartheta',t')\|_s(s,\chi,u)$ iff
$\rho_R\partial_{\Sigma_1\cap\Sigma_2}(t \uparrow u)\gg_\bullet(X_{1\,l}(t,\vartheta)\|X_{2\,s}(u,\chi)) \xrightarrow{\sigma}_{t'} \rho_R\partial_{\Sigma_1\cap\Sigma_2}(t' \uparrow u)\gg_\bullet(X_{1\,l'}(t',\vartheta')\|X_{2\,s}(u,\chi))$

Proof. We will again provide only a proof for the implication in one direction. The proof for the other direction follows the same principle. We will work module timed bisimulation. Assume $(l,\vartheta,t)\|_s(s,\chi,u) \xrightarrow{\sigma}_{t'} (l',\vartheta',t')\|_s(s,\chi,u)$. Since $\sigma \in \Sigma_1 \setminus \Sigma_2$, this can only be due to $(l,\vartheta,t) \xrightarrow{\sigma}_{t'} (l',\vartheta',t')$ and $\mathcal{U}_{t'}(s,\chi,u)$ and $u \leq t'$. Based on Lemmas 5.1 and 5.2, this is equivalent to $X_{1\,l}(t,\vartheta) \xrightarrow{\sigma}_{t'} X_{1\,l'}(t',\vartheta')$ and $\mathcal{U}_{t'}(X_{2\,s}(u,\chi))$ and $u \leq t'$. Since we now have one process that is able to execute an action at time t' and another process that is able to idle until time t', we have $X_{1l}(t,\vartheta) \| X_{2s}(u,\chi) \xrightarrow{\sigma}_{t'} X_{1l'}(t',\vartheta') \| (t' \gg_\bullet X_{2s}(u,\chi))$ and $u \leq t'$. Based on this and the fact that $u \leq t'$, we also have $(t \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi)) \xrightarrow{\sigma}_{t'} (t' \uparrow u) \gg_\bullet (X_{1l'}(t',\vartheta') \| (t' \gg_\bullet X_{2s}(u,\chi)))$ and $u \leq t'$. According to Lemma 4.1 the latter process term is bisimilar to the process term $(t' \uparrow u) \gg_\bullet (X_{1l'}(t',\vartheta') \| X_{2s}(u,\chi))$. Since $\sigma \in \Sigma_1 \setminus \Sigma_2$, we also have $\rho_R \partial_{\Sigma_1 \cap \Sigma_2}(t \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi)) \xrightarrow{\sigma}_{t'} \rho_R \partial_{\Sigma_1 \cap \Sigma_2}(t' \uparrow u) \gg_\bullet (X_{1l'}(t',\vartheta') \| X_{2s}(u,\chi))$ □

Again, the execution of actions is only part of the dynamic behaviour of a system. In some sense, the process term corresponding to the synchronisation of two locations should exhibit the same idling behaviour. This is expressed by Lemma 5.5.

Lemma 5.5 (*Coupled Idle Behaviour*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1 \rangle$ and $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2 \rangle$ be arbitrary timed automata. Then for all $l \in L_1, s \in L_2, \vartheta, \chi \in \mathbb{V}, u, t, t' \in \mathbb{T}$,
$\mathcal{U}_{t'}(l,\vartheta,t) \|_s (s,\chi,u))$ iff
$\mathcal{U}_{t'}(\rho_R \partial_{\Sigma_1 \cap \Sigma_2}(t \ \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi)))$

Proof. Assume $\mathcal{U}_{t'}((l,\vartheta,t) \|_s (s,\chi,u))$. This can only be because both $\mathcal{U}_{t'}(l,\vartheta,t)$ and $\mathcal{U}_{t'}(s,\chi,u)$. According to Lemma 5.2, we then also have $\mathcal{U}_{t'}(X_{1l}(t,\vartheta))$ and $\mathcal{U}_{t'}(X_{2s}(u,\chi))$ and by definition, we then also have $\mathcal{U}_{t'}(X_{1l}(t,\vartheta) \| X_{2s}(u,\chi))$ and thus $\mathcal{U}_{t'}(\rho_R \partial_{\Sigma_1 \cap \Sigma_2}(t \ \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi)))$.
Assume $\mathcal{U}_{t'}(\rho_R \partial_{\Sigma_1 \cap \Sigma_2}(t \ \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi)))$. This is due to $\mathcal{U}_{t'}(X_{1l}(t,\vartheta) \| X_{2s}(u,\chi))$, which in turn must be due to $\mathcal{U}_{t'}(X_{1l}(t,\vartheta))$ and $\mathcal{U}_{t'}(X_{2s}(u,\chi))$. By Lemma 5.2, we then also have both $\mathcal{U}_{t'}(l,\vartheta,t)$ and $\mathcal{U}_{t'}(s,\chi,u)$ and hence the desired $\mathcal{U}_{t'}((l,\vartheta,t) \|_s (s,\chi,u))$. □

Theorem 5.2 (*Soundness of synchronisation*).
Let $X_1 = \langle L_1, L_1^0, \Sigma_1, \mathcal{C}_1, \iota_1, E_1 \rangle$, $X_2 = \langle L_2, L_2^0, \Sigma_2, \mathcal{C}_2, \iota_2, E_2 \rangle$ and $X_3 = \langle L_3, L_3^0, \Sigma_3, \mathcal{C}_3, \iota_3, E_3 \rangle$ be timed automata, then the following equation holds:

$$X_1 \|_s X_2 \equiv X_3 \Leftrightarrow [\![X_1 \|_s X_2]\!] = [\![X_3]\!]$$

Proof. If $R : X_1 \|_s X_2 \leftrightarroweq_t X_3$, then $R' : [\![X_1 \|_s X_2]\!] \leftrightarroweq_t [\![X_3]\!]$, where

$$\begin{aligned} R' \quad = \quad & \{\langle \rho_R \partial_{\Sigma_1 \cap \Sigma_2}((t \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi))), X_{3q}(v,\zeta) \rangle \\ & | ((l,\vartheta,t),(s,\chi,u)) R (q,\zeta,v) \wedge \vartheta,\chi,\zeta \in \mathbb{V} \wedge t,u,v \in \mathbb{T} \wedge l \in L_1 \wedge s \in L_2 \wedge q \in L_3 \} \end{aligned}$$

If $R : [\![X_1 \|_s X_2]\!] \leftrightarroweq_t [\![X_3]\!]$, then $R' : X_1 \|_s X_2 \leftrightarroweq_t X_3$, where

$$\begin{aligned} R' \quad = \quad & \{\langle ((l,\vartheta,t),(s,\chi,u)),(q,\zeta,v) \rangle \\ & | \rho_R \partial_{\Sigma_1 \cap \Sigma_2}((t \uparrow u) \gg_\bullet (X_{1l}(t,\vartheta) \| X_{2s}(u,\chi))) R X_{3q}(v,\zeta) \wedge \vartheta,\chi,\zeta \in \mathbb{V} \wedge t,u,v \in \mathbb{T} \wedge \\ & l \in L_1 \wedge s \in L_2 \wedge q \in L_3 \} \end{aligned}$$

□ □

Both Theorems 5.1 and 5.2 express that timed automata can be translated to process algebra whilst still retaining the original semantical view of a process. This means that we can reason about timed automata using equational theories.

Note that an inverse translation only exists for a subset of process terms. For instance, the process term $\sum_{t:\mathbb{T}} a^{\varsigma}t \cdot b^{\varsigma}2t \cdot \delta^{\varsigma}\mathbf{0}$ cannot be translated to a timed automaton. This is due to the limitations that are posed on the clock constraints.

5.2 Interpretations of Hybrid Automata

The focus of this section will be on hybrid automata. There is a great deal of overlap between timed automata and hybrid automata. Hence, any process algebraic term serving as the interpretation of a hybrid automaton will have many characteristics of a timed automaton. Most proofs in this section are adaptations of the proofs given in the previous section.

First, a number of (intuitive) abbreviations are introduced, supporting readability.

Definition 5.4 (*Abbreviations*).
Let $X = \langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$ be a hybrid automaton. Then, the set $E_m \subseteq E$ is defined as follows:

$$E_m = \{m \xrightarrow{\sigma,\varphi,\rho} m' | \sigma \in \Sigma \wedge m' \in M \wedge \varphi \in \Phi(\mathcal{V}) \wedge \rho \in \Delta(\mathcal{V} \cup \mathcal{V}')\}$$

For all edges $e \in E_m$, we will use the following abbreviations:

$$\begin{array}{lcl} \sigma_e & \overset{\text{abbr.}}{=} & \pi_2(e) \\ \varphi_e & \overset{\text{abbr.}}{=} & \pi_3(e) \\ \rho_e & \overset{\text{abbr.}}{=} & \pi_4(e) \\ m_e & \overset{\text{abbr.}}{=} & \pi_5(e) \end{array}$$

The interpretation of hybrid automata is provided in Def. 5.5.

Definition 5.5 (*Process Algebraic Interpretations of a Hybrid Automaton*).
Let $X = \langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$ be a hybrid automaton. Then $[\![X]\!] : \mathcal{P}$ is the process algebraic interpretation of X, where

$$\begin{array}{lcl} [\![X]\!] & = & \sum_{m:M^0,\vartheta_{\mathbf{0}}:\mathbb{V}} [\vartheta_{\mathbf{0}} \models \mathcal{I}(m)] \ ::\!\rightarrow X_m(\mathbf{0}, \vartheta_{\mathbf{0}}) \\ & & \\ X_m & = & \lambda t : \mathbb{T}.\lambda\vartheta : \mathbb{V}.t \gg_{\bullet} \sum_{u:\mathbb{T},f:\mathcal{D}^{\mathcal{V}}_{u-t}} [f \models \iota(m) \wedge f(\mathbf{0}) = \vartheta \wedge (f \cup \dot{f})]\omega_f \models \theta(m)] \ ::\!\rightarrow \\ & & \sum_{e:E_m,\vartheta':\mathbb{V}} (\sigma_e \cdot X_{m_e}(u, \vartheta') \lhd f(u-t) \models \varphi_e \ \wedge f(u-t) \cup \vartheta' \models \kappa_e \rhd \delta\ ) \text{‘}u \end{array}$$

To provide a better intuition on the process algebraic interpretation, we will again employ a small and instructive example.

Example 5.2. Let X be the hybrid automaton depicted in Fig. 7. The automaton models a thermostat that measures a room's temperature, switching on and off a heater if necessary.

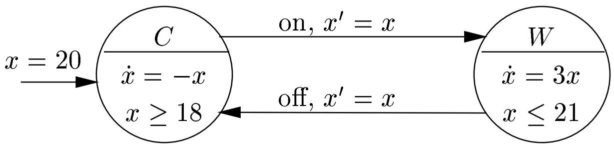

Figure 7: *A simple thermostat*

The automaton is straightforwardly translated to the process algebraic term $[\![X]\!]$:

$$\begin{array}{lcl} [\![X]\!] & = & X_C(\mathbf{0}, 20) \\ X_C & = & \lambda t : \mathbb{T}.\lambda x : \mathbb{R}.\ t \gg_{\bullet} \sum_{u:\mathbb{T},f:\mathcal{D}^{\mathbb{R}}_{u-t}} \\ & & [(\forall_{v\in[\mathbf{0},u-t]} f(v) \geq 18) \wedge f(\mathbf{0}) = x \wedge (\forall_{v\in\omega_f} \dot{f}(v) = -f(v))] \ ::\!\rightarrow \text{on}\text{‘}u \cdot X_W(u, f(u-t)) \\ X_W & = & \lambda t : \mathbb{T}.\lambda x : \mathbb{R}.\ t \gg_{\bullet} \sum_{u:\mathbb{T},f:\mathcal{D}^{\mathbb{R}}_{u-t}} \\ & & [(\forall_{v\in[\mathbf{0},u-t]} f(v) \leq 21) \wedge f(\mathbf{0}) = x \wedge (\forall_{v\in\omega_f} \dot{f}(v) = 3f(v))] \ ::\!\rightarrow \text{off}\text{‘}u \cdot X_C(u, f(u-t)) \end{array}$$

Using the process algebraic equational theory would allow for a further reduction of the complexity of the above process algebraic term.

In the process algebraic interpretation, the behaviours of a control mode of a hybrid automaton are captured by a "dedicated" process term. Hence, the behaviour of the process term and the control mode should be strongly coupled. This relation is expressed in Lemma 5.6.

Lemma 5.6 (*Coupled Behaviour*).
Let $X = \langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$ be an arbitrary hybrid automaton. Then, for all $m \in M$, $\vartheta \in \mathbb{V}$, $t \in \mathbb{T}$:
$(m, \vartheta, t) \xrightarrow{\sigma}_{t'} (m', \vartheta', t')$ iff $X_m(t, \vartheta) \xrightarrow{\sigma}_{t'} X_{m'}(t', \vartheta')$

Proof. Assume $(m, \vartheta, t) \xrightarrow{\sigma}_{t'} (m', \vartheta', t')$. Hence, there must exist a control switch $e' = m \xrightarrow{\sigma,\varphi,\rho} m'$, such that there exists a witness $g \in \mathcal{D}^{\mathbb{V}}_{t'-t}$, for which the conditions $g(\mathbf{0}) = \vartheta$ and $g \models \iota(m)$ and $(g \cup \dot{g})\rceil\omega_g \models \theta(m)$ and $g(t'-t) \models \varphi$ and $g(t'-t) \cup \vartheta' \models \rho$ are true. This means there exists a valuation ν, such that $\nu(e) = e'$ and $\nu(u) = t'$ and $\nu(f) = g$ and $\nu(\vartheta') = \vartheta'$. Hence, $X_m(t, \vartheta) \xrightarrow{\sigma}_{t'} X_{m'}(t', \vartheta')$.
Assume $X_m(t, \vartheta) \xrightarrow{\sigma}_{t'} X_{m'}(t', \vartheta')$. Hence, there must exist a valuation ν, such that $\nu(e) = m \xrightarrow{\sigma,\varphi,\rho} m'$ and $\nu(u) = t'$, and $\nu(f) \models \iota(m)$ and $(\nu(f)\nu(\dot{f}))\rceil\omega_{\nu(f)} \models \theta(m)$ and $\nu(f)(t'-t) \models \varphi$ and $\nu(f)(t'-t) \cup \nu(\vartheta') \models \rho$ and $\nu(\vartheta') = \vartheta'$. But then also $(m, \vartheta, t) \xrightarrow{\sigma}_{t'} (m', \vartheta', t')$. □

The behaviour of a control mode is not only dictated by the visible actions it prescribes, however, it is also determined by the ultimate idling time it can portray. The "idling behaviour" of the control mode and its dedicated process term is also strongly coupled with the "idling behaviour" of the control mode itself. Their relation is formalised in Lemma 5.7.

Lemma 5.7 (*Coupled Idling Time*).
Let $X = \langle M, M^0, \Sigma, \mathcal{V}, \iota, \theta, \mathcal{I}, E\rangle$ be an arbitrary hybrid automaton. Then, for all $m \in M$, $\vartheta \in \mathbb{V}$, $t \in \mathbb{T}$:
$\mathcal{U}_{t'}(m, \vartheta, t)$ iff $\mathcal{U}_{t'}(X_m(t, \vartheta))$.

Proof. Assume $\mathcal{U}_{t'}(m, \vartheta, t)$. By Lemma 3.6, we only have to consider the case in which $t \leq t'$. Hence, there must exist a witness $g \in \mathcal{D}^{\mathbb{V}}_{t'-t}$, such that $g(\mathbf{0}) = \vartheta$ and $g \models \iota(m)$ and $(g \cup g)\rceil\omega_g \models \theta(m)$. But this is a sufficient condition for the existence of a valuation ν, such that $\nu(f) = g$ and $\nu(u) = t'$. Hence, we also have $\mathcal{U}_{t'}(X_m(t, \vartheta))$.
Assume $\mathcal{U}_{t'}(X_m(t, \vartheta))$. Hence, there must be a valuation ν, such that $\nu(u) = t'$ and $\nu(f)(0) = \vartheta$ and $\nu(f) \models \iota(m)$ and $(\nu(f) \cup \nu(\dot{f}))\rceil\omega_{\nu(f)} \models \theta(m)$. This is a sufficient condition to deduce $\mathcal{U}_{t'}(m, \vartheta, t)$.
□

Lemmas 5.6 and 5.7 show that there exists a strong relation between a control mode and specific process terms of the process algebraic interpretations of a hybrid automaton. This relation is formalised by Theorem 5.3. This theorem actually formalises the soundness of the process algebraic interpretation.

Theorem 5.3 (*Soundness*).
Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \mathcal{V}_1, \iota_1, \theta_1, \mathcal{I}_1, E_1\rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \mathcal{V}_2, \iota_2, \theta_2, \mathcal{I}_2, E_2\rangle$ be two arbitrary hybrid automata. The relation between hybrid automata and their process algebraic interpretation is governed by the following equation:

$$X_1 \equiv X_2 \Leftrightarrow [\![X_1]\!] = [\![X_2]\!]$$

Proof. If $R : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$, then also $R' : [\![X_1]\!] \underline{\leftrightarrow}_t [\![X_2]\!]$, where

$$R' = \{\langle X_{1m}(\vartheta, t), X_{2n}(\chi, u)\rangle | (m, \vartheta, t) R (n, \chi, u) \wedge \vartheta, \chi \in \mathbb{V} \wedge t, u \in \mathbb{T} \wedge m \in M_1 \wedge n \in M_2\}$$

If $R : [\![X_1]\!] \underline{\leftrightarrow}_t [\![X_2]\!]$, then also $R' : X_1^{\overline{s}} \underline{\leftrightarrow}_t X_2^{\overline{s}}$, where

$$R' = \{(m, \vartheta, t), (n, \chi, u)\rangle | X_{1m}(\vartheta, t) \ R \ X_{2n}(\chi, u) \wedge \vartheta, \chi \in \mathbb{V} \wedge u, t \in \mathbb{T} \wedge m \in M_1 \wedge n \in M_2\}$$

In both cases, the fact that R' is a bisimulation follows immediately from Lemmas 5.6 and 5.7. □

The parallel composition of two hybrid automata is defined similar to the parallel composition of timed automata. In fact, this is expressed by the exact same process algebraic interpretations of synchronisation of hybrid automata and synchronisation of timed automata. This is illustrated by Def. 5.6.

Definition 5.6 (*Process Algebraic Interpretation of Synchronising Hybrid Automata*).
Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \mathcal{V}_1, \iota_1, \theta_1, \mathcal{I}_1, E_1\rangle$ and $X_2 = \langle M_2, M_2^0, \Sigma_2, \mathcal{V}_2, \iota_2, \theta_2, \mathcal{I}_2, E_2\rangle$ be hybrid automata. Assume the existence of a set of all valuations for the variables in $\mathcal{V}_1 \cup \mathcal{V}_2$, denoted by $\mathbb{V}$. Let $[\![X_1 \|_s X_2]\!] : \mathcal{P}$ be the process algebraic interpretation of $X_1 \|_s X_2$, defined by:

$$[\![X_1 \|_s X_2]\!] \quad = \quad \rho_R[\ \partial_{\Sigma_1 \cap \Sigma_2}\ ([\![X_1]\!] \ \| [\![X_2]\!])\ ]$$

Here, communication is defined by the binary function γ, given by $\gamma(\sigma, \sigma') = \overline{\sigma}$ if $\sigma = \sigma'$ and δ otherwise. Here, $\overline{\sigma} \notin \Sigma_1 \cup \Sigma_2$. Renaming is defined by the function $R : \overline{\Sigma}_1 \cap \overline{\Sigma}_2 \to (\Sigma_1 \cap \Sigma_2)$ as $R(\overline{\sigma}) = \sigma$ for each $\overline{\sigma} \in \overline{\Sigma}_1 \cap \overline{\Sigma}_2$.

Without proof, we state that the results of the previous section (i.e. Lemmas 5.3, 5.4 and 5.5) also hold for hybrid automata. A combination of these lemmas and Lemmas 5.6 and 5.7, immediately lead to the proof of Theorem 5.4.

Theorem 5.4 (*Soundness of synchronisation*).
Let $X_1 = \langle M_1, M_1^0, \Sigma_1, \mathcal{V}_1, \iota_1, \theta_1, \mathcal{I}_1, E_1\rangle$, $X_2 = \langle M_2, M_2^0, \Sigma_2, \mathcal{V}_2, \iota_2, \theta_2, \mathcal{I}_2, E_2\rangle$ and
$X_3 = \langle M_3, M_3^0, \Sigma_3, \mathcal{V}_3, \iota_3, \theta_3, \mathcal{I}_3, E_3\rangle$ be hybrid automata, then the following equation holds:

$$X_1 \|_s X_2 \equiv X_3 \ \Leftrightarrow\ [\![X_1 \|_s X_2]\!] = [\![X_3]\!]$$

Proof. If $R : X_1 \|_s X_2 \underline{\leftrightarrow}_t X_3$, then $R' : [\![X_1 \|_s X_2]\!] \underline{\leftrightarrow}_t [\![X_3]\!]$, where

$$\begin{aligned} R' \quad = \quad & \{\langle \rho_R \partial_{\Sigma_1 \cap \Sigma_2}((t \uparrow u) \gg_\bullet (X_{1l}(t, \vartheta) \| X_{2s}(u, \chi))), X_{3q}(v, \zeta)\rangle \\ & | ((l, \vartheta, t), (s, \chi, u)) R (q, \zeta, v) \wedge \vartheta, \chi, \zeta \in \mathbb{V} \wedge t, u, v \in \mathbb{T} \wedge l \in L_1 \wedge s \in L_2 \wedge q \in L_3\} \end{aligned}$$

If $R : [\![X_1 \|_s X_2]\!] \underline{\leftrightarrow}_t [\![X_3]\!]$, then $R' : X_1 \|_s X_2 \underline{\leftrightarrow}_t X_3$, where

$$\begin{aligned} R' \quad = \quad & \{\langle ((l, \vartheta, t), (s, \chi, u)), (q, \zeta, v)\rangle \\ & | \rho_R \partial_{\Sigma_1 \cap \Sigma_2}((t \uparrow u) \gg_\bullet (X_{1l}(t, \vartheta) \| X_{2s}(u, \chi))) R X_{3q}(v, \zeta) \wedge \vartheta, \chi, \zeta \in \mathbb{V} \wedge t, u, v \in \mathbb{T} \wedge \\ & l \in L_1 \wedge s \in L_2 \wedge q \in L_3\} \end{aligned}$$

□

6 Closing Remarks

The work discussed in this paper is not unprecedented. In [9], a translation of timed automata to process terms in the process algebra ACP with prefix integration is mentioned. However, timed automata as discussed in [9] are based upon [3]. As such, the notion of *invariants* in a timed automaton has not been taken into account.

In [8], a different approach is taken. There, timed automata serve as the underlying model for the axiom system. This contrasts with our approach, in which we relate a timed automaton and its process algebraic interpretation based on the underlying model of the timed automaton.

To our knowledge, no attempt has yet been made to provide a process algebraic basis for hybrid automata, or a class of hybrid automata. The process algebraic interpretation provided in this paper could lead to a dedicated axiom system which takes hybrid automata as its underlying model, much like has been done in [8].

Translations of hybrid automata to process algebra are interesting since they provide an algebraic perspective on the theory of hybrid automata. Moreover, the translation discussed in this paper provides a basis to extend the process algebra $\mu\mathrm{CRL}_t^\bullet$ to a hybrid process algebra, capable of modelling and analysing hybrid systems. Such hybrid process algebras are important, as they allow for a fundamental investigation of the interplay between continuous variables and discrete events.

In this respect, the difference at the semantical level between hybrid automata and $\mu\mathrm{CRL}_t^\bullet$ requires more attention. It is not yet clear whether the time-reflexive-transitive closure property (also sometimes referred to as *time additivity* or *stutter closure*), present in $\mu\mathrm{CRL}_t^\bullet$ but in general absent in hybrid automata, is too restrictive when dealing with hybrid systems. Absence of this property leads to smaller equivalence classes that are induced by the timed variants of strong bisimulation. As we have seen, this results in the differentiating between systems that exhibit the exact same behaviour when focusing on the time at which actions can occur. On the other hand, the time-reflexive-transitive closure property is an accepted property of systems, suggesting that the property should also hold for (acceptable) hybrid automata.

The translation of timed automata to process algebra discussed in this paper might also provide some insight in the way the renowned techniques for performing automated analysis of timed systems can be applied to process algebras. Techniques that provide some heuristics for algebraic verifications of timed systems would be welcomed in the field of timed process algebras.

Acknowledgements

The author would like to thank Jos Baeten and Kees Middelburg for the numerous discussions and their valuable comments on previous versions of this paper. Furthermore, Jan Friso Groote, Michel Reniers and Mark Voorhoeve are thanked for their valuable comments.

References

[1] ISO/IEC 8807. *A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, 1988.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[3] R. Alur and D.L. Dill. A theory of timed automata. In *Theoretical Computer Science*, volume 126, pages 183–235, 1994.

[4] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[5] J. C. M. Baeten and W. P . Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[6] M. Bezem and J.F. Groote. Invariants in process algebra with data. In *International Conference on Concurrency Theory*, pages 401–416, 1994.

[7] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In *Computer Networks and ISDN Systems*, 1987.

[8] P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems.* PhD thesis, Twente University, November 1999.

[9] W.J. Fokkink. An elimination theorem for regular behaviours with integration. In *Proceedings CONCUR 93*, number 715 in LNCS, pages 432–446. Springer-Verlag, 1993.

[10] J.F. Groote. The syntax and semantics of timed μCRL. Technical Report SEN-R9709, CWI, June 1997.

[11] J.F. Groote, M.R. Reniers, J.J. van Wamel, and M.B. van der Zwaag. Completeness of Timed μCRL. Technical Report SEN-R0034, CWI, November 2000.

[12] J.F. Groote and J.J. van Wamel. Analysis of three hybrid systems in timed μCRL. *Science of Computer Programming*, 39:215–247, 2001.

[13] T. A. Henzinger. The theory of hybrid automata. In *the Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 96)*, pages 278–292, 1996.

[14] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[15] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, 1998.

[16] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[17] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[18] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[19] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–583, October 1969.

[20] K. Larsen, P. Pettersson, and W. Yi. Compositional and symbolic model-checking of real-time systems. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, 1995.

[21] K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Springer International Journal of Software Tools for Technology Transfer*, 1, 1997.

[22] N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, number 1066 in LNCS, pages 496–510. Springer Verlag, 1996.

[23] R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[24] G.D. Plotkin. A Structural Approach to Operational Semantics. Technical Report Daimi Fn-19, University of Aarhus, Department of Computer Science, 1981.

[25] J. Sifakis and S. Yovine. Compositional specifications of timed systems. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science, STACS'96, Grenoble, France*, number 1046 in LNCS, pages 347–359. Springer-Verlag, February 1996.

[26] T.A.C. Willemse. A Process Algebraic Approach to Hybrid Systems. In J.P. Veen, editor, *Proceedings of the first PROGRESS workshop*. Technology Foundation, The Netherlands, October 2000.

A Overview of μCRL$_t^\bullet$

In this appendix we give an overview of all axioms of pCRL$_t^\bullet$ and μCRL$_t^\bullet$.

$$t = t' \qquad \text{for every } t = t' \in E$$

$$\frac{t = t'}{t[e/v] = t'[e/v]} \qquad \text{for every } v \in V_s \text{and } e \in T_s$$

$$\frac{t_1 = t_1' \cdots t_n = t_n'}{F(t_1, \cdots, t_n) = F(t_1', \cdots, t_n')} \qquad \text{for every } F : s_1 \times \cdots \times s_n \to s' \in \Sigma$$

$$\frac{t = t'}{\sum_v t = \sum_v t'}$$

$$t = t \qquad \frac{t = t'}{t' = t} \qquad \frac{t_1 = t_2 \quad t_2 = t_3}{t_1 = t_3}$$

Table 17: *Generalised equational logic for pCRL$_t^\bullet$. E represents the set of axioms of pCRL$_t^\bullet$ and Σ is the signature of the theory pCRL$_t^\bullet$.*

A1	$x+y$	$=$	$y+x$
A2	$x+(y+z)$	$=$	$(x+y)+z$
A3	$x+x$	$=$	x
A4	$(x+y)\cdot z$	$=$	$x\cdot z+y\cdot z$
A5	$(x\cdot y)\cdot z$	$=$	$x\cdot(y\cdot z)$
A6$^-$	$a+\delta$	$=$	a
A7	$\delta\cdot x$	$=$	δ
SUM1	$\sum_v\ x$	$=$	x
SUM3	$\sum_v\ p$	$=$	$\sum_v\ p\ +p$
SUM4	$\sum_v\ (p+q)$	$=$	$\sum_v\ p+\sum_v\ q$
SUM5	$(\sum_v\ p)\cdot x$	$=$	$\sum_v\ (p\cdot x)$
SUM12'	$(\sum_v\ p)\lhd b\rhd\overset{\bullet}{\delta}$	$=$	$\sum_v p\lhd b\rhd\overset{\bullet}{\delta}$
PE	$p\lhd eq(v,w)\rhd\overset{\bullet}{\delta}$	$=$	$p[v/w]\lhd eq(v,w)\rhd\overset{\bullet}{\delta}$
C1	$x\lhd \mathsf{t}\rhd y$	$=$	x
C2	$x\lhd \mathsf{f}\rhd y$	$=$	y
C3	$x\lhd b\rhd y$	$=$	$x\lhd b\rhd\overset{\bullet}{\delta}+y\lhd\neg b\rhd\overset{\bullet}{\delta}$
C4	$(x\lhd b_1\rhd y)\lhd b_2\rhd y$	$=$	$x\lhd(b_1\wedge b_2)\rhd y$
C5	$x\lhd b_1\rhd\overset{\bullet}{\delta}+x\lhd b_2\rhd\overset{\bullet}{\delta}$	$=$	$x\lhd(b_1\vee b_2)\rhd\overset{\bullet}{\delta}$
C6	$(x\lhd b\rhd y)\cdot z$	$=$	$x\cdot z\lhd b\rhd y\cdot z$
C7	$(x+y)\lhd b\rhd z$	$=$	$x\lhd b\rhd z+y\lhd b\rhd z$
SCA	$(x\lhd b\rhd\overset{\bullet}{\delta})\cdot(y\lhd b\rhd\overset{\bullet}{\delta})$	$=$	$x\cdot y\lhd b\rhd\overset{\bullet}{\delta}$
AT1	x	$=$	$\sum_t\ x\,{}^{\varsigma}t$
AT2	$a\,{}^{\varsigma}t\cdot x$	$=$	$a\,{}^{\varsigma}t\cdot(t\gg x)$
ATA1	$a\,{}^{\varsigma}t\,{}^{\varsigma}u$	$=$	$(a\,{}^{\varsigma}t\lhd u\le t\rhd\delta\,{}^{\varsigma}t)\lhd t\le u\rhd\delta\,{}^{\varsigma}u$
ATA2	$(x+y)\,{}^{\varsigma}t$	$=$	$x\,{}^{\varsigma}t+y\,{}^{\varsigma}t$
ATA3	$(x\cdot y)\,{}^{\varsigma}t$	$=$	$x\,{}^{\varsigma}t\cdot y$
ATA4	$(\sum_v\ p)\,{}^{\varsigma}t$	$=$	$(\sum_v\ p\,{}^{\varsigma}t)$
ATA5	$(x\lhd b\rhd y)\,{}^{\varsigma}t$	$=$	$x\,{}^{\varsigma}t\lhd b\rhd y\,{}^{\varsigma}t$
ATB1	$t\gg a\,{}^{\varsigma}u$	$=$	$a\,{}^{\varsigma}u\lhd t\le u\rhd\delta\,{}^{\varsigma}t$
ATB2	$t\gg(x+y)$	$=$	$t\gg x+t\gg y$
ATB3	$t\gg(x\cdot y)$	$=$	$(t\gg x)\cdot y$
ATB4	$t\gg\sum_v p$	$=$	$\sum_v t\gg p$
ATB5	$t\gg(x\lhd b\rhd y)$	$=$	$(t\gg x)\lhd b\rhd(t\gg y)$
ATB6	$t\gg\overset{\bullet}{\delta}$	$=$	$\overset{\bullet}{\delta}$
ID1	$x+\overset{\bullet}{\delta}$	$=$	x
ID2	$\overset{\bullet}{\delta}\cdot x$	$=$	$\overset{\bullet}{\delta}$
ID3	$\overset{\bullet}{\delta}\,{}^{\varsigma}t$	$=$	$\overset{\bullet}{\delta}$

Table 18: *Axioms of* $pCRL_t^{\bullet}$*:* $x,y,z\in V_P$*, process-closed* $p,q\in T_P, a\in AT_\delta, t,u\in V_{\mathbb{T}}, b,b_1,b_2\in V_{\mathbb{B}}$ *and* $v,w\in V$*.*

CM1	$x \Vert y$	$=$	$x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \vert y$
CM2	$a{}^{\backprime}t \mathbin{\lfloor\!\lfloor} x$	$=$	$(a{}^{\backprime}t \ll x) \cdot x$
CM3	$a{}^{\backprime}t \cdot x \mathbin{\lfloor\!\lfloor} y$	$=$	$(a{}^{\backprime}t \ll y) \cdot (t \gg x \Vert y)$
CM4	$(x + y) \mathbin{\lfloor\!\lfloor} z$	$=$	$x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$
CM5	$a \cdot x \vert a'$	$=$	$(a \vert a') \cdot x$
CM6	$a \vert a' \cdot x$	$=$	$(a \vert a') \cdot x$
CM7	$a \cdot x \vert a' \cdot y$	$=$	$(a \vert a') \cdot (x \Vert y)$
CM8	$(x + y) \vert z$	$=$	$x \vert z + y \vert z$
CM9	$x \vert (y + z)$	$=$	$x \vert z + x \vert z$
H8	$x \vert (y \lhd b \rhd z)$	$=$	$(x \vert y) \lhd b \rhd (x \vert z)$
H8'	$(x \lhd b \rhd y) \vert z$	$=$	$(x \vert z) \lhd b (y \vert z)$
H18	$(x \lhd b \rhd y) \mathbin{\lfloor\!\lfloor} z)$	$=$	$(x \mathbin{\lfloor\!\lfloor} z) \lhd b \rhd (y \mathbin{\lfloor\!\lfloor} z)$
CF	$\mathsf{a}(d) \vert \mathsf{a}'(e)$	$=$	$\begin{cases} \gamma(\mathsf{a}, \mathsf{a}') \lhd eq(d, e) \rhd \delta & \\ \quad \text{if } \gamma(\mathsf{a}, \mathsf{a}') \text{ defined} & \\ \delta \quad \text{otherwise} & \end{cases}$
CD1	$\delta \vert a''$	$=$	δ
CD2	$a'' \vert \delta$	$=$	δ
CD3	$\overset{\bullet}{\delta} \vert a$	$=$	$\overset{\bullet}{\delta}$
CD4	$a \vert \overset{\bullet}{\delta}$	$=$	$\overset{\bullet}{\delta}$
ATA7	$(x \vert y){}^{\backprime}t$	$=$	$x{}^{\backprime}t \vert y$
ATA8	$(x \vert y){}^{\backprime}t$	$=$	$x \vert y{}^{\backprime}t$
SUM6	$(\sum_v p) \mathbin{\lfloor\!\lfloor} x$	$=$	$\sum_v (p \mathbin{\lfloor\!\lfloor} x)$
SUM7	$(\sum_v p) \vert x$	$=$	$\sum_v (p \vert x)$
SUM7'	$x \vert (\sum_v p)$	$=$	$\sum_v (x \vert p)$

Table 19: *Axioms for parallelism of* $\mu CRL_t^{\bullet}$*, where* $x, y, z \in V_P$*, process-closed* $p \in T_P, a, a' \in AT_{\delta}^{\bullet}, a'' \in AT_{\delta}, \mathsf{a}, \mathsf{a}' \in Act, b \in V_{\mathbb{B}}, t \in V_{\mathbb{T}}, v \in V$ *and* $d \in \mathcal{D}_s$

DD	$\partial_H(\delta)$	$=$	δ	
DID	$\partial_H(\overset{\bullet}{\delta})$	$=$	$\overset{\bullet}{\delta}$	
D1	$\partial_H(\mathsf{a}(d))$	$=$	$\mathsf{a}(d)$	if $\mathsf{a} \notin H$
D2	$\partial_H(\mathsf{a}(d))$	$=$	$\delta(d)$	if $\mathsf{a} \in H$
D3	$\partial_H(x+y)$	$=$	$\partial_H(x) + \partial_H(y)$	
D4	$\partial_H(x \cdot y)$	$=$	$\partial_H(x) \cdot \partial_H(y)$	
D5	$\partial_H(x \triangleleft b \triangleright y)$	$=$	$\partial_H(x) \triangleleft b \triangleright \partial_H(y)$	
D6	$\partial_H(\sum_v p)$	$=$	$\sum_v \partial_H(p)$	
D7	$\partial_H(x‘t)$	$=$	$\partial_H(x)‘t$	
RD	$\rho_R(\delta)$	$=$	δ	
RID	$\rho_R(\overset{\bullet}{\delta})$	$=$	$\overset{\bullet}{\delta}$	
R1	$\rho_R(\mathsf{a}(d))$	$=$	$R(\mathsf{a})(d)$	if $\mathsf{a} \in \mathsf{dom}(R)$
R2	$\rho_R(\mathsf{a}(d))$	$=$	$\mathsf{a}(d)$	if $\mathsf{a} \notin \mathsf{dom}(R)$
R3	$\rho_R(x+y)$	$=$	$\rho_R(x) + \rho(y)$	
R4	$\rho_R(x \cdot y)$	$=$	$\rho_R(x) \cdot \rho_R(y)$	
R5	$\rho_R(x \triangleleft b \triangleright y)$	$=$	$\rho_R(x) \triangleleft b \triangleright \rho_R(y)$	
R6	$\rho_R(\sum_v p)$	$=$	$\sum_v \rho_R(p)$	
R7	$\rho_R(x‘t)$	$=$	$\rho_R(x)‘t$	
ATC1	$x \ll a‘t$	$=$	$\sum_u x‘u \triangleleft u \leq t \triangleright x‘t$	
ATC2	$x \ll (y+z)$	$=$	$x \ll y + x \ll z$	
ATC3	$x \ll y \cdot z$	$=$	$x \ll y$	
ATC4	$x \ll \sum_v p$	$=$	$\sum_v x \ll p$	
ATC5	$x \ll (y \triangleleft b \triangleright z)$	$=$	$(x \ll y) \triangleleft b \triangleright (x \ll z)$	
ATC6	$x \ll \overset{\bullet}{\delta}$	$=$	$\overset{\bullet}{\delta}$	

Table 20: *Time related axioms of* $\mu CRL_t^{\bullet}$*, where* $x, y, z \in V_P$*, process-closed* $p \in T_P, \mathsf{a} \in Act, a \in AT_\delta, b \in V_{\mathbb{B}}, t, u \in V_{\mathbb{T}}, v \in V$ *and* $H \subseteq Act, R \subseteq Act \times Act$ *and* $d \in \mathcal{D}_s$