# Syntax and Consistent Equation Semantics of Hybrid Chi †

D.A. van Beek[1], K.L. Man[2],
M.A. Reniers[2], J.E. Rooda[1], R.R.H. Schiffelers[1]
[1]Department of Mechanical Engineering
[2]Department of Mathematics and Computer Science
Eindhoven University of Technology, P.O.Box 513
5600 MB Eindhoven, The Netherlands
{d.a.v.beek,k.l.man,m.a.reniers,j.e.rooda,r.r.h.schiffelers}@tue.nl

†This work was partially supported by the EU project HyCon (FP6-IST-511368)

# Abstract

The hybrid $\chi$ (Chi) formalism integrates concepts from dynamics and control theory with concepts from computer science, in particular from process algebra and hybrid automata. It integrates ease of modeling with a straightforward, structured operational semantics. Its 'consistent equation semantics' enforces state changes to be consistent with invariants as in most hybrid automata. Ease of modeling is ensured by means of the following concepts: 1) different classes of variables: discrete and continuous, of subclass jumping or non-jumping, and algebraic; 2) strong time determinism of alternative composition in combination with delayable guards; 3) integration of urgent and non-urgent actions; 4) differential algebraic equations as a process term as in mathematics; 5) steady-state initialization; and 6) several user-friendly modeling extensions. Furthermore, the Chi language incorporates several concepts for complex system specification: 1) process terms for scoping that integrate abstraction, local variables, local channels and recursion definitions; 2) process definition and instantiation that enable process re-use, encapsulation, hierarchical and/or modular composition of processes; and 3) different interaction mechanisms: handshake synchronization and synchronous communication for discrete-event processes that do not share variables, and shared variables for continuous-time processes. The syntax and semantics are illustrated using many different examples. Furthermore, general translations from hybrid automata and PWA systems to $\chi$ are given.

# Contents

# Chapter 1

# Introduction

Hybrid systems related research is based on two, originally different, world views: on the one hand the dynamics and control (DC) world view, and on the other hand the computer science (CS) world view.

The DC world view is that of a predominantly continuous-time system, which is modeled by means of differential (algebraic) equations, or by means of a set of trajectories. Hybrid phenomena are modeled by means of discontinuous functions and/or switched equation systems. The evolution of a hybrid system in the continuous-time domain is considered as a set of functions of time (one for each variable), where each function is continuous apart from a finite number of points.

Analysis and synthesis of hybrid systems in the DC domain are done, among others, by means of piecewise affine (PWA) systems, mixed logic dynamical (MLD) systems or linear complementarity (LC) systems, see [25] for an overview relating these different classes, and see Section 5.2 for a translation of PWA systems to hybrid $\chi$ (hybrid Chi). A different kind of hybrid systems are differential (algebraic) equations with discontinuous right-hand sides, the semantics of which is defined using differential inclusions. Such differential inclusions allow modeling of relays, valves or any kind of on/off switching elements at a high level of abstraction in control systems with so-called sliding modes [23, 44].

The CS world view is that of a predominantly discrete-event system. A well-known model is a (hybrid) automaton, but modeling of discrete-event systems is also based on, among others, process algebra, Petri nets, and data flow languages. For modeling and analysis of hybrid phenomena, discrete-event formalisms are extended in different ways with some form of differential (algebraic) equations. The most influential hybrid system model is that of a hybrid automaton such as defined in [2, 5, 27]. An essential difference between a hybrid automaton and a DC hybrid system model is that where in a DC hybrid model there are no actions, in a hybrid automaton, discontinuities can take place *only* by means of (labeled) actions. By means of actions, a hybrid automaton switches from one mode/location to another mode/location, where in each mode/location all variables behave according to a continuous function of time.

The DC and CS world views each have their own advantages. The CS world view is obviously better suited to discrete-event modeling. On the other hand, a DC model $\dot{x} = \text{step}(t-1)$, where $t$ denotes time and step is a discontinuous function such that step$(x)$ is 0 for $x < 0$ and 1 for $x \geq 0$, is much more straightforward than the corresponding hybrid automaton. Such a hybrid automaton requires two locations, one that specifies behavior for $t \leq 1$ and one for $t \geq 1$, that are connected by an edge with a transition relation and action label.

Clearly, hybrid systems represent a domain where the DC and CS world views meet, and we believe that a formalism that integrates the DC and CS world views is a valuable contribution towards integration of the DC and CS methods, techniques, and tools. The hybrid $\chi$ language is such a formalism. On the one hand, it can deal with continuous-time systems, PWA/MLD/LC systems, and hybrid systems based on sets of ordinary differential equations using discontinuous functions (the DC approach), in combination with algebraic constraints. On the other hand, it can deal with discrete-event systems, without continuous variables or differential equations, and with hybrid systems such that in each mode behavior is continuous (the CS approach).

The intended use of hybrid $\chi$ is for modeling, simulation, verification, and real-time control. Its application domain ranges from physical phenomena, such as dry friction, to large and complex manufacturing systems. Although the semantics is formally defined, including a solution concept, the straightforward and elegant syntax and semantics is also highly suited to non-computer scientists. In the remainder of this paper, we usually refer to hybrid $\chi$ as $\chi$.

The most important concepts in $\chi$ are summarized below:

1. Integration between the DC and CS world views.

   - The DC world view in $\chi$ allows modeling of hybrid phenomena by means of discontinuous functions and/or switched equation systems. For this purpose, $\chi$ has introduced the category of algebraic variables, the trajectory of which can be discontinuous. Furthermore, the convex equality operator, defined in [46], but not explained in detail in this paper, allows modeling of differential inclusions according to the Filippov solution concept [23]. The solution concept has been completely formalized in $\chi$.
   - The CS world view in $\chi$ allows modeling of hybrid phenomena in a way that is strongly influenced by hybrid automata such as defined in [5, 27]. In this respect, the new hybrid $\chi$ language differs considerably from its predecessor defined in [41] which was quite different from hybrid automata. In the new $\chi$ formalism described in this paper, discontinuous changes are modeled by means of actions, and the 'consistent equation semantics' enforces state changes to be consistent with delay predicates, that include the invariant and flow clauses of hybrid automata. This is expressed by the property $p \parallel x = e \underline{\leftrightarrow} p[e/x] \parallel x = e$, that, although not yet proven, we expect to hold. Here, $p[e/x]$ denotes the process term obtained from $p$ by substituting every free occurrence of variable $x$ by its defining expression $e$. For example: $x := y \parallel y = 1$ is bisimilar to $x := 1 \parallel y = 1$. A difference between the consistent equation semantics and hybrid automata is that where the $\chi$ semantics considers $\dot{x} = 1 \wedge \dot{x} = 2$ to be an inconsistent process term, the hybrid automaton with flow clause $\dot{x} = 1 \wedge \dot{x} = 2$ can

enter the location, but cannot delay. The inconsistent process in a hybrid automaton is a location with invariant false.

2. Integration of a straightforward semantics and ease of modeling.

   An important aspect is the conceptual similarity with hybrid automata as mentioned in the previous item. The concepts from hybrid automata have been extended in several ways to facilitate modeling. Where hybrid automata have one category of variables, namely jumping continuous variables, hybrid $\chi$ has, among others, the following categories:

   - Discrete variables, that are comparable to locations in hybrid automata. The concept of discrete variables allows for compact readable specifications. In hybrid automata such variables are sometimes mimicked by real valued variables with a derivative of zero. However, for non-real valued variables, such as variables of type string, the concept of a zero derivative cannot be used.
   - Jumping continuous variables, that correspond to the continuous variables in hybrid automata. The values of these variables are in principle allowed to jump (change) arbitrarily in an action transition, as long as the resulting values satisfy the action/jump predicate. Consider for example a system with four variables: $w, x, y, z$. If the value of $x$ should change to 1, and the other variables should remain unchanged, the action/jump predicate should be $x' = 1, w' = w, y' = y, z' = z$, or $x^+ = 1, w^+ = w^-, y^+ = y^-, z^+ = z^-$, depending on the syntax. Restrictions of the type $v^+ = v^-$ clutter the models, and are therefore often omitted in informal hybrid automata specifications. In order to allow fully formal models, without the clutter associated with the restrictions on non-jumping variables, $\chi$ has an additional class of variables: the non-jumping continuous variables.
   - Non-jumping continuous variables. The values of these variables are not allowed to change in action transitions, unless their changes are explicitly specified, for example by means of assigning a new value to such a variable.
   - Algebraic variables that can have discontinuous trajectories, as already discussed in the item on integration between the DC and CS world views.

   Other concepts that enable integration of a straightforward semantics and ease of modeling are:

   - Strong time deterministic alternative composition operator. Where in the previous version of hybrid $\chi$ [41] and discrete-event $\chi$ [13] the passage of time could result in making a choice between the two operands of the alternative composition operator (weak time determinism), as is the case in many process algebras, in the current hybrid $\chi$ semantics, the passage of time can never result in such a choice. In fact, the passage of time can only result in changes to valuations (values of variables). It cannot result in any other changes in process terms. In the previous versions of $\chi$, alternative composition $\dot{x} = 1 \,[]\, x := 1$ could non-deterministically choose between doing a delay of $t$ to $\dot{x} = 1$, or doing the (undelayable) action $x := 1$ and then terminate. Strong time deterministic alternative composition means that alternative composition can delay only

if both process terms can delay together, so that $\dot{x} = 1 \,[]\, x := 1$ can only do the (non-delayable) action $x := 1$, and then terminate. Hybrid automata have a comparable choice mechanism, apart from initialization. In a hybrid automaton, action transitions cannot disappear as a result of time passing. They can only be disabled for the period of time that the guard evaluates to false in the valuation prescribed by the trajectory of the variables. Also, time passing cannot result in the choice of a different location. The only changes in a hybrid automaton as a result of time passing are changes in the values of the variables. Only initially, depending on the initial edges and invariants, different initial locations may be selected as a result of time passing.

- Delayable guards. Where the previous version of hybrid $\chi$ [41] and discrete-event $\chi$ [13] had non-delayable guards, such as found in many process algebras, the current hybrid $\chi$ semantics has delayable guards. A non-delayable guard cannot perform a delay when it is false. A delayable guard can delay when it is false until it becomes true, and thus facilitates modeling. Consider for example a valve $\alpha$ that must be switched on when the temperature $T$ becomes bigger then $T_{\max}$. Using a delayable guard, this can be modeled simply by $T \geq T_{\max} \rightarrow \alpha := \text{true}$.
  Delayable guards ensure that in $b \rightarrow h\,!\,b$, the value of expression $b$ that is sent via channel $h$ is always true. Note that $h\,!\,b$ can either do the send action, or delay for an arbitrary period of time. Non-delayable guards may lead to un-intuitive behavior, because the value of $b$ that is sent may be false. Consider the process term:

$$x := 0;\ (\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \,[]\, \Delta 10) \parallel \Delta 5;\ h\,?\,y),$$

  where $\Delta\, s$ can delay for $t$ time-units to $\Delta\, s - t$, and $\Delta\, 0$ can terminate by means of an internal action.
  Using non-delayable guards, the process term can perform the assignment, followed by a delay of at most 5, and after an internal action transforms into

$$\dot{x} = 1 \parallel (h\,!\,x \,[]\, \Delta 5) \parallel h\,?\,y.$$

  The guard that was true has disappeared in delaying. If the communication via channel $h$ takes place now, a value of 5 is sent, which does not conform to $x \leq 3$.
  Using delayable guards on the other hand, the process term can do the assignment followed by a delay of at most 3, and transforms into:

$$\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \,[]\, \Delta 7) \parallel \Delta 2;\ h\,?\,y,$$

  where the value of $x$ is 3. Communication is still not possible. After a delay of 2, followed by an internal action, the process term transforms into:

$$\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \,[]\, \Delta 5) \parallel h\,?\,y,$$

  where the value of $x$ is 5, and after another delay of 5 it transforms into:

$$\dot{x} = 1 \parallel (x \leq 3 \rightarrow h\,!\,x \,[]\, \Delta 0) \parallel h\,?\,y.$$

  The time-out takes place, leading to: $\dot{x} = 1 \parallel h\,?\,y$. Due to the delayable guard, that does not disappear while delaying, the communication does not take place, because the guard cannot be satisfied.

- Integrated urgent and non-urgent actions. Where hybrid automata such as [2, 33] have non-urgent actions only, and the previous versions of $\chi$ had urgent actions only, the current hybrid $\chi$ formalism has both. The concept of urgency is defined in a very flexible way: non-delayable actions are by definition urgent and delayable actions are non-urgent. This is achieved without any additional operators. A maximal progress operator as defined in [13] is not needed. The concept of urgency is built into the individual parallel composition, alternative composition and guard operators. Consider the non-delayable action $x := 1$. The following three process terms
  - $\dot{x} = 1 \parallel x := 1$
  - $\dot{x} = 1 \,[]\, x := 1$
  - $x = 0 \curvearrowright (\dot{x} = 1 \parallel x \leq 0 \rightarrow x := 1)$

  can each execute only the action $x := 1$. Here, $x = 0 \curvearrowright p$ denotes a process term $p$ for which the value of $x$ is initially zero. Consider now the delayable action $[x := 1]$. The following three process terms
  - $\dot{x} = 1 \parallel [x := 1]$
  - $\dot{x} = 1 \,[]\, [x := 1]$
  - $x = 0 \curvearrowright (\dot{x} = 1 \parallel x \leq 0 \rightarrow [x := 1])$

  can each execute either the action $x := 1$ or perform a delay. This concept is comparable to so-called urgent transitions that are present in, for example UPPAAL [31], and in certain types of hybrid automata, such as HYTECH [5].

  Communication on channels can also be urgent and non-urgent as in UPPAAL. This is achieved by means of an operator that partitions the set of channels into a set of urgent and a set of non-urgent channels. For the urgent channels, communication must take place as soon as it becomes possible, whereas for the non-urgent channels, no such preference for communication is assumed.
- Equations as in mathematics. Differential algebraic equations are process terms in hybrid $\chi$. Therefore, they are modeled in $\chi$ in the same way as in mathematics.
- Steady state initialization. Dynamical analysis of physical systems often starts in initial steady-state conditions. This means that the initial state is such that all derivatives are zero. In $\chi$, steady state initialization can be easily expressed by means of the signal emission operator. For example, $\dot{x} = 0 \curvearrowright \dot{x} = -x + 1$ represents the steady state initialization ($\dot{x} = 0$ ) of model $\dot{x} = -x + 1$. This means that initially the value of $x$ will be 1. In general, steady state initialization is not possible in this way for hybrid automata, because initial edges and invariants are predicates over variables, not derivatives. When the equations are straightforward enough, the modeler can explicitly calculate steady state conditions. In the example, variable $x$ could be initialized to 1.
- Modeling extensions. Ease of modeling is further supported in hybrid $\chi$ by extension of the small set of orthogonal core process terms with additional process terms for ease of modeling. These additional process terms can be expressed by means of a straightforward translation into the core process terms.

3. Concepts for complex system specification.

   This includes many concepts already present in previous versions of hybrid $\chi$ [41]:

   - Process terms for scoping that integrate abstraction, local variables, local channels and recursion definitions.
   - Parameterized process definition and process instantiation that enable:
     - process re-use and
     - encapsulation, hierarchical and/or modular composition of processes.
   - CSP communication and synchronization concepts that allow synchronization and communication without sharing of variables.
   - Shared variables, that enable modular composition of continuous or hybrid processes.

The history of the hybrid $\chi$ language dates back quite some time. It was originally designed as a modeling and simulation language for specification of discrete-event, continuous-time or combined discrete-event/continuous-time models. The first simulator [36], however, was suited to discrete-event models only. The simulator was successfully applied to a large number of industrial cases, such as an integrated circuit manufacturing plant, a brewery, and process industry plants [48]. Later, the hybrid language and simulator were developed [21, 47]. For the purpose of verification, the discrete-event part of the language was mapped onto the process algebra $\chi_\sigma$ by means of a syntactical translation. The semantics of $\chi_\sigma$ was defined using a structured operational semantics style (SOS), bisimulation relations were derived, and a model checker was built [13]. In this way, verification of discrete-event $\chi$ models was made possible [12]. The new hybrid $\chi$ language defined in this paper integrates the modeling language and the verification formalism. It integrates, extends and improves the syntax and semantics defined in [42] and [41].

The remainder of this paper is organized as follows. Section 2 describes the syntax of the hybrid $\chi$ language. In Section 3, the semantics of hybrid $\chi$ is formally specified. Several examples in Section 4 are used to illustrate the use of the language. General translation schemes for translating hybrid automata and PWA systems to $\chi$ are given in Section 5. In Section 6, a notion of equivalence is defined, and it is shown to be a congruence for all hybrid $\chi$ operators. Furthermore, some useful properties of closed hybrid $\chi$ process terms are given. Full proofs are presented in the appendices. Section 7 discusses related work, and Section 8 terminates with conclusions and points out directions of future work.

# Chapter 2

# Syntax and informal semantics of the Chi language

This section presents a concise definition of the syntax and informal semantics of hybrid $\chi$. The syntax definition is incomplete in the sense that the syntax of predicates, expressions, etc, is not defined.

## 2.1 Processes

A $\chi$ process is a triple $\langle p, \sigma, E \rangle$, where $p$ denotes a process term, $\sigma$ denotes a valuation, and $E$ denotes an environment. The syntax of process terms is introduced in Section 2.2. Variables in $\chi$ are used to store information, i.e., during execution variables have a value. A valuation is a partial function from variables to values. Syntactically, a valuation is denoted by a set of pairs $\{x_0 \mapsto c_0, \ldots, x_n \mapsto c_n\}$, where $x_i$ denotes a variable and $c_i$ its value. The valuation $\sigma$ and the environment $E$, together define the variables that exist in the $\chi$ process and the variable classes to which they belong.

The variables are grouped into different classes with respect to the delay behavior and action behavior. With respect to the delay behavior, the variables are divided into the following classes:

- The discrete variables, the values of which remain constant while delaying.
- The continuous variables, the values of which change according to a bounded, absolutely continuous function of time while delaying. The values of continuous variables are further restricted by delay predicates that are usually in the form of differential algebraic equations.
- The dotted continuous variables, the values of which change according to a bounded, possibly discontinuous function of time while delaying. The relation between the dotted continuous variables and the continuous variables is explained in Section 3.3.2.

- The algebraic variables, that behave in a similar way as continuous variables. The differences are that algebraic variables may change according to a discontinuous function of time, and algebraic variables are not allowed to occur as dotted variables.
- The predefined variable 'time', that denotes the current time.

With respect to the action behavior, the variables are divided into two classes:

- The non-jumping variables, the values of which by default do not change in actions. Such changes need to be explicitly specified.
- The jumping variables, the values of which by default can jump to arbitrary values in actions. The values after jumping can be restricted by means of the action predicate, or receive process term that caused the jump, or by means of delay predicates / equations.

The discrete and continuous variable classes can be divided into jumping and non-jumping versions. For the other classes, such a division is not possible: the dotted continuous and algebraic variables are by definition jumping with respect to the action behavior, and the predefined variable time is by definition non-jumping.

Further explanation on the semantics of the behavior of the different classes of variables is found in Section 3.3.1 on the action predicate, in Section 3.3.2 on the delay predicate, in Section 3.3.3 on the send and receive process terms, and in Section 3.4.6 on parallel composition.

In $\chi$, an environment is a tuple $(C, J, L, R)$, where $C$ denotes the set of continuous variables, $J$ denotes the set of jumping variables, and $L$ denotes the set of algebraic variables. It is required that $\mathrm{dom}(\sigma) \cap L = \emptyset$, $C \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, $J \subseteq (\mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}) \cup L$, and $(\mathrm{dom}(\sigma) \cup L) \cap \mathrm{dom}(R) = \emptyset$. In the environment, $R$ denotes a recursive process definition. A recursive process definition is a partial function from recursion variables to process terms. Syntactically, a recursive process definition is denoted by a set of pairs $\{X_0 \mapsto p_0, \ldots, X_m \mapsto p_m\}$, where $X_i$ denotes a recursion variable and $p_i$ the process term defining it.

The valuation $\sigma$ captures the values of those variables that are relevant for predicting the future behaviors of a process. The domain of the valuation $\sigma$ in a $\chi$ process $\langle p, \sigma, E \rangle$ consists of the discrete variables, the continuous variables, and the predefined non-jumping variable time. The dotted continuous variables and the algebraic variables are not included in the domain of $\sigma$, because their values depend only on the process term $p$, possibly together with the values of the other variables.

For a $\chi$ process $\langle p, \sigma, (C, J, L, R) \rangle$, the combination of the variable classes for the delay and action behavior leads to the following classes of variables:

- The set of discrete variables $D$ is $\mathrm{dom}(\sigma) \setminus (C \cup \{\mathsf{time}\})$.
  - the set of non-jumping discrete variables is $D \setminus J$,
  - the set of jumping discrete variables is $D \cap J$.

- The set of continuous variables is $C$,
  - the set of non-jumping continuous variables is $C \setminus J$,
  - the set of jumping continuous variables is $C \cap J$.
- The set of (jumping) dotted continuous variables is $\dot{C}$.
- The set of (jumping) algebraic variables is $L$.
- The predefined (non-jumping) variable denoting the current time is $\mathsf{time}$.

Note that it is possible to have $D \cap J \neq \emptyset$ and $L \cap J \neq \emptyset$. Such jumping discrete or jumping algebraic variables can occur as an artefact of the parallel composition of a send and a receive process term, where the receive process term assigns the received value to a discrete or algebraic variable, see Sections 3.3.3 and 3.4.6. From a modeling perspective, discrete or algebraic variables are in principle never explicitly declared as jumping. Discrete variables are not declared as jumping, because their value is not determined by equations. Algebraic variables are not declared as jumping, because they are by definition jumping. In fact, there is no difference between the behavior of an algebraic variable that is in set $J$ and one that is not in the set.

## 2.2 Process terms

Process terms $P$ are used both for modeling purposes and for the definition of the (structured operational) semantics. In Section 2.3, we extend the syntax of $\chi$ process terms with process terms $P_{\mathrm{M}}$ that are used for modeling purposes to ensure better readability of $\chi$ models. The semantics of those process terms is defined in terms of the process terms given in this section.

$$
\begin{array}{rcl}
P & ::= & W : r \gg l_{\mathrm{a}} \quad | \quad u \quad | \quad \delta \quad | \quad \bot \\
& | & [P] \quad | \quad u \curvearrowright P \quad | \quad P ; P \quad | \quad b \rightarrow P \quad | \quad P \,[]\, P \\
& | & P \parallel P \quad | \quad h \,!!\, \mathbf{e}_n \quad | \quad h \,??\, \mathbf{x}_n \quad | \quad \partial_A(P) \quad | \quad \upsilon_{\mathcal{H}}(P) \\
& | & X \quad | \quad \iota_{J^+}(P) \\
& | & [\![_{\mathrm{V}}\, \sigma_\bot, C, L \text{ `}|\text{'} P \,]\!] \quad | \quad [\![_{\mathrm{H}}\, H_0 \text{ `}|\text{'} P \,]\!] \quad | \quad [\![_{\mathrm{R}}\, R \text{ `}|\text{'} P \,]\!]
\end{array}
$$

An informal, concise explanation of this syntax, together with some additional (informal) definitions, is given below. Section 3 gives a more detailed account of their meaning. The operators are listed in descending order of their binding strength as follows ; , $\{\curvearrowright, \rightarrow\}$, $\{\parallel, []\}$. The operators inside the braces have equal binding strength. In addition, operators of equal binding strength associate to the right, and parentheses may be used to group expressions. For example, $p ; q ; r$ means $p ; (q ; r)$.

Strictly speaking, a $\chi$ process term $p$ cannot perform actions nor delays. Only the $\chi$ process $\langle p, \sigma, E \rangle$, that is obtained by adding a valuation and an environment to $p$, can, in principle, perform actions and delays. Therefore, when we informally refer to a process term that performs actions or delays, we refer to the process term together with a valuation and environment.

### 2.2.1 Manipulating the values of variables

In $\chi$, there are several classes of variables, and there are several means to change the value of a variable, depending on the class of variable. The main means for changing the value of variables are the action predicate, for instantaneous changes, and the delay predicate, for the changes of variables over time.

#### Action predicates

An instantaneous change of variables in $\chi$ is always connected to the execution of an action. In action predicates, the action is represented by a label. Other types of action are related to communication, which is treated below in the paragraph on parallelism. *Action predicate* $W : r \gg l_{\mathrm{a}}$ denotes instantaneous changes to the variables from set $W$, by means of an action labeled $l_{\mathrm{a}}$, such that predicate $r$ is satisfied. The predefined global variable time cannot be assigned. The action label $l_{\mathrm{a}}$ is taken from a given set $A_{\mathrm{label}}$ which at least contains the special action label $\tau$ representing the internal or silent step. The non-jumping variables that are not mentioned in $W$ remain unchanged, and the jumping variables, dotted continuous variables, and algebraic variables may obtain arbitrary values.

In this paper, we do not explicitly give a syntax for such predicates $r$. In $r$, variables, dotted continuous variables, and '$^-$' superscripted variables (not dotted) may occur. Of course the use of variables is restricted to the declared variables. A '$^-$' superscripted occurrence of a variable refers to the value of the variable in the extended valuation prior to execution of the action predicate, and a normal un-superscripted occurrence of a variable refers to the value of that variable in the extended valuation that results from the execution of the action predicate. An extended valuation is a valuation over the discrete variables, the continuous variables, the variable time, the algebraic variables, and the dotted continuous variables. A predicate $r$ is satisfied if evaluating the '$^-$' superscripted variables in the original extended valuation and evaluating the normal occurrences of the variables in the obtained extended valuation means that the predicate is true. The reason to use an extended valuation for evaluating action predicate $r$ is that in such predicates also algebraic and dotted continuous variables may be used. Note that it can be the case that different instantaneous changes satisfy the predicate, this may result in non-determinism.

#### Delay predicates

In principle, the continuous and algebraic variables change arbitrarily over time, although, depending on the class of the variable, they have to respect some continuity and boundedness requirements, see Section 3.3.2 for more details. More specific changes of such variables over time are defined by means of delay predicates. A *delay predicate* $u$, usually in the form of a differential algebraic equation, is a predicate over variables (including the variable time), and dotted continuous variables (e.g. $\dot{x}$, for $x$ a continuous variable). A delay predicate restricts the allowed behavior of the continuous and algebraic variables in such a way that the value of the

predicate remains true over time. As is common practice in mathematics, the comma in predicates denotes conjunction. E.g. $u_1, u_2$ denotes the predicate $u_1 \wedge u_2$. Also, both $e_1 \leq \dot{x} \leq e_2$ and $\dot{x} \in [e_1, e_2]$ can be used instead of $\dot{x} \geq e_1,\ \dot{x} \leq e_2$, and likewise for strict inequalities and open intervals.

### 2.2.2 Deadlock and inconsistency

In $\chi$, only consistent processes can do action or delay transitions, and the result of an action or delay transition is always a consistent process. Some process terms are consistent for certain valuations and inconsistent for other valuations. E.g. the delay predicate process term $x \geq 0$ is consistent for all values of $x$ greater or equal to zero. There are also process terms that are consistent or inconsistent for all valuations. The *inconsistent process term* $\bot$ is inconsistent for all valuations. It cannot perform any transition.

The *deadlock process term* $\delta$ cannot perform actions or delays. It is however consistent. Both process terms are needed for the specification of properties only. The consistency requirement enforces constraints on $\chi$ processes comparable to invariants in hybrid automata. Informally, in $\chi$ the delay predicates (equations) must always hold. Consistency ensures that in $x := 1 \parallel y = x$, the values of $x$ and $y$ are 1 after assigning 1 to $x$, independently of the initial value of $y$. Consistency also ensures that inconsistent processes cannot be reached, e.g. in $x := 1 \parallel x = 2$, the assignment to $x$ cannot be executed.

### 2.2.3 Delay enabling operator

Besides the specification of delay by means of delay predicates, arbitrary delay can be described by means of the *delay enabling operator* $[p]$. The resulting behavior is such that arbitrary delays that are consistent with the current state of the process are allowed. As a consequence, any delay behavior of $p$ is neglected. The only way to prevent this process term from delaying is the execution of an action by $p$.

### 2.2.4 Signal emission

*Signal emission operator* $u \curvearrowright p$, where $u$ denotes a predicate over variables and dotted continuous variables as before, requires the predicate $u$ to hold initially in the extended valuation. If this is the case, $u \curvearrowright p$ behaves as $p$, otherwise, the process is considered to be inconsistent.

### 2.2.5 Sequential composition

The *sequential composition* of process terms $p$ and $q$ behaves as process term $p$ until $p$ terminates, and then continues to behave as process term $q$.

### 2.2.6 Conditional

The *guarded process term* $b \rightarrow p$ can do whatever actions $p$ can do under the condition that the guard $b$ evaluates to true using the current extended valuation. All variables and dotted continuous variables are allowed to occur in $b$. The guarded process term can delay according to $p$ under the condition that for the intermediate extended valuations during the delay, the guard $b$ holds. The guarded process term can perform arbitrary delays under the condition that for the intermediate valuations during the delay, possibly excluding the first and last valuation, the guard $b$ does not hold.

### 2.2.7 Choice

The *alternative composition operator* [] allows a non-deterministic choice between different actions of a process. With respect to time behavior, the participants in the alternative composition have to synchronize. This means that the trajectories of the variables have to be agreed upon by both participants. This means that [] is a strong time-deterministic choice operator.

### 2.2.8 Parallelism

Parallelism can be specified by means of the *parallel composition operator* $\|$ . Parallel processes interact by means of shared variables or by means of synchronous point-to-point communication/synchronization via a channel. Channels are denoted as labels (identifiers). A set of channel labels $H$ is assumed. The parallel composition $p \| q$ synchronizes the time behavior of $p$ and $q$, interleaves the action behavior (including the instantaneous changes of variables) of $p$ and $q$, and synchronizes matching send and receive actions. The synchronization of time behavior means that only the time behaviors that are allowed by both $p$ and $q$ are allowed by their parallel composition.

By means of the *send action* $h\,!!\,\mathbf{e}_n$, where $\mathbf{e}_n$ denotes $e_1, \ldots, e_n$ for $n \geq 1$, the values of expressions $e_1, \ldots, e_n$ (evaluated w.r.t. the extended valuation) are sent via channel $h$. For $n = 0$, $h\,!!\,\mathbf{e}_n$ denotes $h\,!!$ and nothing is sent via the channel. By means of the *receive action* $h\,??\,\mathbf{x}_n$, where $\mathbf{x}_n$ denotes $x_1, \ldots, x_n$ for $n \geq 1$, values for $x_1, \ldots, x_n$ are received from channel $h$. For $n = 0$, $h\,??\,\mathbf{x}_n$ denotes $h\,??$, and nothing is received via the channel. Communication in $\chi$ is the sending of values by one parallel process over a channel to another parallel process, where the received values (if any) are stored in variables. In case no values are sent and received, we refer to synchronization instead of communication. For communication, the acts of sending and receiving (values) have to take place in different parallel processes at the same moment in time.

In order to be able to model open systems (i.e. systems that interface with the environment), it is necessary not to enforce communication over the external channels of the model (e.g. the channels that send or receive from the environment). For communication over internal channels, however, the communication of matching send and receive actions, often is not only an option, but an obligation. In such models, the separate occurrence of the send action and the receive

action over an internal channel is undesired. The *encapsulation operator* $\partial_{\mathcal{A}}$, where $\mathcal{A} \subseteq A \setminus \{\tau\}$ is a set of actions ($A$ is the set of all possible actions and $\tau$ is the predefined internal action), is introduced to block the actions from the set $\mathcal{A}$. In order to assure that for internal channels, only the synchronous execution of matching send and receive actions takes place, one can simply put all send and receive actions via internal channels in the set $\mathcal{A}$.

In principle the channels in $\chi$ are non-urgent. This means that communication does not necessarily take place as soon as possible. In order to describe also urgent channels, the *urgent communication operator* $\upsilon_{\mathcal{H}}(p)$, where $\mathcal{H} \subseteq H$ is a set of channel labels, ensures that $p$ can only delay in case no communication of send and receive events via a channel from $\mathcal{H}$ is possible.

### 2.2.9 Recursive definitions

*Process term X* denotes a recursion variable (identifier) that is defined either in the environment of the process, or in a recursion scope operator process term $[\![_{\mathrm{R}} \ldots \mid P \,]\!]$, see below. Among others, it is used to model repetition. Recursion variable $X$ can do whatever the process term of its definition can do.

### 2.2.10 Jump enabling operator

*Jump enabling operator* $\iota_{J^+}(p)$, where $J^+$ denotes a set of variables, is used to (re)define the variables in set $J^+$ as jumping variables.

### 2.2.11 Hierarchical modeling

Thus far, it has been assumed that all variables that are allowed to occur in a $\chi$ process term are either declared in the valuation or in the environment (in the set $L$). To support the hierarchical modeling of systems, it is convenient to allow local declarations of variables. For this purpose, the *variable scope operator* process term $[\![_{\mathrm{V}}\, \sigma_\perp, C, L \mid p \,]\!]$ is introduced, where $\sigma_\perp$ denotes a valuation of local variables, where values may be undefined ($\perp$), $C$ denotes a set of local (non-jumping) continuous variables, and $L$ denotes a set of local algebraic variables. The set of local discrete variables is $\mathrm{dom}(\sigma_\perp) \setminus C$. It is allowed that the local variables have been declared on a more global level already. Any occurrence of a variable from $\mathrm{dom}(\sigma) \cup \dot{C} \cup L$ in process term $p$ refers to the local variable and not to any more global declaration of the same variable name.

For similar purposes, local channels can be declared by means of a *channel scope* process term $[\![_{\mathrm{H}}\, H_0 \mid p \,]\!]$ and local recursive definitions by means of a *recursion scope* process term $[\![_{\mathrm{R}}\, R \mid p \,]\!]$. The channel scope process term $[\![_{\mathrm{H}}\, H_0 \mid p \,]\!]$ is used to declare the channels from the set $H_0 \subseteq H$ to be local. Communication actions via those local channels are abstracted from (replaced by internal action $\tau$) and the separate send and receive actions via local channels

are blocked. The recursion scope process term $[\![_{\mathrm{R}}\ R \mid p\ ]\!]$ is used to declare local recursion definitions by means of the set $R \subseteq RS$ (see Section 3.1 for the definition of $RS$).

## 2.3 Modeling extensions

For many of the process terms and operators introduced before, there is additional, more user-friendly syntax available, the so-called modeling extensions. In this section, all of these modeling extensions are expressed in terms of the syntax introduced in the previous section.

### 2.3.1 Processes

A $\chi$ model is of the following form:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s_1, \ldots, s_k \\
,\ \mathsf{cont}\ x_1, \ldots, x_n \\
,\ \mathsf{alg}\ z_1, \ldots, z_m \\
,\ \mathsf{chan}\ h_1, \ldots, h_l \\
,\ i \\
,\ X_1 \mapsto p_1,\ \ldots,\ X_r \mapsto p_r \\
\mid\ p \\
\rangle
\end{array}
$$

where

- $s_1, \ldots, s_k$ denote the discrete variables,
- $x_1, \ldots, x_n$ denote the non-jumping continuous variables,
- $z_1, \ldots, z_m$ denote the algebraic variables,
- $h_1, \ldots, h_l$ denote the channels,
- $i$ denotes an initialization predicate that restricts the allowed values of the variables initially,
- $X_1 \mapsto p_1,\ \ldots,\ X_r \mapsto p_r$ denote the recursion definitions,
- $p$ is a process term defining the behavior of the model.

Note that it is required that the discrete, continuous, and algebraic variables are all different. Note also that jumping continuous variables can be specified by redefining some of the (non-jumping) continuous variables $x_1, \ldots, x_n$ as jumping by means of the jump enabling operator $\iota_{J^+}(p)$, where $J^+ \subseteq \{x_1, \ldots, x_n\}$, or by means of its modeling equivalent $(\mathsf{jump}\ \ldots \mid p)$.

Besides the variables mentioned in the model defined above, the existence of the predefined reserved global variable time which denotes the current time, the value of which is initially zero, is assumed. This variable cannot be declared. It can only be used in expressions in process term $p$.

The above $\chi$ model is an abbreviation for the set of $\chi$ processes defined by:

$$
\begin{array}{l}
\langle\ \partial_{A_{\mathrm{ia}}}(\upsilon_{\{h_1,\ldots,h_l\}}((i \wedge \mathsf{time} = 0) \curvearrowright p)) \\
,\ \sigma_{sxt} \\
,\ (\ \{x_1, \ldots, x_n\} \\
\quad ,\ \emptyset \\
\quad ,\ \{z_1, \ldots, z_m\} \\
\quad ,\ \{X_1 \mapsto p_1,\ \ldots,\ X_r \mapsto p_r\} \\
\quad ) \\
\rangle,
\end{array}
$$

namely for each valuation $\sigma_{sxt}$, with $\mathrm{dom}(\sigma_{sxt}) = \{s_1, \ldots, s_k, \mathsf{time}, x_1, \ldots, x_n\}$, a separate $\chi$ process. In the $\chi$ process, $A_{\mathrm{ia}}$ represents the internal send and receive actions.

As a shorthand, the keyword preceeding variables of a certain type is omitted when there are no variables of that type, and the keyword chan is omitted when there are no local channel declarations. Also the initialization predicate $i$ and the recursive definitions $X_1 \mapsto p_1,\ \ldots,\ X_r \mapsto p_r$ may be omitted, indicating a predicate that always holds and an empty list of recursive definitions, respectively.

### 2.3.2 Process terms

For many of the process terms introduced before, there is additional, more user-friendly syntax available:

$$
\begin{array}{rcl}
P_{\mathrm{M}} & ::= & \mathrm{skip} \mid \mathbf{x}_n := \mathbf{e}_n \mid \mathbf{x}_n : r \mid h\,!\,\mathbf{e}_n \mid h\,?\,\mathbf{x}_n \\
& \mid & \Delta_d(P) \mid \Delta d \mid *P \mid *b : P \\
& \mid & (\mathsf{jump}\ \mathbf{y}_m\ \text{`}|\text{'}\ P) \\
& \mid & [\![\ \mathsf{disc}\ \mathbf{s}_k, \mathsf{cont}\ \mathbf{x}_n, \mathsf{alg}\ \mathbf{z}_l, \mathsf{chan}\ \mathbf{h}_m, i, L_R\ \text{`}|\text{'}\ P\ ]\!] \\
& \mid & l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)
\end{array}
$$

The operators of $P$ and $P_{\mathrm{M}}$ are listed in descending order of their binding strength as follows: $\{*, *:\}, ;\ , \{\curvearrowright,\ \rightarrow\ \}, \{\|\ , []\}$.

### Skip

Process term skip is an abbreviation for an action predicate that can only perform an internal action ($\tau$) without changing the valuation.

$$\boxed{\text{skip} \triangleq \emptyset : \text{true} \gg \tau}$$

### Multi-assignment

Multi-assignment $\mathbf{x}_n := \mathbf{e}_n$ for $n \geq 1$ is an abbreviation for an internal action that changes variables $x_1, \ldots, x_n$ to the values of expressions $e_1, \ldots, e_n$, respectively. For $n = 1$, this gives a normal assignment $x := e$.

$$\boxed{\mathbf{x}_n := \mathbf{e}_n \triangleq \{\mathbf{x}_n\} : x_1 = e_1^- \wedge \cdots \wedge x_n = e_n^- \gg \tau}$$

Here $e^-$ denotes the result of replacing all variables $v$ in $e$ by their '$^-$' superscripted version $v^-$. For example, the translation of process term $x := 2x + yz$ is defined as $\{x\} : x = 2x^- + y^- z^-$, and the translation of $x, y := x + y, x - y$ is defined as $\{x, y\} : (x = x^- + y^-) \wedge (y = x^- - y^-)$.

### Action predicate

Action predicate $\mathbf{x}_n : r$ denotes instantaneous changes to the variables $x_1, \ldots, x_n$, by means of an internal action $\tau$, such that predicate $r$ over variables, dotted variables, and '$^-$' superscripted variables is satisfied.

$$\boxed{\mathbf{x}_n : r \triangleq \{\mathbf{x}_n\} : r \gg \tau}$$

### Delayable send and receive

Process terms $h\,!\,\mathbf{e}_n$, and $h\,?\,\mathbf{x}_n$ are the respective delayable equivalents of $h\,!!\,\mathbf{e}_n$ and $h\,??\,\mathbf{x}_n$. They are defined by means of the delay enabling operator $[p]$, which adds arbitrary delay behavior to $p$.

$$\boxed{\begin{array}{lcl} h\,!\,\mathbf{e}_n & \triangleq & [h\,!!\,\mathbf{e}_n] \\ h\,?\,\mathbf{x}_n & \triangleq & [h\,??\,\mathbf{x}_n] \end{array}}$$

**Delay operators**

By means of the delay operator $\Delta_d(p)$, a process term is forced to delay for the amount of time units specified by the value of numerical expression $d$, and then proceeds as $p$. The abbreviation $\Delta d$ denotes a process term that first delays for $d$ time units, and then terminates by means of an internal action $\tau$.

$$\begin{array}{lcl} \Delta_d(p) & \triangleq & [\![_{\mathrm{V}}\, \{t \mapsto \bot\}, \emptyset, \emptyset \mid t = \mathsf{time} + d \curvearrowright \mathsf{time} \geq t \rightarrow p\, ]\!] \\ \Delta d & \triangleq & \Delta_d(\mathrm{skip}) \end{array}$$

In the definition of $\Delta_d(p)$, $t$ denotes a fresh variable, not occurring free in $p$. Delays are only defined for non-negative values of $d$. Therefore, we assume that the value of $d$ in the extended valuation is non-negative.

**Repetition operators**

Process term $*p$ represents the infinite repetition of process term $p$. Guarded repetition $*b : p$ can be interpreted as "while $b$ do $p$".

$$\begin{array}{lcl} *p & \triangleq & [\![_{\mathrm{R}}\, \{X \mapsto p;\ X\} \mid X\, ]\!] \\ *b : p & \triangleq & [\![_{\mathrm{R}}\, \{X \mapsto b \rightarrow \mathrm{skip};\ p;\ X\ [] \neg b \rightarrow \mathrm{skip}\} \mid X\, ]\!] \end{array}$$

In the definition of $*p$ and $*b : p$, recursion variable $X$ denotes a fresh recursion variable not occurring free in $p$.

**Jump enabling operator**

Jump enabling operator $(\mathsf{jump}\ \mathbf{y}_m \mid p)$, where $\mathbf{y}_m$ denotes a comma separated list of variables, is used to redefine the variables $\mathbf{y}_m$ as jumping variables.

$$(\mathsf{jump}\ \mathbf{y}_m \mid p) \triangleq \iota_{\{\mathbf{y}_m\}}(p)$$

**Scope operator**

The modeling scope operator process term

$$[\![\ \mathsf{disc}\ \mathbf{s}_k, \mathsf{cont}\ \mathbf{x}_n, \mathsf{alg}\ \mathbf{z}_l, \mathsf{chan}\ \mathbf{h}_m, i, L_R\ \text{`}|\text{'}\ p\, ]\!]$$

is used to declare a scope consisting of local discrete variables $s_1, \ldots, s_k$, local (non-jumping) continuous variables $x_1, \ldots, x_n$, local algebraic variables $z_1, \ldots, z_l$, local channels $h_1, \ldots, h_m$, initialization predicate $i$, and local recursion definition list $L_R$. The variables all have to be different.

$$
\begin{array}{l}
[\![\ \mathsf{disc}\ \mathbf{s}_k \\
\ ,\ \mathsf{cont}\ \mathbf{x}_n \\
\ ,\ \mathsf{alg}\ \mathbf{z}_l \\
\ ,\ \mathsf{chan}\ \mathbf{h}_m \\
\ ,\ i \\
\ ,\ L_R \\
\ \mid\ p \\
]\!]
\end{array}
\quad \triangleq \quad
\begin{array}{l}
[\![_{\mathrm{V}}\ \sigma_{sx} \\
\quad ,\ \ \{x_1, \ldots, x_n\} \\
\quad ,\ \ \{z_1, \ldots, z_m\} \\
\quad \mid\ \ [\![_{\mathrm{H}}\ \{h_1, \ldots, h_l\} \\
\qquad \mid\ \upsilon_{\{h_1,\ldots,h_l\}}([\![_{\mathrm{R}}\ \{L_R\} \mid i \curvearrowright p\ ]\!]) \\
\qquad ]\!] \\
]\!]
\end{array}
$$

Here $L_R$ denotes the recursion definitions $X_1 \mapsto p_1,\ \ldots,\ X_r \mapsto p_r$, $\sigma_{sx}$ denotes a valuation with $\mathrm{dom}(\sigma_{sx}) = \{s_1, \ldots, s_k,\ x_1, \ldots, x_n\}$, and $\sigma_{sx}$ is undefined for all elements from its domain: $\forall_{v \in \mathrm{dom}(\sigma_{sx})}\ \sigma_{sx}(v) = \bot$.

In a similar way as defined for $\chi$ processes, the keyword preceding variables of a certain type is omitted when there are no variables of that type, and the keyword chan is omitted when there are no local channel declarations. Also the initialization predicate $i$ and the recursion definitions may be omitted, indicating a predicate that always holds and an empty list of recursion definitions, respectively.

**Process instantiation**

Process instantiation process term $l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$, where $l_{\mathrm{p}}$ denotes a process label, enables (re)-use of a process definition. A process definition is specified once, but the associated processes can be instantiated many times, possibly with different parameters: external variables $\mathbf{x}_k$, external channels $\mathbf{h}_m$, and expressions $\mathbf{e}_n$.

Chi specifications in which process instantiations $l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$ are used have the following structure:

$$
\begin{array}{l}
pd_1 \\
\vdots \\
pd_j \\
\\
\langle\ \mathsf{disc}\ \ldots,\ \mathsf{cont}\ \ldots,\ \mathsf{alg}\ \ldots,\ \mathsf{chan}\ \ldots,\ i,\ L_R \mid p\ \rangle,
\end{array}
$$

where for each process instantiation $l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$ occurring in $p$, a matching process definition $pd_j$ of the form

$$
\begin{array}{l}
l_{\mathrm{p}}(\mathsf{ext}\ \mathbf{x}'_k, \mathsf{chan}\ \mathbf{h}'_m, \mathsf{val}\ \mathbf{v}_n) = \\
[\![\ \mathsf{disc}\ \mathbf{z}_d,\ \mathsf{cont}\ \mathbf{z}_c,\ \mathsf{alg}\ \mathbf{z}_a,\ \mathsf{chan}\ \mathbf{h}''_{m'},\ i,\ \mathbf{X} \mapsto \mathbf{p} \\
\mid\ p_{\mathrm{body}} \\
]\!]
\end{array}
$$

must be present among the $j$ process definitions $pd_1 \ldots pd_j$. Here $l_{\mathrm{p}}$ denotes a process label, $\mathbf{x}_k$ denotes the 'actual external' variables $x_1, \ldots, x_k$, $\mathbf{h}_m$ denotes the 'actual external' channels $h_1, \ldots, h_m$, $\mathbf{e}_n$ denotes the expressions $e_1, \ldots, e_n$, $\mathbf{x'}_k$ denotes the 'formal external' variables $x'_1, \ldots, x'_k$, $\mathbf{h'}_m$ denotes the 'formal external' channels $h'_1, \ldots, h'_m$, $\mathbf{v}_n$ denotes the 'value parameters' $v_1, \ldots, v_n$, $\mathbf{h''}_{m'}$ denotes the local channels $h''_1, \ldots, h''_{m'}$, $i$ denotes the initialization predicate, and $\mathbf{X} \mapsto \mathbf{p}$ denotes the recursion definitions $X_1 \mapsto p_1, \ \ldots, \ X_r \mapsto p_r$. In a similar way, $\mathbf{z}_d$, $\mathbf{z}_c$, and $\mathbf{z}_a$ denote comma separated lists of local variables.

In process term $p_{\mathrm{body}}$, apart from the local variables $\mathbf{z}_d$, $\mathbf{z}_c$, $\mathbf{z}_a$ and local channels $\mathbf{h''}_{m'}$, also the formal external variables $\mathbf{x'}_k$, formal external channels $\mathbf{h'}_m$, and value parameters $\mathbf{v}_n$ may be used. We assume that the formal external variables $\mathbf{x'}_k$, the value parameters $\mathbf{v}_n$, the local variables $\mathbf{z}_d$, $\mathbf{z}_c$, $\mathbf{z}_a$, and the recursion variables $\mathbf{X}$ are all different. In the same way, the formal external channels $\mathbf{h'}_m$ must be different from the local channels $\mathbf{h''}_{m'}$. Furthermore, all variables and channels used in $p_{\mathrm{body}}$ must be declared.

Formally, the syntactic translation of process instantiation

$$l_{\mathrm{p}}(\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n)$$

with corresponding process definition

$$\begin{array}{l} l_{\mathrm{p}}(\mathsf{ext}\ \mathbf{x'}_k, \mathsf{chan}\ \mathbf{h'}_m, \mathsf{val}\ \mathbf{v}_n) = \\ |[\ \mathsf{disc}\ \mathbf{z}_d\,,\ \mathsf{cont}\ \mathbf{z}_c\,,\ \mathsf{alg}\ \mathbf{z}_a\,,\ \mathsf{chan}\ \mathbf{h''}_{m'} \\ ,\ i \\ ,\ \mathbf{X} \mapsto \mathbf{p} \\ |\ p_{\mathrm{body}} \\ ]| \end{array}$$

is given by

$$\begin{array}{l} |[\ \mathsf{disc}\ \mathbf{z}_d, \mathbf{v}_n\,,\ \mathsf{cont}\ \mathbf{z}_c\,,\ \mathsf{alg}\ \mathbf{z}_a\,,\ \mathsf{chan}\ \mathbf{h''}_{m'} \\ ,\ i \wedge (\mathbf{v}_n = w) \\ ,\ \mathbf{X} \mapsto \mathbf{p} \\ |\ p_{\mathrm{body}} \\ ]|\ [\mathbf{x}_k, \mathbf{h}_m, \mathbf{e}_n / \mathbf{x'}_k, \mathbf{h'}_m, w]. \end{array}$$

This notation denotes the substitution of variables $\mathbf{x'}_k$ by $\mathbf{x}_k$, of channels $\mathbf{h'}_m$ by $\mathbf{h}_m$, and of variable $w$ by expression $\mathbf{e}_n$. The substitution takes place on the initialization predicate $i \wedge (\mathbf{v}_n = w)$, on the recursion definitions $\mathbf{X} \mapsto \mathbf{p}$ and on the process term $p_{\mathrm{body}}$.

The variable $w$ is assumed to be fresh with respect to $\mathbf{x'}_k$, $\mathbf{v}_n$, $\mathbf{z}_d$, $\mathbf{z}_c$, $\mathbf{z}_a$. The substitution is defined in such a way that no variables from $\mathbf{x}_k$ or $\mathbf{e}_n$, and no channels from $\mathbf{h}_m$ become bound. If substitution would cause new bindings, the local variable or local channel that a variable or channel from $\mathbf{x}_k$, $\mathbf{e}_n$, or $\mathbf{h}_m$ would become bound to, is renamed into a fresh variable or fresh channel before the substitution takes place.

The translation declares the value parameters $\mathbf{v}_n$ as local discrete variables with initial values $\mathbf{e}_n$. By convention, however, process term $p_{\text{body}}$ normally does not change the values of these variables.

## 2.4 Data types

The $\chi$ language is statically strongly typed. Besides the classification of variables as defined before, all variables have a type. The type of a variable defines the allowed values of the variable and the allowed operations on the variable. The atomic types are nat (natural numbers, including zero), int (integers), real (real-valued numbers), bool (booleans), string (strings), and enum (enumerations). Type constructors operate on existing types to create structured types. The $\chi$ language defines type constructors to create sets, lists, array tuples, record tuples, dictionaries, functions, and distributions (for stochastic models). Channels also have a type that indicates the type of data that is communicated via the channel. Pure synchronization channels, that do not communicate data, are of the predefined type void. The $\chi$ type system is strictly enforced in the $\chi$ tools. However, since the type system is not formalized, it is omitted from the specifications in this paper.

# Chapter 3

# Semantics of the Chi language

This section presents the structured operational semantics (SOS [39]) of hybrid $\chi$. It associates a hybrid transition system [18] with a $\chi$ process. The semantics is defined only for a subset of the syntactically allowed $\chi$ processes. E.g. the semantics of the $\chi$ process $\langle x \geq 1 \rightarrow p, \sigma, E \rangle$ is defined only for variables $x$ that have a defined value. These additional semantical restrictions on $\chi$ processes, if present, are specified together with the SOS rules for each process term in Sections 3.3 and 3.4.

## 3.1 General description of the SOS

The main purpose of such an SOS is to define the behavior of hybrid $\chi$ processes at a certain chosen level of abstraction. The meaning of a $\chi$ process depends on the values of the variables and on the environment. A set $V$ of variables, and a set $H$ of channel labels are assumed. The values of the variables at a specific moment in time are captured by means of a valuation, i.e., a partial function from the variables to the set of values $\Lambda$ (containing at least the booleans $\mathbb{B}$ and the reals $\mathbb{R}$). The set of all valuations is denoted $\Sigma$: $\Sigma = V \mapsto \Lambda$, and we assume $\sigma \in \Sigma$ and $\mathsf{time} \in \mathrm{dom}(\sigma)$ for all $\chi$ processes $\langle p, \sigma, E \rangle$. Extended valuations also include the values of dotted continuous variables and the algebraic variables. The set of all extended valuations is denoted $\dot{\Sigma}$: $\dot{\Sigma} = (V \cup \dot{V}) \mapsto \Lambda$, where $\dot{V}$ denotes the set of all dotted variables. The set $T$ is used to represent points in time; usually $T = \mathbb{R}_{\geq 0}$. The set of environments $ES$ is defined as $ES = \mathcal{P}(V) \times \mathcal{P}(V) \times \mathcal{P}(V) \times RS$, where $RS = XS \mapsto P$ denotes the set of all partial functions of recursion variables $XS$ to process terms $P$.

The SOS is chosen to represent the following:

1. discrete behavior by means of action transitions:

   (a) $\_ \xrightarrow{\cdot} \_ \subseteq (P \times \Sigma \times ES) \times (\dot{\Sigma} \times A \times \dot{\Sigma}) \times (P \times \Sigma \times ES)$, where $A$ denotes the set of actions, and is defined as $A = A_{\mathrm{label}} \cup A_{\mathrm{com}}$. The set of action labels $A_{\mathrm{label}}$ includes

at least the pre-defined internal action $\tau$. The set of communication actions $A_{\text{com}}$ is defined as $A_{\text{com}} = \{\text{isa}(h, cs), \text{ira}(h, cs, W), \text{ca}(h, cs) \mid h \in H, cs \in \Lambda^*, W \subseteq V\}$, where isa, ira, and ca denote action labels for the internal send action, the internal receive action, and the communication action respectively, $h \in H$ denotes a channel, $cs \in \Lambda^*$ denotes a list $[c_1, \ldots, c_n]$ of values, and $W$ denotes a set of variables. The intuition of an action transition $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$ is that the process $\langle p, \sigma, E\rangle$ executes the discrete action $a \in A$ with extended valuations $\xi$ and $\xi'$ and thereby transforms into the process $\langle p', \sigma', E'\rangle$, where $\sigma'$ and $E'$ denote the accompanying valuation and environment of the process term $p'$, respectively, after the discrete action $a$ is executed.

(b) $\_ \xrightarrow{\_} \langle\surd, \_, \_\rangle \subseteq (P \times \Sigma \times ES) \times (\dot{\Sigma} \times A \times \dot{\Sigma}) \times (\Sigma \times ES)$. The intuition of a (termination) transition $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma', E'\rangle$ is that the process $\langle p, \sigma, E\rangle$ executes the discrete action $a$ with extended valuations $\xi$ and $\xi'$ and thereby transforms into the terminated process $\langle\surd, \sigma', E'\rangle$.

2. continuous behavior by means of time transitions: $\_ \overset{\_}{\longmapsto} \_ \subseteq (P \times \Sigma \times ES) \times (T \times (T \mapsto \dot{\Sigma})) \times (P \times \Sigma \times ES)$. The intuition of a time transition $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$ is that during the time transition, the extended valuation at each time-point $s \in [0, t]$ is given by $\rho(s)$. At the end-point $t$, the resulting process is $\langle p', \sigma', E'\rangle$.

3. consistency by means of a unary relation: $\_ \overset{\_}{\rightsquigarrow} \subseteq (P \times \Sigma \times ES) \times \dot{\Sigma}$. The intuition of a consistency transition $\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$ is that the extended valuation $\xi$ is consistent with process $\langle p, \sigma, E\rangle$, and is thus also consistent with delay predicates in $p$.

In this paper, for all transitions, the domain of the valuation $\sigma$ equals the domain of valuation $\sigma'$, environment $E$ equals environment $E'$.

For all action transitions $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma', E'\rangle$ and $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$, the domain of $\sigma$ is included in the domain of $\xi$, $\text{dom}(\sigma) \subseteq \text{dom}(\xi)$, and $\text{dom}(\xi) = \text{dom}(\xi')$. Extended valuation $\xi$ restricted to $\text{dom}(\sigma)$ equals valuation $\sigma$, and extended $\xi'$ restricted to $\text{dom}(\sigma')$ equals valuation $\sigma'$.

For all time transitions $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p, \sigma', E'\rangle$, $\text{dom}(\rho) = [0, t]$, and for all variables $x \in \text{dom}(\sigma)$, the value in the resulting valuation $\sigma'(x)$ equals the value of the variable in the end-point of the trajectory $\rho(t)(x)$. These properties of the semantics can be found in Section 6.

The relations and predicates mentioned above are defined through so-called deduction rules. A deduction rule is of the form $\frac{H}{r}$, where $H$ is a number of hypotheses separated by commas and $r$ is the result of the rule. The result of a deduction rule can be derived if all of its hypotheses are derived. In case the set of hypotheses is empty, the deduction rule is called an axiom.

In order to increase the readability of the $\chi$ deduction rules, some additional abbreviations are used. Notation $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$, where $q \in P \cup \{\surd\}$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma', E\rangle$, notation $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle q, \sigma'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q, \sigma', E\rangle$, and notation $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ is an abbreviation for $\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

Notation $E \Vdash f_1, \ldots, f_n$, where $f_i$ represents one of the previously defined transition relations (of the forms $\langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$ or $\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle q, \sigma'\rangle$ or $\langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$) is an abbreviation for $E \Vdash f_1, \ldots, E \Vdash f_n$.

Notation

$$\frac{E' \Vdash \langle p_1, \sigma_1\rangle \xrightarrow{\xi_1,a_1,\xi_1'} \left\langle \begin{matrix} q_{11} \\ \vdots \\ q_{1n} \end{matrix}, \sigma_1' \right\rangle, \ldots, \langle p_m, \sigma_m\rangle \xrightarrow{\xi_m,a_m,\xi_m'} \left\langle \begin{matrix} q_{m1} \\ \vdots \\ q_{mn} \end{matrix}, \sigma_m' \right\rangle,\ C}{E \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,b,\xi'} \left\langle \begin{matrix} s_1 \\ \vdots \\ s_n \end{matrix}, \sigma' \right\rangle}$$

where $q_{ji}, s_i \in P \cup \{\checkmark\}$, $p_i, r \in P$, and $C$ denotes an optional hypothesis that must be satisfied in the deduction rule, is an abbreviation for the following rules (one for each $i$):

$$\frac{E' \Vdash \langle p_1, \sigma_1\rangle \xrightarrow{\xi_1,a_1,\xi_1'} \langle q_{1i}, \sigma_1'\rangle, \ldots, \langle p_m, \sigma_m\rangle \xrightarrow{\xi_m,a_m,\xi_m'} \langle q_{mi}, \sigma_m'\rangle,\ C}{E \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,b,\xi'} \langle s_i, \sigma'\rangle}$$

The notation $\frac{H}{R}$, where $R$ is a number of results separated by commas, is an abbreviation for a set of deduction rules of the form $\frac{H}{r}$; one for each $r \in R$, and notation $E \ \frac{H}{r}$ is an abbreviation for $\frac{E \Vdash H}{E \Vdash r}$.

Furthermore, notation $\langle p, \sigma, E\rangle \overset{\mathrm{ca}(h,*)}{\nrightarrow}$ denotes $(\nexists_{\xi,cs,\xi',p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p', \sigma', E'\rangle) \wedge (\nexists_{\xi,cs,\xi',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle\checkmark, \sigma', E'\rangle)$, and notation $\langle p, \sigma, E\rangle \xrightarrow{\alpha} \langle p', \sigma', E'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$ for some $\xi$, $a$, and $\xi'$.

## 3.2 Notations and mathematical definitions

Notations $f \in M \to G$ and $g \in M \mapsto G$ define complete function $f$, $\mathrm{dom}(f) = M$, and partial function $g$, $\mathrm{dom}(g) \subseteq M$, both with range $G$.

### 3.2.1 Operators on functions

Based on [32], the following definitions of operators $\restriction$, $\cup$, and $\downarrow$ applied on functions are used. If $f$ is a function, $\mathrm{dom}(f)$ and $\mathrm{range}(f)$ denote the domain and range of $f$, respectively. If $S$ is a set, $f \restriction S$ denotes the restriction of $f$ to $S$, that is, the function $g$ with $\mathrm{dom}(g) = \mathrm{dom}(f) \cap S$, such that $g(c) = f(c)$ for each $c \in \mathrm{dom}(g)$.

If $f$ and $g$ are functions with $\mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset$, then $f \cup g$ denotes the unique function $h$ with $\mathrm{dom}(h) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ satisfying the condition: for each $c \in \mathrm{dom}(h)$, if $c \in \mathrm{dom}(f)$ then $h(c) = f(c)$, and $h(c) = g(c)$ otherwise.

If $f$ is a function whose range is a set of functions and $S$ is a set, then $f \downarrow S$ denotes the function $g$ with $\mathrm{dom}(g) = \mathrm{dom}(f)$ such that $g(c) = f(c) \upharpoonright S$ for each $c \in \mathrm{dom}(g)$. If $f$ is a function whose range is a set of functions, all of which have a particular element $d$ in their domain, then $f \downarrow d$ denotes the function $g$ with $\mathrm{dom}(g) = \mathrm{dom}(f)$ such that $g(c) = f(c)(d)$ for each $c \in \mathrm{dom}(g)$.

If $f$ and $g$ are functions, then $f \prec g$ denotes the function $h$ with $\mathrm{dom}(h) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ such that

$$h(x) = \begin{cases} g(x) & x \in \mathrm{dom}(g) \\ f(x) & x \in \mathrm{dom}(f) \setminus \mathrm{dom}(g) \end{cases}$$

Thus $f \prec g$ merges the functions $f$ and $g$ such that $g$ overrides $f$.

If $f$ and $g$ are functions with the same domain, whose ranges are sets of functions, and $\mathrm{dom}(\mathrm{range}(f)) \cap \mathrm{dom}(\mathrm{range}(g)) = \emptyset$, then $f \dot{\cup} g$ denotes the unique function $h$ with $\mathrm{dom}(h) = \mathrm{dom}(f)$ satisfying the condition $h(c) = f(c) \cup g(c)$ for each $c \in \mathrm{dom}(h)$.

### 3.2.2 Notations

Let $x \in V$ be a variable, $S, C, L \subseteq V$ be sets of variables, $\sigma \in \Sigma$ be a valuation, $e$ be an expression over variables and constants, and $t \in T$ be a time-point, then the following notations are defined:

- $\sigma(x)$ denotes the value of variable $x$ in valuation $\sigma$. We use the similar notation $\sigma(e)$ * to denote the value of expression $e$ for valuation $\sigma$.
- $\dot{S}$ denotes the set of dotted variables $\{\dot{x} \mid x \in S\}$.
- $\xi^{\dot{C}L} \in (\dot{C} \cup L) \to \Lambda$ denotes an arbitrary valuation with domain $\dot{C} \cup L$.
- $\xi_\sigma$ is an abbreviation for $\xi \upharpoonright \mathrm{dom}(\sigma)$.
- Function $\Xi \in (\Sigma \times \mathcal{P}(V) \times \mathcal{P}(V) \times \mathcal{P}(V)) \to \mathcal{P}(\dot{\Sigma})$ returns a set of extended valuations, given a valuation, the set of continuous variables, the set of jumping variables, and the set of algebraic variables. Formally, function $\Xi$ is defined as:

  $$\Xi(\sigma, C, J, L) = \{\xi \mid \mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \cup \dot{C} \cup L,\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\ \xi(x) = \sigma(x)\}.$$

  The domain of the extended valuations is given by $\mathrm{dom}(\sigma) \cup \dot{C} \cup L$. The values of the variables in $\mathrm{dom}(\sigma) \setminus J$ are given by $\sigma$. The jumping variables $J$, the dotted variables $\dot{C}$ and the algebraic variables $L$ are allowed to change arbitrarily.

*In previous papers on the semantics of $\chi$, $\bar{\sigma}(e)$ was used instead of $\sigma(e)$

- $\Omega_{\sigma E t}$, where environment $E$ denotes the tuple $(C, J, L, R)$, is an abbreviation for $\Omega(\sigma, C, L, \text{true}, t)$. Here, $\Omega$ is the solution function as defined in Section 3.3.2.
- $\rho_\sigma$ is an abbreviation for $\rho \downarrow \text{dom}(\sigma)$.

## 3.3 Deduction rules for atomic process terms

### 3.3.1 Action predicate

Action predicate process term $W : r \gg l_\text{a}$ denotes instantaneous changes to the variables from set $W \subseteq (\text{dom}(\sigma) \setminus \{\mathsf{time}\}) \cup L$, by means of an action labeled $l_\text{a} \in A_\text{label}$, such that predicate $r$ over variables from extended valuations $\text{dom}(\xi^-)$ and $\text{dom}(\xi')$ is satisfied, see Rule 1, where $\xi, \xi' \in (\text{dom}(\sigma) \cup \dot{C} \cup L) \to \Lambda$.

The values of the variables from $\text{dom}(\sigma)$ in $\xi$ are given by $\sigma$. The dotted variables $\dot{C}$ and the algebraic variables $L$ in $\xi$ can in principle take any value ($\xi = \sigma \cup \xi^{\dot{C}L}$) as long as the action predicate $r$ is satisfied ($\xi^- \cup \xi' \models r$). Variables occurring with a '$^-$' superscript in $r$ are evaluated in $\xi^-$, which denotes an extended valuation with $\text{dom}(\xi^-) = \{x^- \mid x \in \text{dom}(\xi)\}$, and $\xi^-(x^-) = \xi(x)$. For extended valuation $\xi'$, the values of the discrete and the non-jumping variables ($\text{dom}(\sigma) \setminus J \cup W$) are given by $\sigma$. The jumping variables $J$, the variables from set $W$, the dotted variables $\dot{C}$ and the algebraic variables $L$ are allowed to change such that the action predicate is satisfied.

Rule 2 states that an action predicate is consistent with extended valuation $\sigma \cup \xi^{\dot{C}L}$.

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J \cup W, L),\ \xi^- \cup \xi' \models r}{(C, J, L, R) \Vdash \langle W : r \gg l_\text{a}, \sigma \rangle \xrightarrow{\xi,\ l_\text{a},\ \xi'} \langle \checkmark, \xi'_\sigma \rangle}\ 1$$

$$\frac{}{(C, J, L, R) \Vdash \langle W : r \gg l_\text{a}, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 2$$

### 3.3.2 Delay predicate

Delay predicate $u$ is a predicate over variables from $\text{dom}(\sigma) \cup \dot{C} \cup L$.

$$\frac{\rho \in \Omega(\sigma, C, L, u, t)}{(C, J, L, R) \Vdash \langle u, \sigma \rangle \xrightarrow{t,\rho} \langle u, \rho_\sigma(t) \rangle}\ 3$$

$$\frac{\sigma \cup \xi^{\dot{C}L} \models u}{(C, J, L, R) \Vdash \langle u, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 4$$

Function $\Omega \in \Sigma \times \mathcal{P}(V) \times \mathcal{P}(V) \times U \times T \to \mathcal{P}(T \mapsto \dot{\Sigma})$, where $U$ denotes the set of all predicates over $V$ and $\dot{V}$, returns a set of trajectories from time to an extended valuation for the variables and dotted variables, given a valuation representing the current values of the discrete and continuous variables, the set of continuous variables, the set of algebraic variables, a delay predicate and a time point that denotes the duration of the trajectory. Formally, function $\Omega$ is defined as:

$$
\begin{array}{ll}
\Omega(\sigma, C, L, u, t) = & \\
\{\ \rho & \\
\mid\ \rho \in [0,t] \mapsto ((\mathrm{dom}(\sigma) \cup \dot{C} \cup L) \to \Lambda) & \\
,\ t \geq 0 & \\
,\ \forall s \in [0,t] : & \rho(s) \models u \\
,\ \forall x \in \dot{C} \cup L : & \rho \downarrow x \text{ is a bounded function that is absolutely} \\
 & \text{continuous except for a finite number of points.} \\
,\ \forall x \in \mathrm{dom}(\sigma) \setminus (\{\mathsf{time}\} \cup C) : & \rho \downarrow x \text{ is a constant function.} \\
,\ \forall x \in C : & \rho \downarrow x \text{ is an absolutely continuous function.} \\
,\ \forall x \in \mathrm{dom}(\sigma) : & (\rho \downarrow x)(0) = \sigma(x) \\
,\ \forall s \in [0,t], x \in C : & \left\{ \begin{array}{l} (\rho \downarrow x)(s) = (\rho \downarrow x)(0) + \int_0^s (\rho \downarrow \dot{x})(s')ds' \\ \rho \downarrow x \text{ is differentiable in } s \Rightarrow \\ \quad (\rho \downarrow \dot{x})(s) = (\frac{d}{dt}(\rho \downarrow x))(s) \end{array} \right. \\
,\ \forall s \in [0,t] : & \rho(s)(\mathsf{time}) = \sigma(\mathsf{time}) + s \\
\} &
\end{array}
$$

Trajectory $\rho$ satisfies the predicate $u$ for all time points of its domain ($\forall_{s \in [0,t]}\ \rho(s) \models u$). The trajectory $\rho \downarrow x$ of each dotted variable (from set $\dot{C}$) and of each algebraic variable (from set $L$) is restricted to a bounded function that is absolutely continuous except for a finite number of points. The trajectory of each continuous variable is absolutely continuous. The trajectory of each discrete variable $x \in \mathrm{dom}(\sigma) \setminus (\{\mathsf{time}\} \cup C)$ is restricted to a constant function. The initial value (starting-point) of the trajectory of each discrete and continuous variable, equals its value in $\sigma$ ($\forall_{x \in \mathrm{dom}(\sigma)}\ (\rho \downarrow x)(0) = \sigma(x)$).

The relation between the trajectory of a continuous variable $x$ and the trajectory of its 'derivative' $\dot{x}$ is given by the Caratheodory solution concept [23] $(\rho \downarrow x)(s) = (\rho \downarrow x)(0) + \int_0^s (\rho \downarrow \dot{x})(s')ds'$. This allows a non-smooth (but continuous) trajectory for a differential variable in the case that the trajectory of its 'derivative' is non-smooth or even discontinuous, as in, for example, $\langle\ \mathsf{cont}\ y,\ y = 0 \mid \dot{y} = \mathrm{step}(\mathsf{time} - 1)\ \rangle$, where $\mathrm{step}(x)$ equals 0 for $x \leq 0$ and 1 for $x > 0$.

The disadvantage of the Caratheodory solution concept is that it introduces spurious solutions as in, for example, $\langle\ \mathsf{cont}\ y,\ \mathsf{alg}\ z \mid y = \mathsf{time}, z = \dot{y}\ \rangle$. Here, the trajectory for $z$ should be the constant function 1. The Caratheodory solution concept, however, allows trajectories for $z$ that are 1 almost everywhere, except for discontinuity points where any other value is allowed. To prevent such spurious discontinuities in the case that the trajectory of a differential variable $x$ is smooth, and thus $\rho \downarrow x$ is differentiable in $s$, the trajectory of its derivative $\rho \downarrow \dot{x}$ should indeed

be the derivative function of the trajectory of the differential variable $x$: $(\rho \downarrow \dot{x})(s) = (\frac{d}{dt}(\rho \downarrow x))(s)$.

In some SOS rules describing delay behavior, abbreviation $\Omega_{\sigma E t}$, which denotes $\Omega(\sigma, C, L, \text{true}, t)$, is used as a hypothesis. The true predicate does not restrict $t$ and the trajectory $\rho$ other than by means of the default restrictions. Among others, the discrete variables remain constant, and the trajectory of each continuous variable is a bounded, absolutely continuous function that starts in the value of the continuous variable in $\sigma$.

### 3.3.3 Send and receive

Send and receive process terms $h \,!!\, \mathbf{e}_n$ and $h \,??\, \mathbf{x}_n$ denote undelayable sending of expression $\mathbf{e}_n$ via channel $h$, and undelayable receiving of information via channel $h$ into variable(s) $\mathbf{x}_n$, respectively.

The values of expressions $e_1, \ldots, e_n$ which are sent via channel $h$ are evaluated in extended valuation $\xi$, see Rule 5, where $\mathbf{e}_n$ denotes $e_1, \ldots, e_n$, $[\xi(\mathbf{e}_n)]$ denotes the list of values $[\xi(e_1), \ldots, \xi(e_n)]$ for $n \geq 1$, and $\xi(e)$ denotes the value of expression $e$ for extended valuation $\xi$. The case that $n$ equals 0, represents the case where nothing is sent via the channel, and $\mathbf{e}_0$ and $[\xi(\mathbf{e}_0)]$ denote an empty expression and an empty list, respectively. For $n \geq 1$, the receive process term $h \,??\, x_1, \ldots, x_n$ can receive the list of values $[c_1, \ldots, c_n]$, see Rule 6, where $\mathbf{x}_n$ denotes $x_1, \ldots, x_n$, $\{\mathbf{x}_n\}$ denotes the set $\{x_1, \ldots, x_n\}$ ($\{\mathbf{x}_n\} \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\} \cup L$), $[\mathbf{c}_n]$ denotes the list of values $[c_1, \ldots, c_n]$, and $\xi'(\mathbf{x}_n) = \mathbf{c}_n$ is an abbreviation for $\xi'(x_1) = c_1, \ldots, \xi'(x_n) = c_n$. We assume that all variables in $\mathbf{x}_n$ are different: $x_i = x_j \implies i = j$. For $n = 0$, nothing is received, so that $\mathbf{x}_0$ and $\mathbf{c}_0$ are empty, and $\xi'(\mathbf{x}_0) = \mathbf{c}_0$ always holds. Furthermore, we assume $\{\mathbf{x}_n\} \subseteq (\mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}) \cup L$.

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J, L)}{(C, J, L, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \xrightarrow{\xi,\ \mathrm{isa}(h, [\xi(\mathbf{e}_n)]),\ \xi'} \langle \surd, \xi'_\sigma \rangle} 5$$

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L),\ \xi'(\mathbf{x}_n) = \mathbf{c}_n}{(C, J, L, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \xrightarrow{\xi,\ \mathrm{ira}(h, [\mathbf{c}_n], \{\mathbf{x}_n\}),\ \xi'} \langle \surd, \xi'_\sigma \rangle} 6$$

$$\frac{}{(C, J, L, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} 7$$

$$\frac{}{(C, J, L, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} 8$$

### 3.3.4 Consistent deadlock

Process term $\delta$ cannot perform any action transitions, nor time transitions. It is, however, consistent with extended valuation $\sigma \cup \xi^{\dot{C}L}$. This means that process $\langle\ \delta, \sigma, E\ \rangle$ corresponds with a hybrid automaton consisting of one location with flow predicate false, invariant true and no outgoing edges.

$$\frac{}{(C,\ J,\ L,\ R) \Vdash \langle \delta, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 9$$

### 3.3.5 Inconsistent process term

The inconsistent process term $\perp$ is considered to be in an inconsistent state from its start. An example is the delay predicate false, for which there is no extended valuation in which this predicate holds. It corresponds with a hybrid automaton consisting of one location with invariant false. Like process term $\delta$, process term $\perp$ cannot perform any action transitions, nor time transitions. Process term $\perp$ originates from the process algebra with propositional signals $\mathrm{ACP}_{\mathrm{ps}}$ ([7]).

## 3.4 Deduction rules for operators

### 3.4.1 Delay enabling operator

By means of the delay enabling operator $[p]$, time transitions of arbitrary duration are allowed for the behavior of $p$ (see Rule 11). Time transitions of $p$ itself are neglected. The delay enabling operator does not affect the action behavior of $p$ (see Rule 10). It is consistent with extended valuation $\sigma \cup \xi^{\dot{C}L}$ (see Rule 12).

$$E\ \frac{\langle p, \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{\langle [p], \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle}\ 10$$

$$E\ \frac{\rho \in \Omega_{\sigma E t}}{\langle [p], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [p], \rho_\sigma(t) \rangle}\ 11 \qquad \frac{}{(C,\ J,\ L,\ R) \Vdash \langle [p], \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 12$$

### 3.4.2 Signal emission operator

The signal emission operator $u \curvearrowright p$ ensures that $p$ starts its behavior from an extended valuation $\xi$ in which initialization predicate $u$ is satisfied. This operator was inspired by the signal emission operator from the process algebra with propositional signals $\mathrm{ACP}_{\mathrm{ps}}$ [7], which was also used in [11].

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix},\sigma'\rangle,\ \xi \models u}{\langle u \curvearrowright p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix},\sigma'\rangle}\ 13$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\ \rho(0) \models u}{\langle\ u \curvearrowright p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}\ 14$$

$$E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow},\ \xi \models u}{\langle\ u \curvearrowright p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 15$$

### 3.4.3 Sequential composition operator

The sequential composition of process terms $p$ and $q$ behaves as process term $p$ until $p$ terminates, and then continues to behave as process term $q$. When $p$ terminates, its right-hand extended valuation $\xi'$ must be consistent with $q$ (see Rule 16).

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark,\sigma'\rangle,\ \langle q,\sigma'\rangle \overset{\xi'}{\rightsquigarrow}}{\langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q,\sigma'\rangle}\ 16 \quad E\ \frac{\langle p,\sigma\rangle \xrightarrow{\alpha} \langle p',\sigma'\rangle}{\langle p;\ q,\sigma\rangle \xrightarrow{\alpha} \langle p';\ q,\sigma'\rangle}\ 17$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}{\langle p;\ q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p';\ q,\sigma'\rangle}\ 18 \quad E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p;\ q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 19$$

### 3.4.4 Guard operator

The guarded process term $b \to p$ can do whatever actions $p$ can do under the condition that the guard evaluates to true using extended valuation $\xi$ (we assume $b$ to be defined in $\xi$ in Rule 20). Evaluating the guard in $\xi$ ensures that when guard operators are nested with signal emission operators, actions can be executed only if all initialization predicates and all guards

hold, independently of the order. Furthermore, the values of the dotted variables and algebraic variables are defined in $\xi$, whereas they are not defined in $\sigma$.

The guarded process term can delay according to $p$ under the condition that for all intermediate valuations the guard evaluates to true ($\forall_{s\in[0,t]}\ \rho(s) \models b$).

The guarded process term can perform arbitrary delays under the condition that for the intermediate valuations, possibly excluding the first and last valuation, the guard does not hold ( $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$). This ensures that, for example, the process $\langle\ \mathsf{disc}\ x,\ x = 1 \mid \mathsf{time} \geq x \rightarrow \mathrm{skip}\ \rangle$ behaves as expected: it can first do a time transition of 1, such that the value of the current time time becomes 1, and thereafter it can do a $\tau$ action to the terminated process. If the condition in Rule 22 would be $\forall_{s\in[0,t]}\ \rho(s) \models \neg b$, then a time transition of 1 would be impossible. This is because the value of the guard should also be false for the last time point of the time transition, so that the point where the value of time equals 1 could not be reached. The condition $\rho(0) \models b \Rightarrow \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma'\rangle$ in Rule 22 which states that $p$ must be able to delay for a duration of 0 if the guard is initially true, ensures that undelayable actions in $p$ have priority over delay behavior of a guard that is initially true and continues as false. The condition $\rho(t) \models b \Rightarrow \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ in Rule 22 states that $p$ must be consistent if the guard holds in the end-point of the trajectory. This ensures that it is impossible to delay to an inconsistent state.

Rule 23 states that the guarded process term is consistent if $p$ is consistent with extended valuation $\xi$, and $b$ holds in this extended valuation. Rule 24 states that the guarded process term is consistent with an extended valuation in which the guard does not hold.

$$E\ \frac{\langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \xi \models b}{\langle b \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle}\ 20$$

$$E\ \frac{\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle,\ \forall_{s\in[0,t]}\ \rho(s) \models b}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p', \sigma'\rangle}\ 21$$

$$E\ \frac{\begin{array}{c}\rho \in \Omega_{\sigma E t},\ \forall_{s\in(0,t)}\ \rho(s) \models \neg b,\\ \rho(0) \models b \Rightarrow \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma'\rangle\\ \rho(t) \models b \Rightarrow \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}\end{array}}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p, \rho_\sigma(t)\rangle}\ 22$$

$$E\ \frac{\langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow},\ \xi \models b}{\langle b \rightarrow p, \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 23$$

$$\frac{\sigma \cup \xi^{\dot{C}L} \models \neg b}{(C, J, L, R) \Vdash \langle b \rightarrow p, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} \; 24$$

### 3.4.5 Alternative composition operator

Applying the alternative composition operator to process terms $p$ and $q$ models a non-deterministic choice between $p$ and $q$ for action transitions. Process term $p$ can perform action transitions only if the initial extended valuation is consistent with $q$, as specified in Rule 25. Consider for example the following process term: $y = 1 \; [] \, x := y$. This corresponds to a hybrid automaton with one location with flow predicate true, invariant $y = 1$, and an outgoing edge with jump condition $x := y$. The invariant $y = 1$ ensures that the value of $y$ equals 1 when the outgoing edge is taken.

The passage of time by itself cannot result in making a choice (see Rule 26). This is called strong time-determinism, as defined in [37]. Consider for example the $\chi$ process $\langle (h \,!; \; p \; [] \; \Delta 10; \; q) \parallel r, \sigma, E \rangle$ for some $p, q, r \in P, \sigma \in \Sigma$ and $E \in ES$. The alternative composition specifies a send process term with a time-out of 10 time units. Depending on $r$, either the send succeeds first, followed by $p$, or the time-out succeeds first, followed by $q$, regardless of $q$. This is different from, for example, the $\chi_\sigma$ process algebra [13]. There, $\langle h \,! \; [] \; (\Delta 10; \; \Delta 1), \sigma \rangle$ does not make a choice after expiration of $\Delta 10$.

Rule 27 states that an alternative composition is consistent with an extended valuation if both alternatives are consistent with that valuation.

$$E \;\; \frac{\langle p, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}, \sigma' \rangle, \; \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle p \; [] \, q, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}, \sigma' \rangle, \; \langle q \; [] \, p, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix} \checkmark \\ p' \end{smallmatrix}, \sigma' \rangle} \; 25$$

$$E \;\; \frac{\langle p, \sigma \rangle \overset{t, \rho}{\longmapsto} \langle p', \sigma' \rangle, \; \langle q, \sigma \rangle \overset{t, \rho}{\longmapsto} \langle q', \sigma' \rangle}{\langle p \; [] \, q, \sigma \rangle \overset{t, \rho}{\longmapsto} \langle p' \; [] \, q', \sigma' \rangle} \; 26$$

$$E \;\; \frac{\langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}, \; \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle p \; [] \, q, \sigma \rangle \overset{\xi}{\rightsquigarrow}} \; 27$$

### 3.4.6 Parallel composition operator

The parallel composition of process terms $p$ and $q$ has as its behavior with respect to action transitions the interleaving of the behaviors of $p$ and $q$ (see Rule 29). Process term $p$ can only

perform action transitions from an extended valuation $\xi$ which is consistent with $q$. Furthermore, the resulting extended valuation $\xi'$ must be consistent with $q$ (see Rule 29).

The parallel composition allows the synchronization of matching send and receive actions. A send action $\mathrm{isa}(h, cs)$ and a receive action $\mathrm{ira}(h', cs', W)$ match iff $h = h'$ and $cs = cs'$; i.e. the channels used for sending and receiving are the same, and also the values sent and the values received are identical. Furthermore, the resulting extended valuations $\xi'$ of both the send action and the receive action have to be the same. In order to be able to receive values in variables of the same scope as the send process term, the variables of which the value changes due to the receive action are passed on to the send process term. This is achieved by means of set $W$ on the receive action, and the addition of this set $W$ to the set of jumping variables in the environment where the send action takes place (see Rule 28). The result of the synchronization is a communication action that is represented by $\mathrm{ca}(h, cs)$ as defined by Rule 28.

The time transitions of the process terms that are put in parallel have to synchronize to obtain the time transition (with the same time step $t$ and trajectory $\rho$) of their parallel composition as defined by Rule 30.

A parallel composition of two process terms is consistent with an extended valuation if both process terms are consistent with that extended valuation (see Rule 31).

$$\frac{(C,\, J \cup W,\, L,\, R) \Vdash \langle p, \sigma \rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \left\langle \begin{matrix} \checkmark \\ p' \\ \checkmark \\ p' \end{matrix}, \sigma' \right\rangle, \quad (C,\, J,\, L,\, R) \Vdash \langle q, \sigma \rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \left\langle \begin{matrix} \checkmark \\ \checkmark \\ q' \\ q' \end{matrix}, \sigma' \right\rangle}{(C,\, J,\, L,\, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \left\langle \begin{matrix} \checkmark \\ p' \\ q' \\ p' \parallel q' \end{matrix}, \sigma' \right\rangle, \quad \langle q \parallel p, \sigma \rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \left\langle \begin{matrix} \checkmark \\ p' \\ q' \\ q' \parallel p' \end{matrix}, \sigma' \right\rangle} \; 28$$

$$E \; \frac{\langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}, \; \langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} \checkmark \\ p' \end{matrix}, \sigma' \rangle, \; \langle q, \sigma' \rangle \overset{\xi'}{\rightsquigarrow}}{\langle p \parallel q, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} q \\ p' \parallel q \end{matrix}, \sigma' \rangle, \; \langle q \parallel p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix} q \\ q \parallel p' \end{matrix}, \sigma' \rangle} \; 29$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\ \langle q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle q',\sigma'\rangle}{\langle p \parallel q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p' \parallel q',\sigma'\rangle}\ 30$$

$$E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow},\ \langle q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p \parallel q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 31$$

### 3.4.7 Action encapsulation operator

The behavior of the action encapsulation applied to a process term $\partial_{\mathcal{A}}(p)$ is the same as the behavior of its argument with the restriction that actions from the set $\mathcal{A}$ ($\mathcal{A} \subseteq A \setminus \{\tau\}$) cannot be executed (see Rule 32). Action encapsulation has no effect on time transitions and consistency, as defined by Rules 33 and 34.

$$E\ \frac{\langle p,\sigma\rangle \overset{\xi,a,\xi}{\longrightarrow} \langle \overset{\checkmark}{p'},\sigma'\rangle,\ a \notin \mathcal{A}}{\langle \partial_{\mathcal{A}}(p),\sigma\rangle \overset{\xi,a,\xi}{\longrightarrow} \langle \overset{\checkmark}{\partial_{\mathcal{A}}(p')},\sigma'\rangle}\ 32$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}{\langle \partial_{\mathcal{A}}(p),\sigma\rangle \overset{t,\rho}{\longmapsto} \langle \partial_{\mathcal{A}}(p'),\sigma'\rangle}\ 33 \qquad E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle \partial_{\mathcal{A}}(p),\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 34$$

### 3.4.8 Urgent communication operator

The urgent communication operator $\upsilon_{\mathcal{H}}(p)$ gives communication actions via channels from set $\mathcal{H} \subseteq H$ a higher priority than time transitions. Action behavior and consistency are not affected by the urgent communication operator, see Rules 35 and 37. Time transitions are allowed only if at each intermediate state while delaying no communication actions via channels from $\mathcal{H}$ are possible.

$$E\ \frac{\langle p,\sigma\rangle \overset{\alpha}{\rightarrow} \langle \overset{\checkmark}{p'},\sigma'\rangle}{\langle \upsilon_{\mathcal{H}}(p),\sigma\rangle \overset{\alpha}{\rightarrow} \langle \overset{\checkmark}{\upsilon_{\mathcal{H}}(p')},\sigma'\rangle}\ 35$$

$$E\ \frac{\begin{array}{c}\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\\ \forall_{s\in[0,t)}\ (\langle p,\sigma\rangle \overset{s,\rho}{\longmapsto} \langle p_s,\sigma_s,\rangle,\ \langle p_s,\sigma_s\rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle p',\sigma'\rangle,\ \forall_{h\in\mathcal{H}}\ \langle p_s,\sigma_s,E\rangle \overset{\mathrm{ca}(h,*)}{\not\rightarrow})\end{array}}{\langle \upsilon_{\mathcal{H}}(p),\sigma\rangle \overset{t,\rho}{\longmapsto} \langle \upsilon_{\mathcal{H}}(p'),\sigma'\rangle}\ 36$$

$$E \quad \frac{\langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle \upsilon_{\mathcal{H}}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}} \; 37$$

where $\rho_{-s}$ denotes the trajectory $\rho$ shifted left by $s$ time-units and starting at 0: $\mathrm{dom}(\rho_{-s}) = [0, t-s]$, and $\forall t' \in \mathrm{dom}(\rho_{-s}) : \rho_{-s}(t') = \rho(t'+s)$, where $t$ denotes the end-point of the domain of $\rho$: $\mathrm{dom}(\rho) = [0, t]$.

### 3.4.9 Recursion variable

A recursion variable process term $X$ behaves as the process term given by $R(X)$. Here $R(X)$ is the process term that is defined for recursion variable $X$ in function $R$. This is equivalent to syntactically replacing recursion variable $X$ by its defining process term $R(X)$. It is assumed that $X$ is defined in the environment: $X \in \mathrm{dom}(R)$. Function $R$ can be defined in the environment of the $\chi$ process directly, or by means of the recursion scope operator, see Section 3.4.13.

$$(C,\, J,\, L,\, R) \quad \frac{\langle R(X), \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{\langle X, \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle} \; 38 \qquad (C,\, J,\, L,\, R) \quad \frac{\langle R(X), \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle}{\langle X, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle} \; 39$$

$$(C,\, J,\, L,\, R) \quad \frac{\langle R(X), \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle X, \sigma \rangle \overset{\xi}{\rightsquigarrow}} \; 40$$

### 3.4.10 Jump enabling operator

The jump enabling operator applied to a process term $p$ with set $J^+$ ($\iota_{J^+}(p)$) behaves the same as its argument in an environment where the variables from set $J^+$ become jumping variables.

$$\frac{(C,\, J \cup J^+,\, L,\, R) \Vdash \langle p, \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{(C,\, J,\, L,\, R) \Vdash \langle \iota_{J^+}(p), \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{\iota_{J^+}(p')}, \sigma' \rangle} \; 41$$

$$\frac{(C,\, J \cup J^+,\, L,\, R) \Vdash \langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle}{(C,\, J,\, L,\, R) \Vdash \langle \iota_{J^+}(p), \sigma \rangle \overset{t,\rho}{\longmapsto} \langle \iota_{J^+}(p'), \sigma' \rangle} \; 42$$

$$\frac{(C,\, J \cup J^+,\, L,\, R) \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{(C,\, J,\, L,\, R) \Vdash \langle \iota_{J^+}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}} \; 43$$

### 3.4.11 Variable scope operator

By means of the variable scope operator, local variables are introduced in a $\chi$ process. A variable scope operator process term

$$[\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!],$$

that is used in an environment $(C, J, L, R)$, with valuation $\sigma$, and where $\sigma_{\mathrm{dx}_\perp}$ denotes a local valuation with domain $\{\mathbf{d}, \mathbf{x}\}$, $\mathbf{d}$ denotes the local discrete variables $d_1, \ldots, d_k$, $\mathbf{x}$ denotes the local (non-jumping) continuous variables $x_1, \ldots, x_n$, and $\mathbf{g}$ denotes the local algebraic variables $g_1, \ldots, g_m$, behaves as $p$ after taking the union of the respective categories (discrete, continuous and algebraic) of local and global variables and taking the union of the local and global valuation. To ensure that all local variables are fresh with respect to the global variables, the local variables are first renamed. Thus $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, in the rules below, denote fresh variables $d'_1, \ldots, d'_k$, $x'_1, \ldots, x'_n$, $g'_1, \ldots, g'_m$ with respect to $\mathrm{dom}(\sigma) \cup L \cup \{\mathbf{d}\} \cup \{\mathbf{x}\} \cup \{\mathbf{g}\}$. The local variables $d_1, \ldots, d_k, x_1, \ldots, x_n, g_1, \ldots, g_m$ are assumed to be all different. Notation $p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}]$ denotes the process term that is obtained by substitution of the variables $\mathbf{d}, \mathbf{x}, \mathbf{g}$ in $p$ by $\mathbf{d}', \mathbf{x}', \mathbf{g}'$, respectively. After execution of an action or a delay transition, the local variables of the variable scope operator are renamed back to their original names.

The variable scope operator is the only operator that affects the set of continuous variables $C$ and the set of algebraic variables $L$ from the environment. In this way, it is ensured that the discrete, continuous, or algebraic variables in any $\chi$ process $\langle p, \sigma, E\rangle$ remain discrete, continuous, or algebraic, respectively. Continuous variables, on the other hand, can change from initialized (non-jumping) continuous variables to jumping continuous variables, using the jump enabling operator (see Section 3.4.10).

The local variables are invisible outside of the scope operator. This is done by means of data abstraction. For action transitions, data abstraction takes place by restricting the extended valuations, and the valuation of the resulting process, to the global variables, and by keeping only the global variables in the set $W$ of the internal receive actions. For time transitions, data abstraction takes place by restricting the trajectory to the global variables. In this way, all changes to local variables are removed.

Action transition abstraction function $\kappa \in \dot{\Sigma} \times A \times \dot{\Sigma}$ is defined as follows. For arbitrary receive actions $\mathrm{ira}(h, cs, W)$:

$$\kappa_{\sigma\dot{C}L}(\xi, \mathrm{ira}(h, cs, W), \xi') = \xi_{\sigma\dot{C}L},\ \mathrm{ira}(h, cs, W \cap (\mathrm{dom}(\sigma) \cup L)),\ \xi'_{\sigma\dot{C}L},$$

and for all other actions:

$$\kappa_{\sigma\dot{C}L}(\xi, a, \xi') = \xi_{\sigma\dot{C}L}, a, \xi'_{\sigma\dot{C}L},$$

where extended valuations $\xi_{\sigma\dot{C}L}$ and $\xi'_{\sigma\dot{C}L}$ denote $\xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$, respectively. Furthermore, in the rules below, the following abbreviations are used: valuation $\sigma'_\sigma$ denotes $\sigma' \restriction \mathrm{dom}(\sigma)$, and trajectory $\rho_{\sigma\dot{C}L}$ denotes $\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$.

Valuation $\sigma_{\mathrm{dx}_\perp} \in \{\mathbf{d}, \mathbf{x}\} \mapsto (\Lambda \cup \{\perp\})$ and valuation $\sigma_{\mathrm{d'x'}} \in \{\mathbf{d}', \mathbf{x}'\} \mapsto \Lambda$ define the same values for all (renamed) variables for which $\sigma_{\mathrm{dx}_\perp}$ is defined. For the undefined variables in $\sigma_{\mathrm{dx}_\perp}$, $\sigma_{\mathrm{d'x'}}$

has an arbitrary value: $\forall\, v \in \mathrm{dom}(\sigma_{\mathrm{dx}_\perp}) : \sigma_{\mathrm{dx}_\perp}(v) \neq \perp \Rightarrow \sigma_{\mathrm{d'x'}}(v[\mathbf{d}', \mathbf{x}'/\mathbf{d}, \mathbf{x}]) = \sigma_{\mathrm{dv}_\perp}(v)$, where $v[\mathbf{d}', \mathbf{x}'/\mathbf{d}, \mathbf{x}]$ denotes the renamed version of variable $v$.

$$\frac{(C \cup \{\mathbf{x}'\},\, J,\, L \cup \{\mathbf{g}'\},\, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}} \rangle \xrightarrow{\xi, a, \xi'} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{\begin{array}{c}(C,\, J,\, L\,,\, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!], \sigma \rangle \xrightarrow{\kappa_{\sigma \dot{C} L}(\xi, a, \xi')} \\ \langle \overset{\checkmark}{[\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid p'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}'] \,]\!]}, \sigma'_\sigma \rangle\end{array}} \; 44$$

$$\frac{(C \cup \{\mathbf{x}'\},\, J,\, L \cup \{\mathbf{g}'\},\, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}} \rangle \xmapsto{t, \rho} \langle p', \sigma' \rangle}{\begin{array}{c}(C,\, J,\, L\,,\, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!], \sigma \rangle \xmapsto{t, \rho_{\sigma \dot{C} L}} \\ \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'\})[\mathbf{d}, \mathbf{x}/\mathbf{d}', \mathbf{x}'], \{\mathbf{x}\}, \{\mathbf{g}\} \mid p'[\mathbf{d}, \mathbf{x}, \mathbf{g}/\mathbf{d}', \mathbf{x}', \mathbf{g}'] \,]\!], \sigma'_\sigma \rangle\end{array}} \; 45$$

$$\frac{(C \cup \{\mathbf{x}'\},\, J,\, L \cup \{\mathbf{g}'\},\, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}} \rangle \overset{\xi}{\rightsquigarrow}}{(C,\, J,\, L\,,\, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p \,]\!], \sigma \rangle \overset{\xi_{\sigma \dot{C} L}}{\rightsquigarrow}} \; 46$$

### 3.4.12 Channel scope operator

By means of the channel scope operator, local channels can be introduced in a $\chi$ process. By means of action abstraction, communication actions on local channels are made invisible outside of the scope operator.

Action abstraction takes place by substituting communication actions $\mathrm{ca}(h, cs)$ using a local channel ($h \in H_0$) by internal $\tau$ actions (see Rule 47). The internal send and receive actions ($\mathrm{isa}(h, cs)$ and $\mathrm{ira}(h, cs, W)$) on a local channel $h$ are blocked, because Rule 47 only specifies behavior for communication actions $\mathrm{ca}(h, cs)$. Therefore, internal send and receive actions are not visible outside of the scope operator. Function $\mathrm{ch} \in A \to H \cup \{\perp\}$ extracts the channel label from an action. It is defined as $\mathrm{ch}(\mathrm{ca}(h, cs)) = h$, $\mathrm{ch}(\mathrm{isa}(h, cs)) = h$, $\mathrm{ch}(\mathrm{ira}(h, cs, W)) = h$, and $\mathrm{ch}(l_{\mathrm{a}}) = \perp$, where $l_{\mathrm{a}} \in A_{\mathrm{label}}$.

$$E\ \frac{\langle p, \sigma \rangle \xrightarrow{\xi, \mathrm{ca}(h, cs), \xi'} \langle \overset{\checkmark}{p'}, \sigma' \rangle,\ h \in H_0}{\langle [\![_{\mathrm{H}}\, H_0 \mid p \,]\!], \sigma \rangle \xrightarrow{\xi, \tau, \xi'} \langle \overset{\checkmark}{[\![_{\mathrm{H}}\, H_0 \mid p' \,]\!]}, \sigma' \rangle} \; 47$$

$$E\ \frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix}\checkmark\\ p'\end{matrix},\sigma'\rangle,\ \mathrm{ch}(a)\notin H_0}{\langle [\![_{\mathrm{H}}\ H_0\ \mid\ p\ ]\!],\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix}\checkmark\\ [\![_{\mathrm{H}}\ H_0\ \mid\ p'\ ]\!]\end{matrix},\sigma'\rangle}\ 48$$

$$E\ \frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}{\langle [\![_{\mathrm{H}}\ H_0\ \mid\ p\ ]\!],\sigma\rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{H}}\ H_0\ \mid\ p'\ ]\!],\sigma'\rangle}\ 49$$

$$E\ \frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle [\![_{\mathrm{H}}\ H_0\ \mid\ p\ ]\!],\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 50$$

### 3.4.13 Recursion scope operator

By means of the recursion scope operator, local recursion definitions are introduced in a $\chi$ process. The application of the recursion scope operator to a process term $p$ with a 'global' valuation $\sigma$ and a 'global' environment $(C, J, L, R)$ behaves as $p$ after the addition of local recursion definitions to the global recursion definitions. In the rules below, $\mathbf{X} \mapsto \mathbf{q}$ denotes the recursion definitions $X_1 \mapsto q_1,\ \ldots,\ X_r \mapsto q_r$. To prevent redefinition of recursion definitions already existing in the environment, the local recursion variables are renamed to fresh variables if they are already defined in the environment. In fact, $X'_i = X_i$ $(1 \le i \le r)$ if $X_i \notin \mathrm{dom}(R)$ and $X'_i$ denotes a fresh variable with respect to $\mathrm{dom}(R)$ if $X_i \in \mathrm{dom}(R)$. Notation $p[\mathbf{X}'/\mathbf{X}]$ denotes the process term that is obtained by substitution of the variables $\mathbf{X}$ in $p$ by $\mathbf{X}'$.

$$\frac{(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}],\sigma\rangle \xrightarrow{\alpha} \langle \begin{matrix}\checkmark\\ p'\end{matrix},\sigma'\rangle}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\}\ \mid\ p\ ]\!],\sigma\rangle \xrightarrow{\alpha} \langle \begin{matrix}\checkmark\\ [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\}\ \mid\ p'[\mathbf{X}/\mathbf{X}']\ ]\!]\end{matrix},\sigma'\rangle}\ 51$$

$$\frac{(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}],\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\}\ \mid\ p\ ]\!],\sigma\rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\}\ \mid\ p'[\mathbf{X}/\mathbf{X}']\ ]\!],\sigma'\rangle}\ 52$$

$$\frac{(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}],\sigma\rangle \overset{\xi}{\rightsquigarrow}}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\}\ \mid\ p\ ]\!],\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 53$$

Consider, for example, the process term $[\![_{\mathrm{R}}\ X \mapsto Y, Y \mapsto x := 0 \mid [\![_{\mathrm{R}}\ Y \mapsto x := 1 \mid X\ ]\!]]\!]$. Local recursion variable $Y$ with definition $Y \mapsto x := 1$ conflicts with the recursion variable definition $Y \mapsto x := 0$ from the outer scope. The renaming of the local variable in the rules of the recursion scope operator ensures that the process term behaves as $[\![_{\mathrm{R}}\ X \mapsto Y, Y \mapsto x := 0 \mid [\![_{\mathrm{R}}\ Z \mapsto x := 1 \mid X\ ]\!]]\!]$. Thus, the value of variable $x$ becomes 0. The renaming also ensures that the use of repetition $*p$ and guarded repetition $*b : p$, which are defined in Section 2.3.2 as $[\![_{\mathrm{R}}\ \{X \mapsto p;\ X\} \mid X\ ]\!]$ and $[\![_{\mathrm{R}}\ \{X \mapsto b \rightarrow \mathrm{skip};\ p;\ X\ [\!]\ \neg b \rightarrow \mathrm{skip}\} \mid X\ ]\!]$, respectively, cannot override existing recursion definitions using the same recursion variable $X$.

# Chapter 4

# Examples

## 4.1 Diode

An ideal diode can either block or conduct the current. When it blocks, the diode voltage $v \leq 0$, and the current $i = 0$. When it conducts, the diode voltage $v = 0$, and the current $i \geq 0$. Figure 4.1 shows the characteristics of an ideal diode.

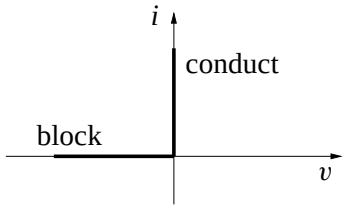


Figure 4.1: Characteristics of an ideal diode.

Using $\chi$, the process term modeling the diode is a single delay predicate:

$$(v < 0 \Rightarrow i = 0),\ (i > 0 \Rightarrow v = 0),\ v \leq 0,\ i \geq 0$$

where $\Rightarrow$ denotes logical implication, and the comma denotes conjunction.

## 4.2 Half wave rectifier circuit

Figure 4.2 shows a half wave rectifier circuit. It consists of a diode $D$, two resistors with resistance $R_0$ and $R_1$, respectively, a capacitor with capacity $C_0$, and a voltage source with voltage $v_0$. In the $\chi$ model, symbols $f$, $\pi$, $C_0$, $R_0$ and $R_1$ denote constants.

The $\chi$ model is as follows:

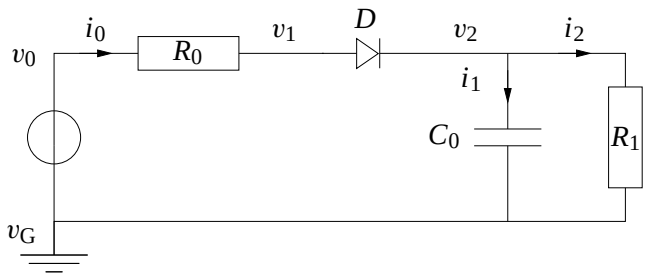


Figure 4.2: Half wave rectifier circuit.

$$
\begin{aligned}
&\langle\ \mathsf{cont}\ v_G, v_0, v_1, v_2, i_0, i_1, i_2 \\
&,\ v_2 = 0 \\
&\mid\ v_0 = \sin(2\pi f\,\mathsf{time}) \\
&\parallel\ R(i_0, v_0, v_1, R_0) \\
&\parallel\ D(i_0, v_1, v_2) \\
&\parallel\ C(i_1, v_2, v_G, C_0) \\
&\parallel\ R(i_2, v_2, v_G, R_1) \\
&\parallel\ v_G = 0 \\
&\parallel\ i_0 = i_1 + i_2 \\
&\rangle
\end{aligned}
$$

The process definitions of a diode $D$, a resistor $R$, and a capacitor $C$ follow below.

$$D(\mathsf{ext}\ i, v_{\text{in}}, v_{\text{out}}) = \\ [\![\ (v_{\text{out}} > v_{\text{in}} \Rightarrow i = 0),\ (i > 0 \Rightarrow v_{\text{in}} = v_{\text{out}}),\ v_{\text{out}} \geq v_{\text{in}},\ i \geq 0\ ]\!]$$

$$R(\mathsf{ext}\ i, v_{\text{in}}, v_{\text{out}}, \mathsf{val}\ R) = [\![\ v_{\text{in}} - v_{\text{out}} = i\,R\ ]\!]$$

Note that the same identifier $R$ is used to denote both the value parameter and the process name in the resistor model. Using a different name for the value parameter, e.g. $R_0$:

$$R(\mathsf{ext}\ i, v_{\text{in}}, v_{\text{out}}, \mathsf{val}\ R_0) = [\![\ v_{\text{in}} - v_{\text{out}} = i\,R_0\ ]\!]$$

does not change the meaning of the model. The capacitor model is:

$$C(\mathsf{ext}\ i, v_{\text{in}}, v_{\text{out}}, \mathsf{val}\ C) = [\![\ \mathsf{cont}\ v \mid v = v_{\text{in}} - v_{\text{out}},\ C\dot{v} = i\ ]\!]$$

After replacing the process instantiations by their process bodies as defined in Section 2.3.2, the following $\chi$ process is obtained:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ v_{\mathrm{G}}, v_0, v_1, v_2, i_0, i_1, i_2 \\
,\ v_2 = 0 \\
|\ v_0 = \sin(2\pi f \mathsf{time}) \\
\|\ [\![\ \mathsf{disc}\ R,\ R = R_0 \\
\quad |\ v_0 - v_1 = i_0 R \\
\quad ]\!] \\
\|\ [\![\ (v_2 > v_1 \Rightarrow i = 0),\ (i > 0 \Rightarrow v_1 = v_2),\ v_2 \geq v_1,\ i \geq 0\ ]\!] \\
\|\ [\![\ \mathsf{disc}\ C,\ \mathsf{cont}\ v,\ C = C_0 \\
\quad |\ v = v_2 - v_{\mathrm{G}},\ C\dot{v} = i_1 \\
\quad ]\!] \\
\|\ [\![\ \mathsf{disc}\ R,\ R = R_1 \\
\quad |\ v_2 - v_{\mathrm{G}} = i_2 R \\
\quad ]\!] \\
\|\ v_{\mathrm{G}} = 0 \\
\|\ i_0 = i_1 + i_2 \\
\rangle
\end{array}
$$

## 4.3 A game of billiards

Figure 4.3 shows a billiard table of dimensions $l$ and $h$ with a ball, as defined in [2]. Initially, the position and velocity of the ball are given by $(x_0, y_0)$ and $(v_{\mathrm{x}0}, v_{\mathrm{y}0})$, respectively. If the ball reaches a vertical side it rebounds, i.e. the sign of the horizontal velocity component $v_{\mathrm{x}}$ changes. The same occurs with the vertical velocity component $v_{\mathrm{y}}$ when the ball reaches a horizontal side. The combination of the signs of velocity components gives four different directions of movements.

The movement of the ball is modeled by the following $\chi$ model, where $l$, $h$, $x_0$, $y_0$, $v_{\mathrm{x}0}$, and $v_{\mathrm{y}0}$ denote constants. Each possible combination of directions is represented by a recursion definition. The recursion variables *RU*, *LU*, *LD*, and *RD* correspond with the directions right-up, left-up, left-down, and right-down, respectively.

The relation between the position $(x, y)$ and the velocity $(v_x, v_y)$ is given by $\dot{x} = v_x$, and $\dot{y} = v_y$. When a collision occurs, the velocity changes instantaneously.

$$
\begin{array}{lll}
\langle\ \mathsf{disc}\ v_{\mathrm{x}}, v_{\mathrm{y}},\ \mathsf{cont}\ x, y & & \\
,\ v_{\mathrm{x}} = v_{\mathrm{x}0}, v_{\mathrm{y}} = v_{\mathrm{y}0}, x = x_0, y = y_0 & & \\
,\ RU \mapsto x \leq l, y \leq h & []\, x = l & \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}}\ ;\ LU \\
 & []\, y = h & \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}}\ ;\ RD \\
,\ LU \mapsto x \geq 0, y \leq h & []\, x = 0 & \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}}\ ;\ RU \\
 & []\, y = h & \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}}\ ;\ LD \\
,\ LD \mapsto x \geq 0, y \geq 0 & []\, x = 0 & \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}}\ ;\ RD \\
 & []\, y = 0 & \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}}\ ;\ LU \\
,\ RD \mapsto x \leq l, y \leq h & []\, x = l & \rightarrow v_{\mathrm{x}} := -v_{\mathrm{x}}\ ;\ LD \\
 & []\, y = 0 & \rightarrow v_{\mathrm{y}} := -v_{\mathrm{y}}\ ;\ RU
\end{array}
$$

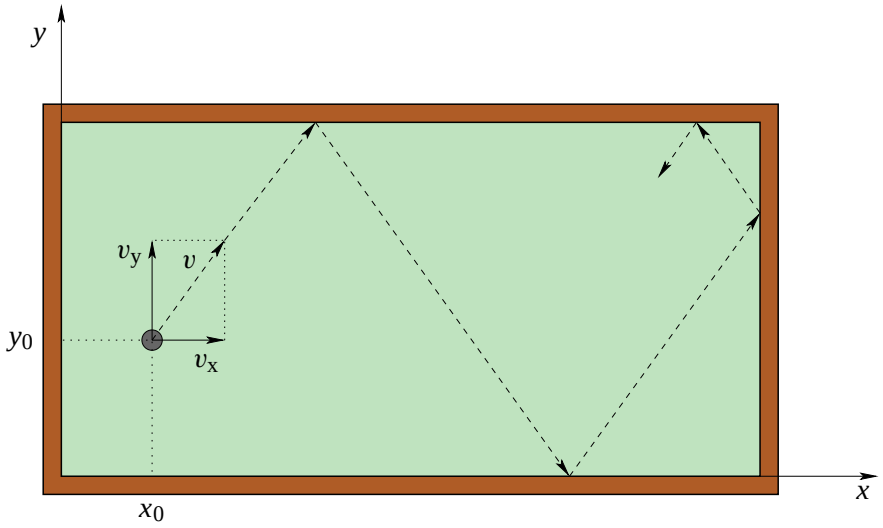


Figure 4.3: Billiard table

$$
\begin{array}{l}
\mid \dot{x} = v_x, \dot{y} = v_y \parallel RU \\
\rangle
\end{array}
$$

In the following model, the four recursion definitions are merged into one definition with ODE $\dot{x} = v_x, \dot{y} = v_y$. The delay enabling brackets $[\ldots]$ around the assignments, e.g. $[v_x := -v_x]$, are required. Without them, a zero velocity of either $v_x$ or $v_y$ would result in an infinite trace of assignments $v_x := 0$ or $v_x := 0$, respectively, because without the delay enabling brackets, the assignment and thus the alternative composition cannot delay.

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ v_x, v_y,\ \mathsf{cont}\ x, y \\
,\ v_x = v_{x0}, v_y = v_{y0}, x = x_0, y = y_0 \\
\mid *\ (\ \dot{x} = v_x, \dot{y} = v_y, 0 \le x \le l, 0 \le y \le h \\
\quad []\ (x = 0 \vee x = l) \rightarrow [v_x := -v_x] \\
\quad []\ (y = 0 \vee y = h) \rightarrow [v_y := -v_y] \\
\quad ) \\
\rangle
\end{array}
$$

## 4.4 Constrained pendulum

Figure 4.4 shows a constrained pendulum that is also defined in [50, 14]. The equations of motion of this pendulum are given by Equation 4.1. The angle between the pendulum and the vertical is denoted by $\theta$, $\omega$ denotes the angular velocity of the pendulum, and $l$ denotes the distance between the rotation point and the mass.

$$\begin{aligned} \dot{\theta} &= \omega \\ ml\dot{\omega} &= -mg\sin(\theta) - dl\omega \end{aligned} \tag{4.1}$$

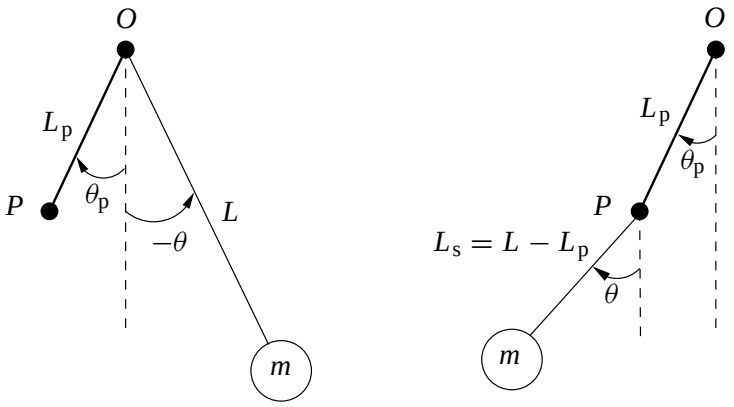


Figure 4.4: Constrained Pendulum.

The mass and maximum length of the pendulum are represented by $m$ and $L$, respectively. The damping coefficient and the acceleration due to gravity are denoted by $d$ and $g$. The angle of the constraint is denoted by $\theta_\mathrm{p}$. In order to keep the example as small and clear as possible, it is assumed that $\theta_\mathrm{p} \geq 0$ and $|\theta| \leq \pi/2$. Also, it is assumed that the pendulum always remains in a straight line from the rotation point to the end point. The $\chi$ model is:

$\langle$ cont $\theta, \omega,$ alg $l$
, $\theta = \theta_0, \omega = \omega_0$
, long $\mapsto l = L$, $\theta \leq \theta_\mathrm{p}$ [] $[\omega := \frac{L}{L_\mathrm{s}}\omega]$; short
, short $\mapsto l = L_\mathrm{s}$, $\theta \geq \theta_\mathrm{p}$ [] $[\omega := \frac{L_\mathrm{s}}{L}\omega]$; long
| skip; long [] skip; short
‖ $\dot{\theta} = \omega$
, $ml\dot{\omega} = -mg\sin(\theta) - dl\omega$
$\rangle$,

where $\theta_0$ and $\omega_0$ denote constants representing the initial values of $\theta$ and $\omega$, respectively. When $\theta \leq \theta_\mathrm{p}$ or $\theta \geq \theta_\mathrm{p}$, the pendulum can delay in mode long or short, respectively. In mode long, the assignment $\omega := \frac{L}{L_\mathrm{s}}\omega$ can be executed only if the new state after the assignment to $\omega$ is consistent with the constraints $l = L_\mathrm{s}$, $\theta \geq \theta_\mathrm{p}$ of mode short, because a process cannot enter an inconsistent state. Therefore, mode switches are possible only for $\theta = \theta_\mathrm{p}$. The delay enabling brackets are needed around the assignment in $[\omega := \frac{L}{L_\mathrm{s}}\omega]$, because otherwise the assignment and the alternative composition would not be able to delay.

## 4.5 Dry friction phenomenon

Figure 4.5 shows a driving force $F_\mathrm{d}$ applied to a body on a flat surface with frictional force $F_\mathrm{f}$. When the body is moving with positive velocity $v$, the frictional force is given by $F_\mathrm{f} = \mu F_\mathrm{N}$, where $F_\mathrm{N} = mg$. When the velocity of the body equals zero and $-\mu_0 F_\mathrm{N} \leq F_\mathrm{d} \leq \mu_0 F_\mathrm{N}$, where $\mu_0 > \mu$, the frictional force neutralizes the applied driving force.

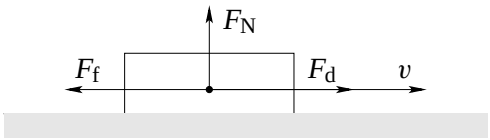


Figure 4.5: Dry Friction.

In the $\chi$ specification of the dry friction phenomenon, modes neg, stop, and pos are specified by means of mode definitions. Mode stop can be maintained for as long as the delay predicate $v = 0, -\mu_0 F_\mathrm{N} \leq F_\mathrm{d} \leq \mu_0 F_\mathrm{N}$ can delay. One of the two skip actions can be executed when the corresponding guard $F_\mathrm{d} \leq -\mu_0 F_\mathrm{N}$ or $F_\mathrm{d} \geq \mu_0 F_\mathrm{N}$ becomes true, and the state is consistent with the constraints $v \leq 0$ or $v \geq 0$ of the respective mode neg or pos that is activated after the skip action.

The mode pos (or neg) is maintained until the guard $F_\mathrm{d} < \mu_0 F_\mathrm{N}$ (or $F_\mathrm{d} > -\mu_0 F_\mathrm{N}$) becomes true. The skip action after the guard cannot delay. Therefore, when the guard becomes true, the skip action must be executed. Subsequently, mode stop becomes active again.

Initially either mode neg, stop or pos is chosen by means of the internal skip action (skip; neg [] skip; stop [] skip; pos), based on the initial values of $v$ and $F_\mathrm{d}$. Identifier $f$ denotes some function $\in \mathbb{R} \to \mathbb{R}$; $m$, $F_\mathrm{N}$, $\mu_0$, $\mu$ ($\mu_0 > \mu$) denote constants.

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x, v,\ \mathsf{alg}\ F_\mathrm{d} \\
,\ x = 0, v = 0 \\
,\ \mathrm{stop} \mapsto v = 0,\ -\mu_0 F_\mathrm{N} \leq F_\mathrm{d} \leq \mu_0 F_\mathrm{N}\ []\ [F_\mathrm{d} \leq -\mu_0 F_\mathrm{N} \to \mathrm{skip}];\ \mathrm{neg} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad []\ [F_\mathrm{d} \geq \mu_0 F_\mathrm{N} \quad \to \mathrm{skip}];\ \mathrm{pos} \\
,\ \mathrm{pos} \mapsto m\dot{v} = F_\mathrm{d} - \mu F_\mathrm{N},\ v \geq 0\ []\ F_\mathrm{d} < \mu_0 F_\mathrm{N} \to \mathrm{skip};\ \mathrm{stop} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad []\ [F_\mathrm{d} \leq -\mu_0 F_\mathrm{N} \to \mathrm{skip}];\ \mathrm{neg} \\
,\ \mathrm{neg} \mapsto m\dot{v} = F_\mathrm{d} + \mu F_\mathrm{N},\ v \leq 0\ []\ F_\mathrm{d} > -\mu_0 F_\mathrm{N} \to \mathrm{skip};\ \mathrm{stop} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad []\ [F_\mathrm{d} \geq \mu_0 F_\mathrm{N} \to \mathrm{skip}];\ \mathrm{pos} \\
|\ F_\mathrm{d} = f(\mathsf{time}), \dot{x} = v \\
\|\ \mathrm{skip};\ \mathrm{neg}\ []\ \mathrm{skip};\ \mathrm{stop}\ []\ \mathrm{skip};\ \mathrm{pos} \\
\rangle
\end{array}
$$

## 4.6 Railroad gate control

Consider a train on a circular track, a gate and a controller. When the train approaches the gate, the controller must lower the gate. The controller has a reaction delay $u$ of at most 5 time units. After the train has passed the gate the controller must raise the gate. The purpose of the model is to determine whether or not the gate is always fully lowered when the train is at a certain distance from the gate. Figures 4.6, 4.7, and 4.8 show automaton models of the train, gate and controller respectively. Together, they form the rail gate control system as defined in [27]. Note that in a vertex, the predicate at the top denotes the flow predicate, and the predicate at the bottom denotes the invariant predicate of the vertex. Furthermore, as usual, event labels of edges which do not have to synchronize with other edges, and init predicates false are omitted in the figures.

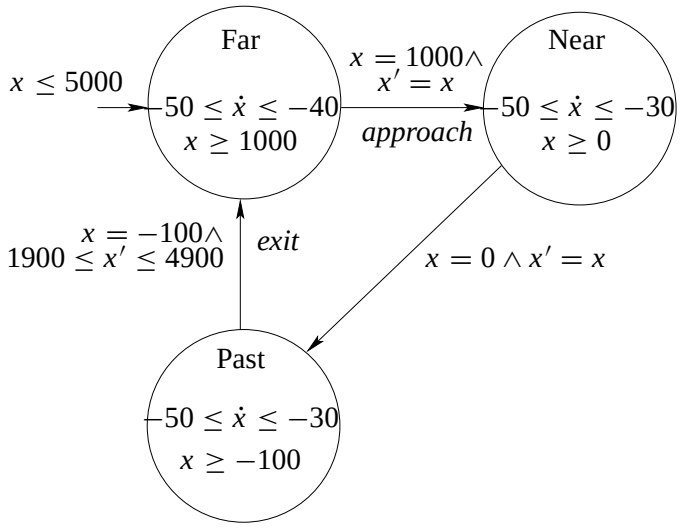


Figure 4.6: *Train* Automaton.

The $\chi$ model takes into account that there is only one train on the circular track, as defined in [27], which implies that the transitions of the self loops of the controller automaton can never occur. Figure 4.9 shows the iconic $\chi$ model of the railroad gate controller. The dashed lines with arrow heads represent synchronization channels (*approach*, *exit*, *raise*, *lower*), no data is communicated.

The following process is a formal specification of the informal iconic model from Figure 4.9. Variable $x$ and $y$ are initialized to a value $\leq 5000$ and 90, respectively. The maximum reaction delay (5 time units) of the controller is specified in its process instantiation $C$.

$\langle$ cont $x, y$
, chan *approach*, *exit*, *raise*, *lower*
, $x \leq 5000, y = 90$

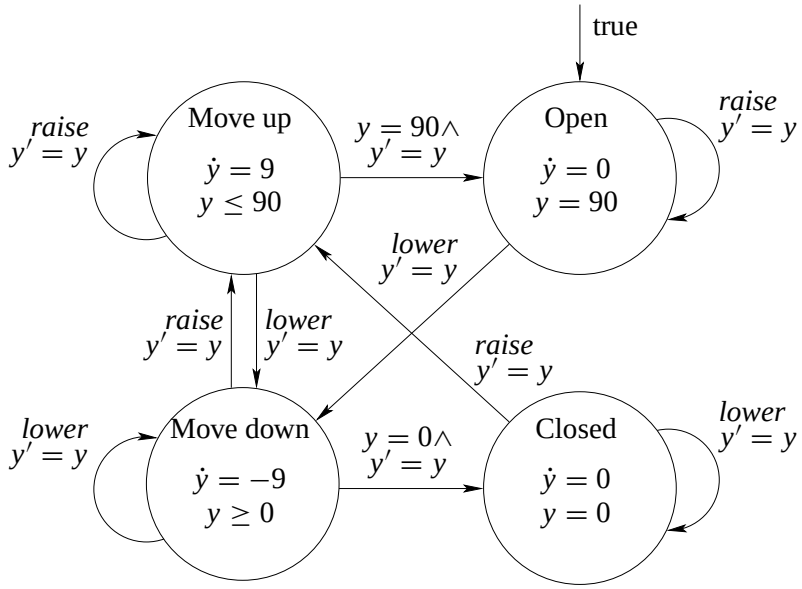


Figure 4.7: *Gate* Automaton.

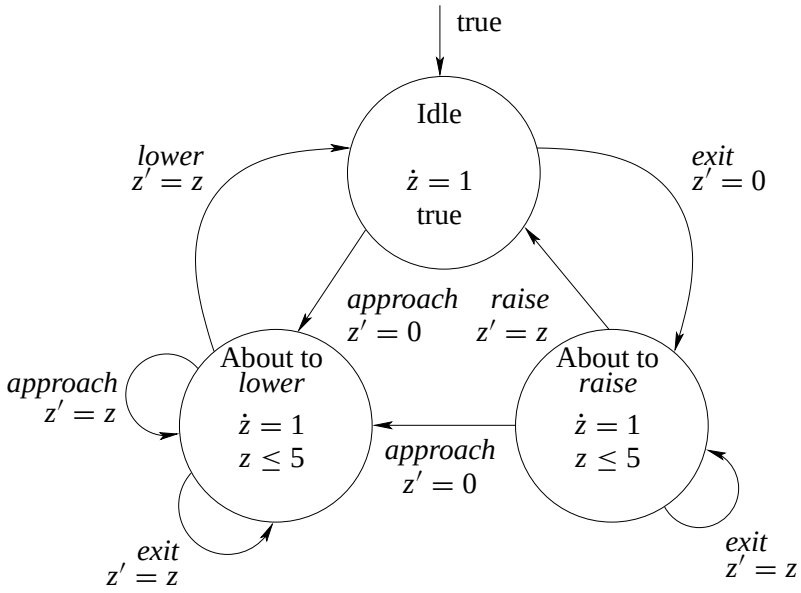


Figure 4.8: *Control* Automaton.

$\mid$ *Train*$(x, approach, exit)$
$\parallel$ *Gate*$(y, raise, lower)$
$\parallel$ $C(approach, exit, raise, lower, 5)$

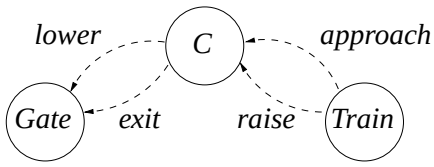


Figure 4.9: Iconic $\chi$ model of the railroad gate controller.

$\rangle$

The train is modeled by the following process definition:

$$
\begin{array}{l}
\mathit{Train}(\mathsf{ext}\ x,\ \mathsf{chan}\ \mathit{approach}, \mathit{exit}) = \\
[\![\ *(\ (\dot{x} \in [-40, -50]\ []\ x \geq 1000 \rightarrow \mathit{approach}\,!!) \\
\quad ;\ (\dot{x} \in [-30, -50]\ []\ x \leq -100 \rightarrow \mathit{exit}\,!!;\ x : x \in [1900, 4900]) \\
\quad ) \\
]\!]
\end{array}
$$

The process definition consists of an infinite loop $*(\cdots)$. The velocity $\dot{x}$ of the train can be any function of time, the value of which remains between $-50$ and $-40$. The process waits until the train has reached position $x = 1000$ and then synchronizes with the controller (*approach*!!). The train is now approaching the gate. If the train has reached the exit position, such that $x = -100$, the process synchronizes with the controller, the position $x$ of the train is reset to a value between 1900 and 4900, and the loop is re-executed.

The gate is modeled by the following process definition:

$$
\begin{array}{l}
\mathit{Gate}(\mathsf{ext}\ y,\ \mathsf{chan}\ \mathit{raise}, \mathit{lower}) = \\
[\![\ \mathsf{disc}\ n,\ n = 0 \\
|\ \dot{y} = n \\
\|\ *(\ n < 0 \wedge y \leq 0 \rightarrow n := 0 \\
\quad []\ n > 0 \wedge y \geq 90 \rightarrow n := 0 \\
\quad []\ \mathit{raise}\,?;\ n := 9 \\
\quad []\ \mathit{lower}\,?;\ n := -9 \\
\quad ) \\
]\!]
\end{array}
$$

It consists of a parallel composition of an equation ($\dot{y} = n$), where $n$ denotes a local discrete variable, and an infinite loop. This infinite loop is an alternative composition of four process terms. The first process term waits until the gate is lowered ($y = 0$) and then stops the gate from lowering ($n := 0$). The second process term waits until the gate is raised ($y = 90$). The third and fourth process term wait for synchronization with the controller in order to raise or

lower the gate (*raise*? and *lower*?, respectively). The four process terms delay in parallel until $y$ becomes equal to 0 or 90, or one of the synchronizations (*raise*? or *lower*?) succeeds.

The controller is modeled by the following process definition:

$$
\begin{array}{l}
C(\mathsf{chan}\ \mathit{approach}, \mathit{exit}, \mathit{raise}, \mathit{lower}, \mathsf{val}\ u) = \\
[\![\ \mathsf{disc}\ \mathit{atr},\ \mathit{atr} = \mathrm{false} \\
\mid *(\ \mathit{approach}?;\ \mathit{atr} := \mathrm{false};\ (\Delta u\ []\ [\mathrm{skip}]);\ \Delta t;\ \mathit{lower}!! \\
\quad []\ \mathit{exit}?;\ \mathit{atr} := \mathrm{true} \\
\quad []\ \mathit{atr} \rightarrow (\Delta u\ []\ [\mathrm{skip}]);\ \mathit{atr} := \mathrm{false};\ \mathit{raise}!! \\
\quad ) \\
]\!]
\end{array}
$$

The main part is an infinite loop of three alternatives. The process waits for one of the following events to occur: an approaching train (*approach*?), a leaving train (*exit*?), or if *atr* is true, the end of the reaction delay ($\Delta u$ [] [skip]) that precedes raising of the gate. Process term $\Delta u$ [] [skip], where $u$ denotes the maximum reaction delay in the controller, models a non-deterministic delay between 0 and $u$.

Boolean variable *atr* is true if and only if the hybrid automaton that models the controller is in control mode (vertex) 'About to raise'. Note that variable $z$ in the hybrid automaton is used to model a clock. In $\chi$, clocks need not be modeled explicitly. Delay process terms $\Delta u$ are used for that purpose.

## 4.7 Glider take-off

Figure 4.10 shows a glider that is towed off the ground by a tow plane. The position, velocity and acceleration of the tow plane are given by $x_1$, $v_1$, $a$, respectively. The position and velocity of the glider are given by $x_2$ and $v_2$.

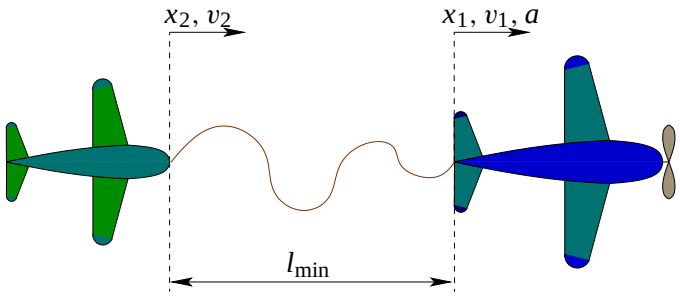


Figure 4.10: Glider take-off.

Initially, the tow plane and glider are standing still at a distance of $l_{\min}$. After one unit of time, the tow plane very slowly accelerates ($a := 0.02$) until the tow cable is at its maximum length

of $l_{\max}$. At that moment, the velocity of the glider jumps discontinuously to the velocity of the tow plane. We assume the mass of the glider to be considerably smaller than the mass of the tow plane. The tow plane then accelerates ($a := 0.5$) until its velocity is at $v_{\max}$. After another $t$ units of time, the glider releases the tow cable, and continues on its own. Its velocity is then assumed to be determined by the air resistance, which is proportional to the squared velocity ($kv_2^2$), and the propelling forces $F$, which we assume constant in the model below:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s, a, \mathsf{cont}\ x_1, x_2, v_1, v_2 \\
,\ s = \text{STOP}, a = 0, x_1 = l_{\min}, x_2 = 0, v_1 = 0 \\
\mid\ \dot{x}_1 = v_1,\ \dot{x}_2 = v_2,\ \dot{v}_1 = a \\
\parallel\ s = \text{STOP} \ \rightarrow v_2 = 0 \\
\parallel\ s = \text{TOW} \ \rightarrow v_2 = v_1 \\
\parallel\ s = \text{FLY} \ \rightarrow \dot{v_2} = F - kv_2^2 \\
\parallel\ \Delta 1;\ a := 0.02 \\
;\ x_1 - x_2 \geq l_{\max} \rightarrow (\mathsf{jump}\ v_2 \mid s := \text{TOW}) \\
;\ a := 0.5;\ v \geq v_{\max} \rightarrow a := 0 \\
;\ \Delta t;\ s := \text{FLY} \\
\rangle
\end{array}
$$

In the model, $k$ denotes a constant, and enumeration variable $s$ denotes the state of the glider. When the distance between the tow plane and the glider becomes equal to the maximum length of the cable, the glider abruptly starts moving. This is modeled by ($\mathsf{jump}\ v_2 \mid s := \text{TOW}$). The jump enabling operator ($\mathsf{jump}\ v_2 \mid \ldots$) enables jumps for continuous variable $v_2$ when assignment $s := \text{TOW}$) is executed. This is necessary, because $v_2$ is declared as a (non-jumping) continuous variable. The only assignment where $v_2$ must be able to jump is the assignment $s := \text{TOW}$, because then $v_2$ must discontinuously change to the value of $v_1$ in order to satisfy equation $v_2 = v_1$ that must hold for $s = \text{TOW}$. In this example, the relation $v_1 = v_2$ in mode TOW is so straightforward, that the jumping behavior of variable $v_2$ when mode TOW becomes active can also be modeled explicitly by means of a multi-assignment ($s, v_2 := \text{TOW}, v_1$), as indicated in the model below. The previous model with the jump enabling operator is more general, because it can also be used in cases where the algebraic constraints are so complex that is becomes difficult, or impossible, to explicitly calculate the new value of the jumping variable after the discontinuity.

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s, a, \mathsf{cont}\ x_1, x_2, v_1, v_2 \\
,\ s = \text{STOP}, a = 0, x_1 = l_{\min}, x_2 = 0, v_1 = 0 \\
\mid\ \dot{x}_1 = v_1,\ \dot{x}_2 = v_2,\ \dot{v}_1 = a \\
\parallel\ s = \text{STOP} \ \rightarrow v_2 = 0 \\
\parallel\ s = \text{TOW} \ \rightarrow v_2 = v_1 \\
\parallel\ s = \text{FLY} \ \rightarrow \dot{v_2} = F - kv_2^2 \\
\parallel\ \Delta 1;\ a := 0.02 \\
;\ x_1 - x_2 \geq l_{\max} \rightarrow s, v_2 := \text{TOW}, v_1 \\
;\ a := 0.5;\ v \geq v_{\max} \rightarrow a := 0
\end{array}
$$

$; \Delta t;\ s := \text{FLY}$
$\rangle$

## 4.8 Bottle filling system

The bottle filling system from Figure 4.11 consists of a liquid storage tank, and two identical bottle filling lines.

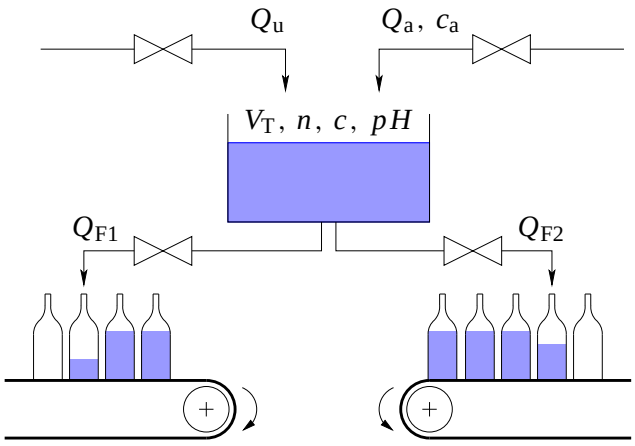


Figure 4.11: The bottle filling system.

The bottles are filled with liquid from the storage tank. A control system keeps the volume $V$ in the storage tank between 2 and 10, and the pH level (acidity) of the liquid in the storage tank between 7 and 7.1. The liquid in the storage tank slowly becomes less acid (pH level increases). To correct this, a strong acid is dribbled into the storage tank when the acidity of the liquid becomes too low ($pH \geq 7.1$).

Figure 4.12 shows the iconic model of the bottle filling system. The lines ending in a small circle represent shared variables ($V_T$, $Q_{F_1}$, $Q_{F_1}$).

The acid and liquid supply processes are not modeled, since we consider the acid always to be available, and we are not interested in the amount of acid that is used. The $\chi$ specification of the bottle filling system is as follows:

$\langle\ \mathsf{cont}\ V_T,\ \mathsf{alg}\ Q_{F1}, Q_{F2}$
$,\ V_T = 2$
$|\ T(V_T, Q_{F1}, Q_{F2})$
$\|\ F(V_T, Q_{F1})$
$\|\ F(V_T, Q_{F2})$
$\rangle$

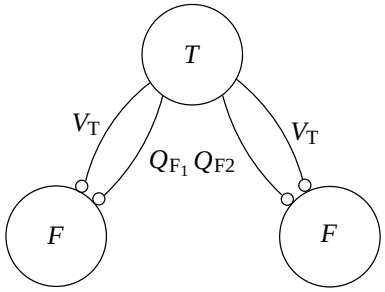


Figure 4.12: Iconic representation of the bottle filling system model.

The storage tank and the two bottle filling lines are connected by means of the variables $Q_{F1}$, and $Q_{F2}$, respectively. Since a bottle may start filling only if the storage tank contains at least a volume of 2, the volume $V_T$ of the storage tank is available in both bottle filling processes.

The molar quantity and molar concentration of the acid in the storage tank are denoted by $n$ and $c$, respectively. The incoming flows of liquid and acid of the liquid storage tank $T$ are denoted by $Q_u$ and $Q_a$, respectively. The outgoing flows to the two bottle filling processes are denoted by $Q_{F1}$, and $Q_{F2}$, respectively.

It is assumed that the liquids are incompressible, and that the volumes of the fluids remain the same when they are mixed. In such a case, the volume $V$ of the mixed liquid equals the sum of its components which leads to the following equation

$$\dot{V} = Q_u + Q_a - Q_{F1} - Q_{F2}.$$

Next, the mass balance (actually mol balance) for the dissolved substance is derived. Acid comes into the tank by means of the flows $Q_u$ and $Q_a$. Acid leaves the tank in outgoing flows $Q_{F1}$ and $Q_{F2}$. Because the concentrations are in [mol/m$^3$], they can be directly multiplied with the flows (in [m$^3$/s]), which leads to

$$\dot{n} = c_u Q_u + c_a Q_a - c Q_{F1} - c Q_{F2},$$

where $c_u$ and $c_a$ denote the concentrations of acid in the flows $Q_u$ and $Q_a$.

$$n = cV.$$

The gradual reduction of the acidity of the liquid is modeled by means of a constant $K_{loss}$, which leads to

$$\dot{n} = c_u Q_u + c_a Q_a - c Q_{F1} - c Q_{F2} - K_{loss} V.$$

It is assumed that the acid is completely decomposed. Taking into account that the units of $c$ are in [mol/m$^3$] instead of [mol/l], the pH is given by

$$pH = -\log c/1000.$$

The $\chi$ specification of the liquid storage tank follows below, where symbols $Q_{seta}$, $Q_{setu}$, $c_a$, $c_u$, and $K_{loss}$ denote constants:

$$
\begin{array}{l}
T(\mathsf{ext}\ V, Q_{\mathrm{F1}}, Q_{\mathrm{F2}}) \\
[\![\ \mathsf{disc}\ \alpha, \beta,\ \mathsf{cont}\ n,\ \mathsf{alg}\ pH, c, Q_{\mathrm{a}}, Q_{\mathrm{u}} \\
,\ \alpha = 0, \beta = 0, pH = 7 \\
|\quad \dot{V} = Q_{\mathrm{u}} + Q_{\mathrm{a}} - Q_{\mathrm{F1}} - Q_{\mathrm{F2}} \\
,\quad \dot{n} = c_{\mathrm{u}} Q_{\mathrm{u}} + c_{\mathrm{a}} Q_{\mathrm{a}} - c Q_{\mathrm{F1}} - c Q_{\mathrm{F2}} - K_{\mathrm{loss}} V \\
,\quad n = cV \\
,\ pH = -\log c/1000 \\
,\ Q_{\mathrm{a}} = \alpha Q_{\mathrm{seta}} \\
,\ Q_{\mathrm{u}} = \beta Q_{\mathrm{setu}} \\
\|\ *(\ pH \geq 7.1 \rightarrow \alpha := 1;\ pH \leq 7 \rightarrow \alpha := 0\ ) \\
\|\ *(\ V \leq 5 \rightarrow \beta := 1;\ V \geq 10 \rightarrow \beta := 0\ ) \\
]\!]
\end{array}
$$

The model of the liquid storage tank $T$ illustrates that a differential variable, such as variable $n$, is not necessarily initialized. In this case, instead, the algebraic variable $pH$ is initialized ($pH = 7$). The continuous variables of the bottle filling system with tank $T$, can be declared in many different ways.

In most cases, the differential variables, in this case $V$ and $n$, are declared as (non-jumping) continuous variables. The other variables, not occurring with a dot (derivative) are then declared as algebraic variables. This ensures that the differential variables can be assigned new values, causing discontinuities. The algebraic variables will then simultaneously jump to their new values satisfying the equations. This declaration scheme is used in process $T$. Note that variable $V$ is an external variable that is declared as a (non-jumping) continuous variable in the preceding $\chi$ process that defines the complete bottle filling system. Note that even though $pH$ is an algebraic variable, which is not normally assigned new values, $pH$ can be initialized, in this case to a value of 7, in the initialization predicate.

In process $T$, the only discontinuities in continuous variables occur in the flows $Q_{\mathrm{F1}}$, $Q_{\mathrm{F2}}$, $Q_{\mathrm{a}}$, and $Q_{\mathrm{u}}$, that are switched on and off discontinuously in process $T$, and in process $F$ that follows below. Therefore, the algebraic variables apart from these flows could just as well have been declared as (non-jumping) continuous variables as in $\mathsf{cont}\ n, pH, c$.

The behavior of the model is explained as follows. Initially, the $pH$ of the liquid in the storage tank equals 7. It is assumed that the pH level of the incoming liquid is 7 or more, since the acidity controller can only make the acidity of the storage tank increase, causing the pH to decrease. If the pH value exceeds the maximum value ($pH \geq 7.1$), the acid valve is opened ($\alpha := 1$) so that acid is dribbled into the tank. Dribbling of the acid continues until the pH value comes back at 7, where after the valve is closed ($\alpha := 0$). In a similar way, the controller tries to keep the level of the storage tank between 5 and 10.

The model of a bottle filling line follows below, where symbols $Q_{\mathrm{setF}}$, and $t_{\mathrm{tr}}$ denote constants.

$$
\begin{array}{l}
F(\mathsf{ext}\ V_{\mathrm{T}}, Q_{\mathrm{F}}) = \\
[\![\ \mathsf{disc}\ \alpha,\ \mathsf{cont}\ V
\end{array}
$$

$$
\begin{array}{l}
, \ \alpha = 0, V = 0 \\
| \quad \dot{V} = Q_{\mathrm{F}} \\
, \ Q_{\mathrm{F}} = \alpha Q_{\mathrm{setF}} \\
\| * (\ pH \leq 7.1 \rightarrow \alpha := 1 \\
\quad ; * \alpha = 1 : (\ V \geq 1 \rightarrow \alpha := 0 \\
\qquad\qquad [] \, V_{\mathrm{T}} \leq 0.5 \rightarrow \alpha := 0; \ V_{\mathrm{T}} \geq 0.7 \rightarrow \alpha := 1 \\
\qquad\qquad ) \\
\quad ; \ \Delta t_{\mathrm{tr}}; \ V := 0 \\
\quad ) \\
]\|
\end{array}
$$

The valve switching the flow $Q_{\mathrm{F}}$ is modeled by means of discrete variable $\alpha$. When the acidity of the liquid in the storage tank is less than or equal to 7.1, the bottle filling process can be started ($\alpha := 1$). Filling stops when the volume in the storage tank drops below 0.5 ($V_{\mathrm{T}} \leq 0.5 \rightarrow \alpha := 0$). Filling resumes when the volume in the storage tank is at least 0.7. When the bottle is full ($V \geq 1$), the valve is closed. The time needed to place a new bottle under the filling nozzle is given by $t_{\mathrm{tr}}$. After that, the bottle volume is reset to 0, which models the arrival of a new bottle, and the filling process is repeated.

## 4.9 Conveyor system

Figure 4.13 shows a conveyor system that is used for transportation and buffering of boxes. It consists of a line of conveyor belts driven by motors. Each conveyor belt is equipped with a sensor (represented in the figure by a small rectangle), that detects the presence of a box.

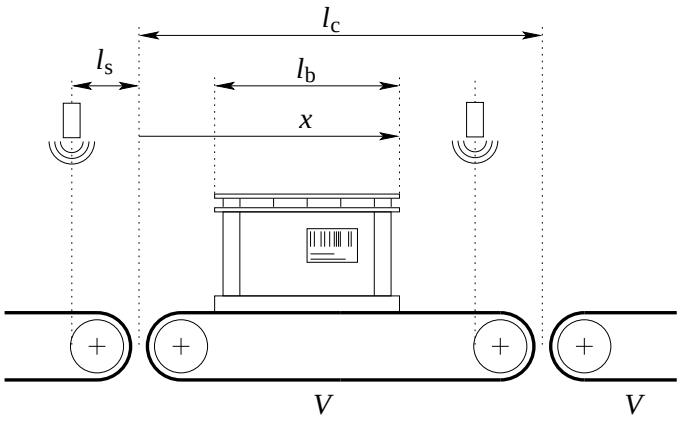


Figure 4.13: The conveyor system.

Figure 4.14 shows the iconic model of a generator $G$, two conveyor belts $V$ and the associated control processes $C$. Processes $V_{\mathrm{E}}$ and $C_{\mathrm{E}}$ are added to obtain a closed system; they do not

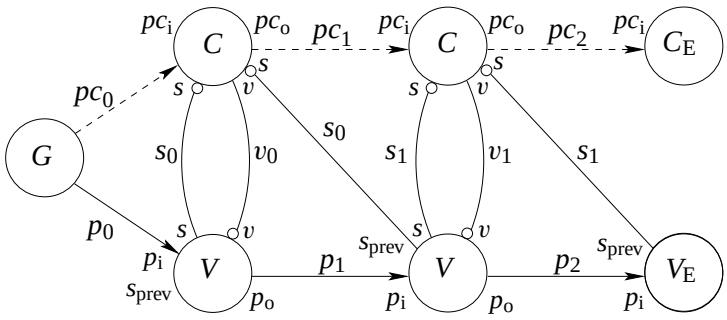


Figure 4.14: Iconic model of the conveyor system.

model actual behavior. The model is a simplified version of the model treated in [45]. The dashed lines with arrow heads represent directed synchronization channels ($pc_0$, $pc_1$, $pc_2$), the solid lines with arrow heads represent directed communication channels ($p_0$, $p_1$, $p_2$), and the lines ending in a small circle represent shared variables ($s_0$, $s_1$, $v_0$, $v_1$). The $\chi$ specification of the iconic model from Figure 4.14 is as follows:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s_G, s_0, s_1, l_s,\ \mathsf{cont}\ v_0, v_1 \\
,\ \mathsf{chan}\ pc_0, pc_1, pc_2,\ p_0,\ p_1,\ p_2 \\
,\ s_0 = \mathrm{false}, s_1 = \mathrm{false}, l_s = 2, v_0 = 0, v_1 = 0 \\
|\ G(pc_0,\ p_0) \\
\|\ C(v_0, s_0, pc_0, pc_1) \\
\|\ C(v_1, s_1, pc_1, pc_2) \\
\|\ C_E(pc_2) \\
\|\ V(s_G, s_0, v_0,\ p_0,\ p_1, 10, 15, l_s) \\
\|\ V(s_0, s_1, v_1,\ p_1,\ p_2, 10, 15, l_s) \\
\|\ V_E(s_1,\ p_2, 10, l_s) \\
\rangle\ ,
\end{array}
$$

where $l_s$ represents the distance $l_s$ as shown in Figure 4.13. Channels $p_0, p_1, p_2$ are used to communicate box numbers. Channels $pc_0, pc_1, pc_2$ are directed synchronization channels. Variables $s_0, s_1, v_0, v_1$ are shared variables, where $s_0$ and $s_1$ represent the sensors that indicate the presence of a box, and $v_0$ and $v_1$ are actuators that determine the velocity of the respective conveyors. The process definitions are:

$$
\begin{array}{l}
G(\mathsf{chan}\ pc_o,\ p_o) = \\
[\![\ \mathsf{disc}\ box,\ box = 1 \\
|\ *(\ pc_o!;\ p_o!box;\ box := box + 1\ ) \\
]\!]
\end{array}
$$

$$
C(\mathsf{ext}\ v, s,\ \mathsf{chan}\ pc_i, pc_o) =
$$

$$
\begin{array}{l}
[\![ *(\ v := 1;\ \neg s \to pc_{\mathrm{i}}?;\ s \to v := 0;\ pc_{\mathrm{o}}!\ ) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
V(\mathsf{ext}\ s_{\mathrm{prev}}, s, v,\ \mathsf{chan}\ p_{\mathrm{i}}, p_{\mathrm{o}},\ \mathsf{val}\ l_{\mathrm{b}}, l_{\mathrm{c}}, l_{\mathrm{s}}) = \\
[\![ \mathsf{disc}\ box,\ \mathsf{cont}\ x \\
,\ box = 0, x = -1 \\
|\ box = 0 \to \dot{x} = 0 \parallel box \neq 0 \to \dot{x} = v \\
\parallel *(\ p_{\mathrm{i}}?box;\ x := 0 \\
\quad ;\ x \geq l_{\mathrm{b}} - l_{\mathrm{s}} \to s_{\mathrm{prev}} := \mathrm{false} \\
\quad ;\ x \geq l_{\mathrm{c}} - l_{\mathrm{s}} \to s := \mathrm{true} \\
\quad ;\ x \geq l_{\mathrm{c}} \to p_{\mathrm{o}}!box;\ x, box := -1, 0 \\
\quad ) \\
]\!]
\end{array}
$$

$$
C_{\mathrm{E}}(\mathsf{chan}\ pc_{\mathrm{i}}) = [\![ *(\ pc_{\mathrm{i}}?\ ) ]\!]
$$

$$
\begin{array}{l}
V_{\mathrm{E}}(\mathsf{ext}\ s_{\mathrm{prev}},\ \mathsf{chan}\ p_{\mathrm{i}},\ \mathsf{val}\ l_{\mathrm{b}}, l_{\mathrm{s}}) = \\
[\![ \mathsf{disc}\ box \\
|\ *(\ p_{\mathrm{i}}?box;\ \Delta(l_{\mathrm{b}} - l_{\mathrm{s}});\ s_{\mathrm{prev}} := \mathrm{false}\ ) \\
]\!]
\end{array}
$$

After replacing the process instantiations by their process bodies as defined in Section 2.3.2, the following $\chi$ process is obtained:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ s_{\mathrm{G}}, s_0, s_1, l_{\mathrm{s}},\ \mathsf{cont}\ v_0, v_1 \\
,\ \mathsf{chan}\ pc_0, pc_1, pc_2,\ p_0, p_1, p_2 \\
,\ s_0 = \mathrm{false}, s_1 = \mathrm{false}, l_{\mathrm{s}} = 2, v_0 = 0, v_1 = 0 \\
|\ [\![ \mathsf{disc}\ box,\ box = 1 \\
\quad |\ *(\ pc_0!;\ p_0!box;\ box := box + 1\ ) \\
\quad ]\!] \\
\parallel [\![ *(\ v_0 := 1;\ \neg s_0 \to pc_0?;\ s_0 \to v_0 := 0;\ pc_1!\ )]\!] \\
\parallel [\![ *(\ v_1 := 1;\ \neg s_1 \to pc_1?;\ s_1 \to v_1 := 0;\ pc_2!\ )]\!] \\
\parallel [\![ *(\ pc_2?\ ) ]\!] \\
\parallel [\![ \mathsf{disc}\ box, l_{\mathrm{b}}, l_{\mathrm{c}}, \mathrel{\lrcorner}\!l_{\mathrm{s}},\ \mathsf{cont}\ x \\
\quad ,\ box = 0, x = -1, l_{\mathrm{b}} = 10, l_{\mathrm{c}} = 15, \mathrel{\lrcorner}\!l_{\mathrm{s}} = l_{\mathrm{s}} \\
\quad |\ (\ box = 0 \to \dot{x} = 0 \parallel box \neq 0 \to \dot{x} = v_0 \\
\qquad \parallel *(\ p_0?box;\ x := 0 \\
\qquad\quad ;\ x \geq l_{\mathrm{b}} - \mathrel{\lrcorner}\!l_{\mathrm{s}} \to s_{\mathrm{G}} := \mathrm{false} \\
\qquad\quad ;\ x \geq l_{\mathrm{c}} - \mathrel{\lrcorner}\!l_{\mathrm{s}} \to s_0 := \mathrm{true} \\
\qquad\quad ;\ x \geq l_{\mathrm{c}} \to p_1!box;\ x, box := -1, 0 \\
\qquad\quad ) \\
\qquad )
\end{array}
$$

$$
\begin{array}{l}
\quad ]\!] \\
\| \; [\![ \; \mathsf{disc}\ box \mid *(\ pc_2\,?\,box\ ) \; ]\!] \\
\| \; [\![ \; \mathsf{disc}\ box, l_\mathrm{b}, l_\mathrm{c}, \lrcorner l_\mathrm{s}, \ \mathsf{cont}\ x \\
\quad , \ box = 0, x = -1, l_\mathrm{b} = 10, l_\mathrm{c} = 15, \lrcorner l_\mathrm{s} = l_\mathrm{s} \\
\quad \mid \ (\ box = 0 \rightarrow \dot{x} = 0 \parallel box \neq 0 \rightarrow \dot{x} = v_1 \\
\qquad \| *(\ p_1\,?\,box;\ x := 0 \\
\qquad\quad ;\ x \geq l_\mathrm{b} - \lrcorner l_\mathrm{s} \rightarrow s_0 := \text{false} \\
\qquad\quad ;\ x \geq l_\mathrm{c} - \lrcorner l_\mathrm{s} \rightarrow s_1 := \text{true} \\
\qquad\quad ;\ x \geq l_\mathrm{c} \rightarrow\ p_2\,!\,box;\ x, box := -1, 0 \\
\qquad\quad ) \\
\qquad ) \\
\quad ]\!] \\
\| \; [\![ \; \mathsf{disc}\ l_\mathrm{b}, \lrcorner l_\mathrm{s}, box \\
\quad , \ l_\mathrm{b} = 10, \lrcorner l_\mathrm{s} = l_\mathrm{s} \\
\quad \mid *(\ p_2\,?\,box;\ \Delta(l_\mathrm{b} - \lrcorner l_\mathrm{s});\ s_1 := \text{false}\ ) \\
\quad ]\!] \\
\rangle
\end{array}
$$

In the process body of the first instantiation of control process $C$, first the conveyor is switched on ($v_0 := 1$), where $v_0$ is the actuator that determines the velocity of the first conveyor belt. The process subsequently waits until the sensor is off ($\neg s_0$, where $\neg$ means logical not). Initially, the conveyor is empty and the sensor is off. The process then waits until it can synchronize with the preceding control process by executing $pc_0\,?$. This means that a box may enter the conveyor. Subsequently, the process waits until the box has reached the sensor position ($s_0$) so that the sensor is on (value of $s_0$ equals true). Then the conveyor is switched off ($s_0 \rightarrow v_0 := 0$). Subsequently, the control process tries to synchronize with the next control process ($pc_1\,!$). After execution of the synchronization ($pc_1\,!$ in the first control process, simultaneous with $pc_1\,?$ in the second control process), the repetition is re-executed, and the conveyor is switched on again.

In the conveyor process $V$, variable $x$ models the position of the front of the box on the conveyor, and variable *box* stores the identification number of the box. Identification number 0 means that there is no box on the conveyor. In that case, the value of variable $x$ equals and remains -1 ($box = 0 \rightarrow \dot{x} = 0$. When there is a box on the conveyor, it moves with velocity $v$ ($box \neq 0 \rightarrow \dot{x} = v$). The physical representation of variables $l_\mathrm{b}$, $l_\mathrm{c}$, and $l_\mathrm{s}$ is shown in Figure 4.13. The value parameters $l_\mathrm{b}$, $l_\mathrm{c}$, $l_\mathrm{s}$ in the process definition of $V$ are defined as local discrete variables ($l_\mathrm{b}$, $l_\mathrm{c}$, $\lrcorner l_\mathrm{s}$) in the translation of the process instantiation, in accordance with the translation rules defined in Section 2.3.2. Value parameter $l_\mathrm{s}$ is renamed to $\lrcorner l_\mathrm{s}$ by prefixing it with an underscore, because it conflicts with the global variable $l_\mathrm{s}$ that is used in the process instantiations (e.g. $V(s_\mathrm{G}, s_0, v_0, p_0, p_1, 10, 15, l_\mathrm{s})$). The values of the local discrete variables $l_\mathrm{b}$, $l_\mathrm{c}$, and $\lrcorner l_\mathrm{s}$ are defined by initialization predicate $l_\mathrm{b} = 10, l_\mathrm{c} = 15, \lrcorner l_\mathrm{s} = l_\mathrm{s}$.

In the infinite repetition of the process body of the first instantiation of the conveyor process, the process starts by waiting until it can receive a box, and initializes the position of the box to 0 ($p_0\,?\,box;\ x := 0$). When $x = l_\mathrm{b} - \lrcorner l_\mathrm{s}$, the back end of the box has just passed the sensor of the previous "conveyor", which in this case is generator process $G$. Subsequently, the sensor of

the previous conveyor ($s_{\text{prev}}$ in the process definition of $V$, $s_{\text{G}}$ in the first conveyor) is switched off. After that, the conveyor process waits until the box has reached the sensor of the conveyor ($x \geq l_{\text{c}} - \lrcorner l_{\text{s}}$), and the sensor is switched on ($s_0 := \text{true}$). Finally, when the box has reached the end of the conveyor ($x = l_{\text{c}}$), the box is sent to the next conveyor ($p_1 \,!\, box$), and the loop is re-executed. Simultaneously with execution of $p_1 \,!\, box$ by the first conveyor, the second conveyor executes $p_1 \,?\, box$. The box has now crossed the boundary of the two conveyors, and its position is registered by the second conveyor process.

The last conveyor process $V_{\text{E}}$ is always on, meaning that the belt moves with velocity 1. Therefore, the duration of a box on this conveyor is given by $\Delta(l_{\text{b}} - \lrcorner l_{\text{s}})$.

An alternative specification of conveyor process $V$ follows below. In this model, when a box enters a conveyor ($p_{\text{i}} \,?\, box$), a new continuous variable $x$ with initial value 0 is created by means of the scope operator $[\![\ \mathsf{icont}\ x,\ x = 0 \mid \cdots ]\!]$. The position of the box is defined by equation $\dot{x} = v$ until $x \geq l_{\text{b}} - l_{\text{s}}$. Then, sensor $s_{\text{prev}}$ is switched off ($s_{\text{prev}} := \text{false}$), which causes the alternative composition ($\dot{x} = v \,[]\, x \geq l_{\text{b}} - l_{\text{s}} \rightarrow s_{\text{prev}} := \text{false}$) to terminate. The difference between this conveyor model and the previous model, is that in the previous model, absence of a box was modeled by means of predicate $\dot{x} = 0$, whereas in the model below, absence of the conveyor means that variable $x$, which models the position of the front of the conveyor, does not exist.

$$
\begin{array}{l}
V(\mathsf{ext}\ s_{\text{prev}}, s, v,\ \mathsf{chan}\ p_{\text{i}}, p_{\text{o}},\ \mathsf{val}\ l_{\text{b}}, l_{\text{c}}, l_{\text{s}}) = \\
[\![\ \mathsf{disc}\ box \\
\mid *(\ p_{\text{i}} \,?\, box \\
\quad ;\ [\![\ \mathsf{cont}\ x,\ x = 0 \\
\qquad \mid (\dot{x} = v \,[]\, x \geq l_{\text{b}} - l_{\text{s}} \rightarrow s_{\text{prev}} := \text{false}) \\
\qquad ;\ (\dot{x} = v \,[]\, x \geq l_{\text{c}} - l_{\text{s}} \rightarrow s := \text{true}) \\
\qquad ;\ (\dot{x} = v \,[]\, x \geq l_{\text{c}} \rightarrow p_{\text{o}} \,!\, box) \\
\qquad ]\!] \\
\quad ) \\
]\!]
\end{array}
$$

# Chapter 5

# Translations of other formalisms to Chi

## 5.1 Translation of a hybrid automaton to Chi

In this section, the hybrid automaton model of [27] is related to the $\chi$ formalism.

### 5.1.1 Description Hybrid Automaton

A hybrid automaton [27] consists of the following components:

- A finite set of (real-valued) variables $X = \{x_1, \ldots, x_n\}$, the set $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ which denotes the first derivatives of these variables, and the set $X' = \{x'_1, \ldots, x'_n\}$ which denotes the primed variables that represent values at the conclusion of a discrete change.
- A finite directed multi-graph $(V, E)$, where $V$ denotes a set of vertices (control modes) and $E$ denotes a set of edges (control switches).
- Three vertex labeling functions init, inv, and flow that assign to each control mode $v \in V$ a predicate for initial, invariant and flow conditions, respectively. The free variables of the initial and invariant predicates are from $X$. The free variables of the flow predicates are from $X \cup \dot{X}$.
- An edge labeling function jump, that assigns to each edge $e \in E$, a jump condition which is a predicate whose free variables are from $X \cup X'$.
- A finite set $\Sigma$ of events, and an edge labeling function event : $E \to \Sigma$ that assigns to each edge an event.

In order to translate a hybrid automaton to $\chi$, two additional functions are defined on a hybrid automaton: function edges $\in V \to \mathcal{P}(E)$ returns a set of outgoing edges for a location, and function target $\in E \to V$ returns the target vertex of an edge. Furthermore, function $\mathcal{T}$ translates a jump predicate to the predicate $r$ of a $\chi$ action predicate ($W : r \gg l_a$) by renaming variables occurring without a prime in a jump predicate to variables with superscript '$^-$', and renaming variables occurring with a prime '$'$' to variables without the prime. E.g. $\mathcal{T}(x' = 2x + y \wedge x \geq 0 \wedge y' = y)$ becomes $x = 2x^- + y^- \wedge x^- \geq 0 \wedge y = y^-$. In the latter expression, $x^-$ and $y^-$ refer to the values of $x$ and $y$, respectively, before the discrete jump, and $x$ and $y$ refer to the value of variables $x$ and $y$ after the discrete jump. The class of hybrid automata to be translated to $\chi$ is restricted to the hybrid automata without initial time non-determinism. In this section, we consider hybrid automata where the initial condition of all but one control modes equals false. The one control mode with the initial condition that may be not equal to false is called the *initial* control mode.

Furthermore, it should be possible to rewrite each flow predicate into one of the following forms: $\dot{\mathbf{x}} = f(\mathbf{x})$, $\dot{\mathbf{x}} \in f(\mathbf{x})$ or the predicate true. This means that we do not consider flow predicates such as false, or $\dot{x} = 0 \wedge \dot{x} = 1$.

### 5.1.2 Translation Scheme

Consider a hybrid automaton model which belongs to the class of automata as defined in the previous section, with $n$ variables ($X = \{x_1, \ldots, x_n\}$), $k$ control modes ($V = \{v_1, \ldots, v_k\}$), and one initial control mode $v_1$. The translation to a corresponding $\chi$ specification is defined as follows:

$$
\begin{array}{l}
\langle\ \mathsf{cont}\ x_1, \ldots, x_n \\
,\ \mathrm{init}(v_1) \\
,\ v_1 \mapsto \mathrm{flow}(v_1) \wedge \mathrm{inv}(v_1)\ [] \\
\qquad ([]_{e:e\in \mathrm{edges}(v_1)}\ [\emptyset : \mathcal{T}(\mathrm{jump}(e)) \gg \mathrm{event}(e)];\ \mathrm{target}(e)) \\
\quad \vdots \\
,\ v_k \mapsto \mathrm{flow}(v_k) \wedge \mathrm{inv}(v_k)\ [] \\
\qquad ([]_{e:e\in \mathrm{edges}(v_k)}\ [\emptyset : \mathcal{T}(\mathrm{jump}(e)) \gg \mathrm{event}(e)];\ \mathrm{target}(e)) \\
\\
|\ (\mathsf{jump}\ x_1, \ldots, x_n \mid v_1) \\
\rangle
\end{array}
$$

The variables $x_1, \ldots, x_n$ are declared as continuous variables. These variables are initialized by means of initialization predicate $\mathrm{init}(v_1)$. By means of the jump enabling operator process term ($\mathsf{jump}\ x_1, \ldots, x_n \mid \ldots$), all variables become jumping. I.e., in principle they may change arbitrarily during action transitions.

A vertex $v_i$ of the hybrid automaton model is translated using a corresponding recursion variable $v_i$ in the $\chi$ model. The process term associated with this recursion variable consists of the

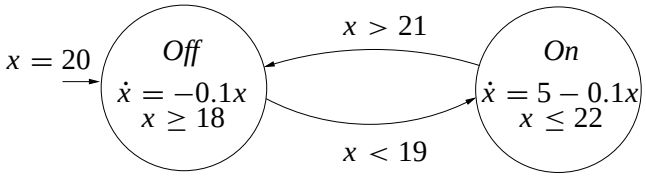


Figure 5.1: A hybrid automaton model of a thermostat

alternative composition of the process term describing the continuous behavior of the vertex, and the alternative compositions of all individual process terms of the outgoing edges of this vertex. Below, these process terms are explained in more detail.

The continuous behavior of a vertex $v_i$ is translated to a delay predicate in $\chi$, consisting of the conjunction of the flow predicate and the invariant of the vertex. For each outgoing edge, the jump predicate of that edge is translated to an action predicate labelled with the event label of the edge ($\emptyset : \mathcal{T}(\text{jump}(e)) \gg \text{event}(e)$). Since all variables are allowed to jump, the set $W$ of the action predicate equals the empty set. The semantics of a hybrid automaton is such that when a guard of an edge is enabled, the transition via this edge can be taken, but it is not required to take this transition. Therefore, the action predicate associated with the edge is made delayable using the delay enabling operator [ ]. After the transition, the behavior is specified by the recursion variable associated with the target vertex (target($e$)).

Note that for set $E = \{e_1, \ldots, e_k\}$, notation $[]_{e:e\in E}\ [\emptyset : \mathcal{T}(\text{jump}(e)) \gg \text{event}(e)];\ \text{target}(e)$, denotes the process term $[\emptyset : \mathcal{T}(\text{jump}(e_1)) \gg \text{event}(e_1)];\ \text{target}(e_1)\ [] \ldots []\ [\emptyset : \mathcal{T}(\text{jump}(e_k)) \gg \text{event}(e_k)];\ \text{target}(e_k)$.

This straightforward translation of a hybrid automaton to a $\chi$ model shows that $\chi$ is expressive enough to model phenomena that are usually studied by means of a hybrid automaton. The translation scheme is illustrated by means of an example in the next section.

### 5.1.3 A thermostat

This example shows the translation of a hybrid automaton model of a thermostat to $\chi$. The hybrid automaton is shown in Figure 5.1. Variable $x$ represents the temperature. The control modes are *On* and *Off*. Initially, the temperature equals 20 degrees, and the heater is off (control mode *Off*). The temperature falls according to the flow condition $\dot{x} = -0.1x$. According to the jump condition $x < 19$, the heater may go on as soon as the temperature falls below 19 degrees. The invariant condition $x \geq 18$ ensures that at the latest the heater will go on when the temperature equals 18 degrees. In the control mode *On*, the heater is on, and the temperature rises according to the flow condition $\dot{x} = 5 - 0.1x$. When the temperature rises above 21 degrees, the heater may turn off. Due to the invariant condition $x \leq 22$, at the latest the heater will turn off when the temperature equals 22 degrees.

Figure 5.1 is taken from [27], where the usual informal notation is used: events on the edges

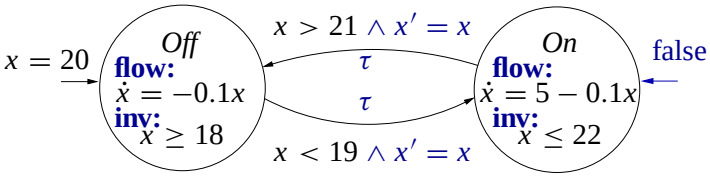


Figure 5.2: Complete hybrid automaton model of a thermostat

are ignored, and the initial and jump conditions are incomplete. In particular, in Figure 5.1 both edges should have an event label, the initial condition of mode *On* equals false, and the jump conditions of the edges should have been $x < 19 \wedge x' = x$ and $x > 21 \wedge x' = x$, respectively. The complete, formal hybrid automata model of the thermostat is shown in Figure 5.2. Using the translation scheme, this model is translated to $\chi$, which results in the following $\chi$ specification:

$\langle$ cont $x$
, $x = 20$
, *Off* $\mapsto$ $\dot{x} = -0.1x \wedge x \geq 18$ [] [$\emptyset : x < 19 \wedge x = x^- \gg \tau$]; *On*
, *On* $\mapsto$ $\dot{x} = 5 - 0.1x \wedge x \leq 22$ [] [$\emptyset : x > 21 \wedge x = x^- \gg \tau$]; *Off*
| (jump $x$ | *Off*)
$\rangle$ .

Since the value of variable $x$ does not change during action transitions, the model can be simplified to

$\langle$ cont $x$
, $x = 20$
, *Off* $\mapsto$ $\dot{x} = -0.1x \wedge x \geq 18$ [] [$\emptyset : x < 19 \gg \tau$]; *On*
, *On* $\mapsto$ $\dot{x} = 5 - 0.1x \wedge x \leq 22$ [] [$\emptyset : x > 21 \gg \tau$]; *Off*
| *Off*
$\rangle$ .

## 5.2 Translations of piecewise affine systems to Chi

In this section, two general translation schemes are given. One scheme defines the translation of continuous-time piecewise affine systems to a $\chi$ specification. The other scheme defines a translation of discrete-time piecewise affine systems to $\chi$.

### 5.2.1 Continuous-time PWA

Continuous-time piecewise affine systems are described by $N$ systems of affine differential equations

$$\begin{array}{rcl} \dot{x}(t) & = & A_i x(t) + B_i u(t) + f_i \\ y(t) & = & C_i x(t) + D_i u(t) + g_i \end{array} \qquad \text{for} \quad \left[ \begin{array}{c} x(t) \\ u(t) \end{array} \right] \in \Omega_i \ ,$$

where $i$ ($i = 1, \ldots, N$) is the number of the mode. Each mode $i$ is defined in a region $\Omega_i$, which is a convex polyhedron, given by a finite number of linear inequalities, in the input/state space. Here, $u(t) \in \mathbb{R}^m$, $x(t) \in \mathbb{R}^n$, and $y(t) \in \mathbb{R}^l$ denote the input, state and output, respectively, at time $t$. Furthermore, $f_i$, and $g_i$ denote constants. In each mode, the trajectories of the state variables $x$ are continuous functions of time. The trajectories of the input/output variables in a mode may be discontinuous functions of time.

Continuous-time PWA systems using the Caratheodory solution concept, can be translated to $\chi$ as follows:

$$\begin{array}{l} \langle\ \mathsf{cont}\ x\,,\ \mathsf{alg}\ y \\ ,\ x = x_0 \\ |\quad (\Omega_1 \Rightarrow \dot{x} = A_1 x + B_1 u + f_1,\ y = C_1 x + D_1 u + g_1) \\ \quad \wedge \\ \quad \vdots \\ \quad \wedge\ (\Omega_N \Rightarrow \dot{x} = A_N x + B_N u + f_N,\ y = C_N y + D_N u + g_N) \\ \rangle \end{array}$$

The state variables $x$ are modeled by means of (non-jumping) continuous variables, with initial value $x_0$. The output variables $y$ are modeled by means of algebraic variables. The behavior of $u$ is not specified, as in the original PWA model. In the $\chi$ model, $u$ could denote a function of time, or $u$ could be defined as an algebraic variable, and additional equations specifying the behavior of $u$ could be added. The behavior associated to a mode $i$ is described by means of a delay predicate $(\Omega_i \Rightarrow \dot{x} = A_i x + B_i u + f_i\,,\ y = C_i x + D_i u + g_i)$.

### 5.2.2 Discrete-time PWA

Discrete-time PWA systems are described by

$$\begin{array}{rcl} x(k+1) & = & A_i x(k) + B_i u(k) + f_i \\ y(k) & = & C_i x(k) + D_i u(k) + g_i \end{array} \qquad \text{for} \quad \left[ \begin{array}{c} x(k) \\ u(k) \end{array} \right] \in \Omega_i,$$

$i = 1, \ldots, N$. Here, $u(k) \in \mathbb{R}^m$, $x(k) \in \mathbb{R}^n$, and $y(k) \in \mathbb{R}^l$ denote the input, state and output, respectively, at the $k$'th time-point. Discrete-time PWA systems can be translated to $\chi$ as follows:

$$\begin{array}{l} \langle\ \mathsf{disc}\ x, y \\ ,\ x = x_0, k = 0 \end{array}$$

$$
\begin{array}{l}
\langle \\
\quad | \ *( \ ( \ \Omega_1 \rightarrow x, y := A_1 x + B_1 u.k + f_1, C_1 x + D_1 u.k + g_1 \\
\qquad [] \\
\qquad \vdots \\
\qquad [] \ \Omega_N \rightarrow x, y := A_N x + B_N u.k + f_N, C_N y + D_N u.k + g_N \\
\qquad ) \\
\quad ; \ k := k + 1 \\
\quad ) \\
\rangle
\end{array}
$$

The state variables $x$ are modeled as discrete variables, and are initialized to $x_0$. The output variables $y$ are also modeled by means of discrete variables. We assume $u$ to denote an array of points, such that $u.i$ denotes the value of $u$ at the $i$'th time-point. In the repetition $*(\quad)$, the state and output variables are assigned new values according to one of the modes. Subsequently, $k$ is increased by one. The behavior associated to a mode is described by means of a multiple assignment $x, y := A_i x + B_i u.k + f_i, C_i x + D_i u.k + g_i$. The alternative composition of the behavior of the modes allows the state and output variables to be assigned new values according to the mode for which the corresponding guard ($\Omega_i$) holds.

# Chapter 6

# Validation of the semantics

In this section, first we consider the well-definedness of the semantics in Section 6.1. Then, in Section 6.2, some properties of the Chi semantics are given.

In Section 6.3, a notion of equivalence is defined, called stateless bisimilarity [35], which is similar to the well-known notion of bisimilarity [38, 34]. It is also shown that this relation is an equivalence and a congruence for all $\chi$ operators. Some useful properties of closed $\chi$ process terms are given in Section 6.4. Many of these properties express intuitions about the meaning of the $\chi$ operators such as the commutativity and associativity of the alternative composition and the parallel composition operator. Other properties are introduced for the purpose of simplifying $\chi$ models. Both the examples treated in the previous section and the properties treated in this section add to the level of confidence one has with respect to the 'correctness' of the semantics.

## 6.1 Well-definedness of the semantics

In the term deduction system negative premises are used in Rule 36 of the urgency communication operator. As a consequence it is not obvious at first sight whether the term deduction system defines a unique transition system for each closed process term. Well-definedness of the term deduction system can be obtained by providing a *stratification* [9]. The mapping that associates with every positive action transition and positive consistency predicate the value 0 and with every positive time transition the value 1, turns out to be a stratification.

## 6.2 Properties of the semantics

In this section, some useful properties about the semantics of $\chi$ are introduced that can be applied in the remainder of the paper (especially in the proofs of the properties in Section 6.4).

With the current set of deduction rules for the semantics of $\chi$, the left-hand and right-hand extended valuation restricted to the model variables are always the same as the initial and resulting

valuation of an action transition for these model variables respectively. A similar reasoning applies to the first and last valuation of a trajectory on a time transition and the initial and resulting valuation respectively. Also note that the environment is never changed in a transition, and the extended valuation that proves consistency restricted to the model variables is the same as the initial valuation.

The following lemma captures these facts.

**Lemma 1** *Let $p$ and $p'$ be closed process terms, $\sigma, \sigma'$ be valuations, $\xi, \xi'$ be extended valuations, $E$ and $E'$ be environments, $a$ be an action, $\rho$ be a trajectory, and $t \in T$. Then*

$$\begin{aligned}
\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle _, \sigma', E'\rangle \quad &\Rightarrow \quad \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma' \\
&\quad \wedge E = E', \\
\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle \quad &\Rightarrow \quad \mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma' \\
&\quad \wedge E = E', \\
\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \quad &\Rightarrow \quad \xi_\sigma = \sigma.
\end{aligned}$$

*where $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\surd}{p'}, \sigma', E'\rangle$ for some $p'$.*

*Proof.* See Appendix A.1. ⊠

For $\chi$ processes that can perform action or time transitions, the result of such a transition (including zero-time transitions) is always a consistent process (e.g., the consistency predicate holds for the resulting process).

**Lemma 2** *Let $p$ and $p'$ be closed process terms, $\sigma$ and $\sigma'$ be valuations, $E$ and $E'$ be environments, $\xi$ and $\xi'$ be extended valuations and $a$ be an action. Then*

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \quad \Rightarrow \quad \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow},$$

*where $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ is an abbreviation for $\exists_{p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\surd}{p'}, \sigma', E'\rangle$.*

*Proof.* See Appendix A.2. ⊠

**Lemma 3** *Let $p$ and $p'$ be closed process terms, $\sigma$ and $\sigma'$ be valuations, $E$ and $E'$ be environments, $t \in T$, and $\rho$ be a trajectory. Then,*

$$\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \quad \Rightarrow \quad \langle p, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow},$$

*where $\langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto}$ is an abbreviation for $\exists_{p',\sigma',E'} \langle p, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$.*

*Proof.* See Appendix A.3.

⊠

The following lemma shows that any variation in the set of jumping variables in the environment of a consistent $\chi$ process has no effect on the consistency transition.

**Lemma 4** *Let $p$ be a closed process term, $\sigma$ be a valuation, $C$, $J$, $W$, $L$ be sets of various classes of $\chi$ variables such that $J$ and $W \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, $R$ be a recursion definition, and $\xi$ be an extended valuation. Then*

$$\langle p, \sigma, (C, J, L, R)\rangle \overset{\xi}{\rightsquigarrow} \quad \Leftrightarrow \quad \langle p, \sigma, (C, J \cup W, L, R)\rangle \overset{\xi}{\rightsquigarrow}.$$

*Proof.* The proof is trivial. The domain of the extended valuation $\xi$ is given by $\mathrm{dom}(\sigma) \cup \dot{C} \cup L$ for all $\chi$ consistency transition rules. Hence, any variation in the set of jumping variables in the environment of a consistent $\chi$ process is irrelevant for the consistency transition. ⊠

## 6.3 Stateless bisimilarity

Two closed $\chi$ process terms are considered equivalent if they have the same behavior (in the bisimulation sense) in case both are considered from the same initial valuation of model variables and the same environment. We also assume that the initial valuation contains at least the free occurrences of variables in the two closed $\chi$ process terms being equivalent.

**Definition 1 (Stateless bisimilarity)** *A stateless bisimulation relation on closed process terms is a relation $R \subseteq P \times P$ such that $\forall (p, q) \in R$, the following holds:*

1. $\forall \sigma, E, \xi, a, \xi', \sigma', E' : \langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E'\rangle$
   $\Leftrightarrow \langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E'\rangle,$
2. $\forall \sigma, E, \xi, a, \xi', p', \sigma', E' : \langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle p', \sigma', E'\rangle$
   $\Rightarrow \exists q' : \langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle q', \sigma', E'\rangle \wedge (p', q') \in R,$
3. $\forall \sigma, E, \xi, a, \xi', q', \sigma', E' : \langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle q', \sigma', E'\rangle$
   $\Rightarrow \exists p' : \langle p, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle p', \sigma', E'\rangle \wedge (p', q') \in R,$
4. $\forall \sigma, E, t, \rho, p', \sigma', E' : \langle p, \sigma, E\rangle \overset{t, \rho}{\longmapsto} \langle p', \sigma', E'\rangle$
   $\Rightarrow \exists q' : \langle q, \sigma, E\rangle \overset{t, \rho}{\longmapsto} \langle q', \sigma', E'\rangle \wedge (p', q') \in R,$

5. $\forall \sigma, E, t, \rho, q', \sigma', E' : \langle q, \sigma, E\rangle \stackrel{t,\rho}{\longmapsto} \langle q', \sigma', E'\rangle$
   $\Rightarrow \exists p' : \langle p, \sigma, E\rangle \stackrel{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle \wedge (p', q') \in R$,

6. $\forall \sigma, E, \xi : \langle p, \sigma, E\rangle \stackrel{\xi}{\rightsquigarrow} \Leftrightarrow \langle q, \sigma, E\rangle \stackrel{\xi}{\rightsquigarrow}$ .

*Two closed process terms $p$ and $q$ are stateless bisimilar, denoted by $p \underline{\leftrightarrow} q$, if there exists a stateless bisimulation relation $R$ such that $(p, q) \in R$.*

Note that in the above definition of stateless bisimilarity it is possible for the left-hand and right-hand extended valuation of model variables as displayed on an action transition, and the valuation of model variables of the state before performing the action transition and the state reached by performing the action transition to be different. Similarly, the valuation of model variables before and after a time transition can be different from the initial and end-point of a trajectory.

As a consequence of Lemma 1, the definition of stateless bisimilarity can be simplified considerably. Yet, with in mind future extensions of the $\chi$ language, it might well be the case that these properties of the semantics are lost. Since we would prefer not to redo all the coming proofs (in such a future), this presentation was chosen.

Stateless bisimilarity is proved to be a congruence with respect to all $\chi$ operators. As a consequence, algebraic reasoning is facilitated, since it is allowed to replace equals by equals in any context.

**Theorem 1 (Congruence)** *Stateless bisimilarity is a congruence with respect to all $\chi$ operators.*

*Proof.* The deduction rules of the $\chi$ language, satisfy the *process-tyft* format of [35]. Therefore, stateless bisimilarity is a congruence. ⊠

## 6.4 Properties of the Chi operators

In this section, some properties of the operators of $\chi$ that hold with respect to stateless bisimilarity are discussed. Most of these correspond well with our intuitions, and hence this can be considered as an additional validation of the semantics. It is not our intention to provide a complete list of such properties (complete in the sense that every equivalence between closed process terms is derivable from those properties).

**Proposition 1 (Signal emission operator)** *The following properties hold for all closed process terms $p \in P$ and predicates $u, u'$:*

| | | |
|---|---|---|
| $\text{true} \curvearrowright p$ | $\underline{\leftrightarrow}$ | $p$ |
| $\text{false} \curvearrowright p$ | $\underline{\leftrightarrow}$ | $\bot$ |
| $u \curvearrowright u$ | $\underline{\leftrightarrow}$ | $u$ |
| $u \curvearrowright (u' \curvearrowright p)$ | $\underline{\leftrightarrow}$ | $(u \wedge u') \curvearrowright p$ |

If a true predicate is emitted, the process term is simply executed. If falsity holds initially, the process term is inconsistent. There is no effect if a predicate $u$ is emitted to itself. A concatenation of signal emissions leads to a signal emission with conjunction of predicates.

**Proposition 2 (Alternative composition)** *The following properties hold for all closed process terms $p, q, r \in P$:*

| | | |
|---|---|---|
| $p \,[]\, p$ | $\underline{\leftrightarrow}$ | $p$ |
| $p \,[]\, q$ | $\underline{\leftrightarrow}$ | $q \,[]\, p$ |
| $(p \,[]\, q) \,[]\, r$ | $\underline{\leftrightarrow}$ | $p \,[]\, (q \,[]\, r)$ |

The alternative composition is idempotent, commutative and associative. The property $p \,[]\, \delta \underline{\leftrightarrow} p$ does not hold. Consider, for example $p = \text{true}$. Then $p \,[]\, \delta$ cannot perform any time transitions, while $p$ can perform arbitrary time transitions. Property $p \,[]\, \delta \underline{\leftrightarrow} \delta$ does not hold either. Consider, for example $p = \text{skip}$. Then $p \,[]\, \delta$ can perform a $\tau$ transition, while $\delta$ cannot.

**Proposition 3 (Guard operator)** *The following properties hold for all closed process terms $p \in P$ and guard $b$:*

| | | |
|---|---|---|
| $\text{true} \rightarrow p$ | $\underline{\leftrightarrow}$ | $p$ |
| $\text{false} \rightarrow p$ | $\underline{\leftrightarrow}$ | true |
| $b \rightarrow \bot$ | $\underline{\leftrightarrow}$ | $\neg b$ |
| $b \rightarrow (p \,[]\, q)$ | $\underline{\leftrightarrow}$ | $b \rightarrow p \,[]\, b \rightarrow q$ |

If a process term is guarded by a true predicate, the process term is simply executed. In case a process term is guarded by a false predicate, process term $\text{false} \rightarrow p$ can perform any time transition, hence equals a true predicate. An inconsistent process term that is guarded by any guard is equivalent to the negation of the guard. The guard distributes over the alternative composition operator.

**Proposition 4 (Sequential composition)** *The following properties hold for all closed process terms $p, q, r \in P$ and guard $b$:*

| | | |
|---|---|---|
| $\delta ; p$ | $\underline{\leftrightarrow}$ | $\delta$ |
| $(p ; q) ; r$ | $\underline{\leftrightarrow}$ | $p ; (q ; r)$ |
| $(p \,[]\, q) ; r$ | $\underline{\leftrightarrow}$ | $p ; r \,[]\, q ; r$ |
| $b \rightarrow (p ; q)$ | $\underline{\leftrightarrow}$ | $(b \rightarrow p) ; q$ |

A deadlock process term followed by some other process terms is equivalent to the deadlock process term itself since the deadlock process term does not terminate successfully, i.e., deadlock is a left-zero element for sequential composition. Sequential composition is associative and alternative composition distributes over sequential composition from the left. A guard distributes to the left argument of a sequential composition.

**Proposition 5 (Parallel composition)** *The following properties hold for all closed process terms* $p, q, r \in P$ *:*

| | | |
|---|---|---|
| $p \parallel q$ | $\underline{\leftrightarrow}$ | $q \parallel p$ |
| $(p \parallel q) \parallel r$ | $\underline{\leftrightarrow}$ | $p \parallel (q \parallel r)$ |

Parallel composition is commutative and associative.

**Proposition 6 (Action encapsulation operator)** *The following properties hold for all closed process terms* $p \in P$*, and sets of actions* $\mathcal{A}$*,* $\mathcal{A}'$*:*

| | | |
|---|---|---|
| $\partial_{\emptyset}(p)$ | $\underline{\leftrightarrow}$ | $p$ |
| $\partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p))$ | $\underline{\leftrightarrow}$ | $\partial_{\mathcal{A}\cup\mathcal{A}'}(p)$ |

If there are no actions to be encapsulated, the application of the action encapsulation operator to a process term $p$ has no effect. Multiple applications of the action encapsulation operator are equivalent to a single application where all the actions to be encapsulated are combined using union of sets of actions.

**Proposition 7 (Inconsistent process)** *The following properties hold for all closed process terms* $p \in P$ *and predicate* $u$*:*

| | | |
|---|---|---|
| $u \curvearrowright \bot$ | $\underline{\leftrightarrow}$ | $\bot$ |
| $p \,[]\, \bot$ | $\underline{\leftrightarrow}$ | $\bot$ |
| $p \parallel \bot$ | $\underline{\leftrightarrow}$ | $\bot$ |
| $\partial_{\mathcal{A}}(\bot)$ | $\underline{\leftrightarrow}$ | $\bot$ |
| $\bot;\ p$ | $\underline{\leftrightarrow}$ | $\bot$ |
| skip; $\bot$ | $\underline{\leftrightarrow}$ | $\delta$ |
| $\bot$ | $\underline{\leftrightarrow}$ | false |

The inconsistent process term is a zero element for signal emission operator, alternative composition, parallel composition and action encapsulation operator. It is also a left-zero element for sequential composition. Going on as $\bot$ after performing an action transition, for example skip, is impossible. Since $\bot$ and false predicate cannot perform any transition, both process terms are equivalent.

# Chapter 7

# Related work

The $\chi$ language is a hybrid process algebra, and is thus related to the other hybrid process algebras: HyPA [17], process algebra for hybrid systems $ACP_{hs}^{srt}$ [11], the $\phi$-Calculus [40], the hybrid formalisms based on CSP [30, 15], and the process algebra [51].

The latter three process algebras [30, 15, 51] differ from $\chi$ in that they do not have shared variables. Shared variables are essential for modular specification of continuous and hybrid systems. The two CSP based formalisms also differ from the other process algebras in that they use a denotational semantics instead of an operational semantics. An operational semantics is generally believed to be more intuitive and easier to understand than a denotational semantics [1].

The $\phi$-Calculus differs from the other process algebras in that continuous behavior is not defined by means of predicates in process terms. Instead, continuous behavior is defined by means of an environment. Process terms operate on the environment to update initial values, vector-fields, and algebraic constraints. In this way, the $\phi$-Calculus can deal with dynamically reconfigurable processes. The resulting differential equations are required to be autonomous. This limits the specification of continuous systems, using the $\phi$-Calculus, to that of ordinary differential equations (ODEs).

The relation between $\chi$, hybrid automata [2], HyPA and the process algebra for hybrid systems $ACP_{hs}^{srt}$ is summarized below. Where HyPA is a conservative extension of ACP [10], and $ACP_{hs}^{srt}$ is an conservative extension of a combination of the process algebra with continuous relative timing [8] and the process algebra with propositional signals [7], hybrid $\chi$ is not an extension of any existing process algebra. Hybrid $\chi$ does aim to be a conservative extension of timed $\chi$, which has been developed simultaneously with hybrid $\chi$. Therefore, time is incorporated by means of a predefined variable time.

The integration between the DC and CS world views in $\chi$ was inspired by HyPA. Also, the use of delay predicates as atomic process term was inspired by HyPA. Hybrid $\chi$ and $ACP_{hs}^{srt}$ were both strongly influenced by hybrid automata. A difference is that where in hybrid $\chi$ the passage of time cannot make a choice in alternative composition, in $ACP_{hs}^{srt}$ the passage of time

can enforce such a choice. In HyPA, the passage of time will always make a choice between the operands of the choice operator. This corresponds to the initial behavior of a hybrid automaton: depending on the initial state, a non-deterministic choice can be made for the first location where continuous behavior or discrete behavior may take place. After this first choice, a hybrid automaton cannot change location as a result of time passing, nor can outgoing edges disappear as a result of time passing.

$\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, $\chi$, and hybrid automata such as defined in [5, 27] share the 'consistent equation semantics'. For a hybrid automaton, the invariant of the current location should hold in the current state, and transitions to a new state and new location are allowed only if the invariant of the new location holds in the new state. Correspondingly, in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and $\chi$, the equations / delay predicates of the process term should be consistent with the current state, and transitions to a new process term are allowed only if the equations / delay predicates of the new process term are consistent with the new state. The hybrid automaton defined in [2] has a different semantics in that it allows transitions to a new location only if the invariant of the *current* location holds in the current state and in the new state.

A difference between $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and $\chi$ in this respect is that where in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ the dotted variables (derivatives) are part of the state (valuation), in $\chi$ they are not. This is because in $\chi$, the valuation together with the process term and the environment represent all that is needed to be able to predict future behavior. The values of the dotted variables are irrelevant for the prediction of future behavior. For the same reason, algebraic variables are not part of the valuation in $\chi$. Their values are determined completely by the process term. The signal emission operator in $\chi$ is inspired by the signal emission operator from the process algebra with relative timing [8], which is also used in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$.

There are several other differences between $\chi$, hybrid automata, $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, and HyPA:

- Where hybrid automata and $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ use continuous variables that are allowed to jump arbitrarily in an action transition with a true reset predicate, and HyPA uses continuous variables that are not allowed to jump in an action transition, unless explicitly specified, $\chi$ uses both classes of continuous variables. Furthermore, $\chi$ adds discrete and algebraic variables.
- Where HyPA does not specify a solution concept for algebraic differential equations, the $\chi$ semantics rigorously defines a solution concept, that also deals with discontinuous trajectories. Of course, since the solution concept of HyPA is a parameter of the semantics, it could use the solution concept defined in $\chi$.
- Where $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and HyPA use non-delayable guards, $\chi$ uses delayable guards.
- Where $\chi$ and some versions of hybrid automata, such as [5], integrate urgent (non-delayable) and non-urgent (delayable) actions in such a way that delaying is not possible when an urgent action can be executed, this feature is unavailable in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and HyPA.
- The modeling extensions present in $\chi$ are unavailable in the other three formalisms, apart from the delay operator, which is also available in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. However, in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, the expression defining the amount of delay cannot contain variables. Furthermore, the scope

operators, and process definition and instantiation process terms for complex system specification are available only in $\chi$, apart from the variable scope operator which is also available in HyPA [49].

Other formalisms for hybrid system specification are hybrid Petri nets [19, 22], hybrid I/O automata [33], and formalisms based on hybrid automata such as Charon [3] and Masaccio [26]. There are many differences and similarities between $\chi$ and these other formalisms. The main difference, however, between $\chi$ and other formalisms is that we consider $\chi$ to be overall better suited to modeling. This may mean that certain phenomena can be modeled in $\chi$ whereas they cannot be modeled in another formalism, or that certain phenomena can be modeled more concisely or more intuitively in $\chi$. This is a result of the following aspects:

1. The integration between the DC and CS world views as explained in Section 1. In this respect $\chi$ differs from the other formalisms mentioned above, apart from HyPA.

2. The combination of concise and intuitive language primitives, well suited to modeling, with a straightforward semantics, well suited to verification. This was in fact the biggest challenge in the design of $\chi$. After numerous attempts to define the language primitives with associated syntax and semantics, it appeared that either the language was well suited to modeling, but with complex semantics, unsuited to verification; or the semantics was straightforward and elegant, but at the same time the language was cumbersome for modeling. The reason for this apparent contradiction is that the requirements for language primitives for verification and the requirements for language primitives for modeling are not the same.

3. The relatively large number of operators dedicated to modeling of discrete-event behavior: This makes it easy to abstract from continuous behavior and specify timed discrete-event models, without any continuous variables and without differential (algebraic) equations. In this respect, $\chi$ has much in common with the hybrid formalisms based on CSP [30, 15], and with $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$.

4. Process instantiation, based on the modeling scope operator: this enables hierarchical composition of processes. It also provides encapsulation and data hiding, and it enables re-use of processes: parameterized processes can be defined once and instantiated many times with the same or different parameters. In this respect, the $\chi$ language is related to Charon and Masaccio, which allow components to be defined and instantiated. The $\chi$ formalism, being a process algebra, does not only allow parallel composition (as Charon and Masaccio) and sequential composition (as Masaccio), but also provides alternative composition.

   Local variables, variable abstraction and/or action abstraction are present also in other formalisms. Hybrid I/O automata [33] define both action abstraction and variable abstraction, which are referred to as hiding of external actions and external variables. Hybrid (I/O) automata, however, need to be ‘compatible’ to allow parallel composition. Hybrid I/O automata, for example, require disjointness of the internal variables of the automata in parallel composition.

In $\chi$, the concepts of variable abstraction and channel abstraction (comparable with action abstraction in other formalisms) are integrated in the modeling scope operator, which also provides a local scope for variables, channels, and recursion definitions. In this respect, the $\chi$ scope operator is a high level modeling primitive unavailable in the other hybrid formalisms. Also, there are no compatibility restrictions on processes for parallel composition. Modular composition of processes is further supported by means of different interaction mechanisms. Processes can interact in three different ways:

- By means of shared variables, which is the main interaction mechanism for continuous-time processes consisting of systems of differential algebraic equations. Interaction between processes in Charon and Masaccio also takes place by means of shared variables. Synchronization by means of actions is, however, not supported.
- By means of channel based 'handshake synchronization'. It is comparable to actions in (hybrid) (I/O) automata and actions in ACP-based process algebras. A difference is that actions can be used to express synchronization between two or more processes. The synchronization mechanism used in $\chi$ is CSP [28] based. A channel can be shared by any number of processes, but synchronization always occurs on a point-to-point basis, so between exactly two processes. Another difference is that the interaction mechanism in $\chi$ also allows synchronous *communication*, as explained below, whereas actions are strictly used for synchronization.
- By means of synchronous communication, which is the CSP interaction mechanism that combines synchronization with data-transfer, as also used in [30, 15].

# Chapter 8

# Conclusions and future work

The hybrid $\chi$ formalism differs considerably from other formalisms. On the one hand, it supports the dynamics and control way of hybrid systems modeling by means of discontinuous functions and/or switched equation systems, possibly leading to discontinuous trajectories. On the other hand, it supports the computer science way of hybrid systems modeling, where actions are used to model discontinuities. With respect to the computer science way of modeling, the $\chi$ formalism is heavily influenced by hybrid automata. The two formalisms both have a choice mechanism where, apart from initialization in a hybrid automaton, the passage of time cannot result in changes other than in the values of variables. Hybrid $\chi$ also shares the consistency concept with many hybrid automata: state changes in $\chi$ need to be consistent with delay predicates, which include the invariant and flow clauses of hybrid automata.

The $\chi$ language furthermore combines ease of modeling with a straightforward, formal semantics. Ease of modeling is ensured by means of different classes of variables, such as discrete, non-jumping continuous, jumping continuous and algebraic variables; by means of its delayable guard that ensures that the guard always holds when the first action of the guarded process term occurs; by means of its integration of urgent and non-urgent actions on the one hand, and urgent and non-urgent channels on the other hand; by means of allowing the modeling of differential algebraic equations as a process term as in mathematics; by means of allowing straightforward steady-state initialization; and by means of several user-friendly modeling extensions.

The $\chi$ formalism is suited to modeling, simulation and verification of (timed) discrete-event systems without (differential) equations, continuous-time systems consisting of ordinary differential equations with algebraic constraints, and combined discrete-event/continuous-time systems. It is especially suited for the specification and analysis of complex systems. This is achieved by means of the process terms for scoping, that integrate abstraction, local variables, local channels and recursion definitions; by means of the process definition and instantiation modeling extensions that enable process re-use, encapsulation, hierarchical and/or modular composition of processes; and by means of the different interaction mechanisms, namely handshake synchronization and synchronous communication that are mainly intended for discrete-event processes that do not share variables, and shared variables that are mainly intended for interaction between

continuous-time processes.

Future work entails among others:

- The definition of (formal) translations from $\chi$ to hybrid automata and input languages of verification tools to enable verification of model properties. For timed models, without differential equations, such tools could be KRONOS [20], UPPAAL [31], Spin [29], and $\mu$CRL [24]. For hybrid models HYTECH [5], CheckMate [43], `d/dt` [6], and the tools [4, 16] are options.
- Redesign of the old hybrid $\chi$ simulator described in [21].
- Where $ACP_{hs}^{srt}$ and HyPA have derived large sets of axioms to support equational reasoning. In $\chi$, so far, only a limited set of properties has been derived. More properties need to be derived.
- Formalization of distributions, that are extensively used to model stochastic behavior in $\chi$ discrete-event simulation models, and possibly extension of the $\chi$ formalism with stochastic differential equations.

## Acknowledgments

The authors would like to thank Jos Baeten, Pieter Cuijpers, Albert Hofkamp, Niek Jansen, Erjen Lefeber, Bas Luttik, Henk Nijmeijer, Sasha Pogromsky, and Frits Vaandrager for helpful comments and stimulating discussions.

# Bibliography


[1] L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, chapter 3, pages 197–292. Elsevier, 2001.

[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(1):3–34, 1995.

[3] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. J. Pappas, and O. Sokolsky. Hierarchical modeling and analysis of embedded systems. *Proceedings of the IEEE*, 91(1):11–28, 2003.

[4] R. Alur, T. Dang, and F. Ivančić. Progress on reachability analysis of hybrid systems via predicate abstraction. In O. Maler and A. Pnueli, editors, *Hybrid Systems : Computation and Control, 6th International Workshop*, Lecture Notes in Computer Science 2623, pages 4–19. Springer-Verlag, 2003.

[5] 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.

[6] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In N. A. Lynch and B. H. Krogh, editors, *Hybrid Systems : Computation and Control, Third International Workshop*, Lecture Notes in Computer Science 1790, pages 20–31. Springer-Verlag, 2000.

[7] J. Baeten and J. Bergstra. Process algebra with propositional signals. *Theoretical Computer Science*, 177(2):381–405, 1997.

[8] J. Baeten and C. Middelburg. Process algebra with timing. In *EACTS Monographs in Theoretical Computer Science*. Springer-Verlag, 2002.

[9] J. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4 (Semantic Modelling), pages 149–268. Oxford University Press, 1995.

[10] J. Baeten and W. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, United Kingdom, 1990.

[11] J. A. Bergstra and C. A. Middelburg. Process algebra for hybrid systems. Technical Report CS-Report 03-06, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2003.

[12] V. Bos and J. J. T. Kleijn. Automatic verification of a manufacturing system. *Robotics and Computer Integrated Manufacturing*, 17(3):185–198, 2000.

[13] V. Bos and J. J. T. Kleijn. *Formal Specification and Analysis of Industrial Systems*. PhD thesis, Eindhoven University of Technology, 2002.

[14] F. Breitenecker and I. Husinsky, editors. *Simulation News Europe*, chapter ARGESIM Comparisons. Number 0-40. EUROSIM, 1990-2004.

[15] Z. Chaochen, W. Ji, and A. P. Ravn. A formal description of hybrid systems. In R. Alur, T. A. Henzinger, and E. D. Sonntag, editors, *Hybrid Systems III - Verification and Control*, Lecture Notes in Computer Science 1066, pages 511–530. Springer-Verlag, 1996.

[16] E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald. Verification of hybrid systems based on counterexample-guided abstraction. In H. Garavel and J. Hatcliff, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 2619, pages 192–207. Springer-Verlag, 2003.

[17] P. J. L. Cuijpers and M. A. Reniers. Hybrid process algebra. Technical Report CS-Report 03-07, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2003. To appear in Journal of Logic and Algebraic Programming, 2004.

[18] P. J. L. Cuijpers, M. A. Reniers, and W. P. M. H. Heemels. Hybrid transition systems. Technical Report CS-Report 02-12, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2002.

[19] R. David and H. Alla. On hybrid Petri nets. *Discrete Event Dynamic Systems: Theory & Applications*, 11(1-2):9–40, 2001.

[20] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In R. Alur, T. A. Henzinger, and E. D. Sonntag, editors, *Hybrid Systems III - Verification and Control*, Lecture Notes in Computer Science 1066, pages 208–219. Springer-Verlag, 1996.

[21] G. Fábián. *A Language and Simulator for Hybrid Systems*. PhD thesis, Eindhoven University of Technology, 1999.

[22] A. D. Febbraro, A. Giua, and G. Menga, editors. *Special Issue on Hybrid Petri Nets*, volume 11, no. 1 and 2 of *Journal of Discrete Event Dynamic Systems*, 2001.

[23] A. F. Filippov. *Differential Equations with Discontinuous Right Hand Sides*. Kluwer Academic Publishers, Dordrecht, 1988.

[24] J. Groote. The syntax and semantics of timed $\mu$CRL. Technical Report SEN-R9709, CWI, The Netherlands, 1997.

[25] W. P. M. H. Heemels, B. D. Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[26] T. A. Henzinger. Masaccio: A formal model for embedded components. In *First IFIP International Conference on Theoretical Computer Science (TCS)*, Lecture Notes in Computer Science 1872, pages 549–563. Springer-Verlag, 2000.

[27] T. A. Henzinger. The theory of hybrid automata. In M. Inan and R. Kurshan, editors, *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series F: Computer and Systems Science*. Springer-Verlag, New York, 2000.

[28] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[29] G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley Professional, Boston, 2003.

[30] H. Jifeng. From CSP to hybrid systems. In A. W. Roscoe, editor, *A Classical Mind, Essays in Honour of C.A.R. Hoare*, pages 171–189. Prentice Hall, 1994.

[31] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[32] N. Lynch, R. Segala, and F. Vaandrager. Hybrid i/o automata. *Inf. Comput.*, 185(1):105–157, 2003.

[33] N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata. Technical Report MIT-LCS-TR-827d, MIT Laboratory for Computer Science, Cambridge, MA 02139, 2003. To appear in Information and Computation.

[34] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

[35] M. Mousavi, M. Reniers, and J. F. Groote. Congruence for SOS with data. In *Proceedings of Nineteenth Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 302–313, Turku, Finland, 2004. IEEE Computer Society Press.

[36] G. Naumoski and W. Alberts. *A Discrete-Event Simulator for Systems Engineering*. PhD thesis, Eindhoven University of Technology, 1998.

[37] X. Nicollin and J. Sifakis. The algebra of timed processes, atp: Theory and application. *Information and Computation*, 114:131–178, 1994.

[38] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings $5^{th}$ GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[39] G. D. Plotkin. A structural approach to operational semantics. Technical Report DIAMI FN-19, Computer Science Department, Aarhus University, 1981.

[40] W. Rounds and H. Song. The $\phi$-calculus: A language for distributed control of reconfigurable embedded systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems : Computation and Control, 6th International Workshop*, Lecture Notes in Computer Science 2623, pages 435–449. Springer-Verlag, 2003.

[41] R. R. H. Schiffelers, D. A. van Beek, K. L. Man, M. A. Reniers, and J. E. Rooda. Formal semantics of hybrid Chi. In K. G. Larsen and P. Niebert, editors, *Formal Modeling and Analysis of Timed Systems: First International Workshop, FORMATS 2003*, volume 2791 of *Lecture Notes in Computer Science*, pages 151–165. Springer-Verlag, 2003.

[42] R. R. H. Schiffelers, D. A. van Beek, K. L. Man, M. A. Reniers, and J. E. Rooda. A hybrid language for modeling, simulation and verification. In S. Engell, H. Guéguen, and J. Zaytoon, editors, *IFAC Conference on Analysis and Design of Hybrid Systems*, pages 235–240, Saint-Malo, Brittany, France, 2003.

[43] B. I. Silva, K. Richeson, B. H. Krogh, and A. Chutinan. Modeling and verification of hybrid dynamical system using CheckMate. In S. Engell, S. Kowalewski, and J. Zaytoon, editors, *Hybrid Dynamical Systems—Proc. of 4th International Conference on Automation of Mixed Processes*, pages 323–328, Dortmund, 2000.

[44] V. I. Utkin. *Sliding Modes in Control Optimization*. Springer-Verlag, Berlin, 1992.

[45] D. A. van Beek, S. H. F. Gordijn, and J. E. Rooda. Integrating continuous-time and discrete-event concepts in modelling and simulation of manufacturing machines. *Simulation Practice and Theory*, 5(7-8):653–669, 1997.

[46] D. A. van Beek, A. Pogromsky, H. Nijmeijer, and J. Rooda. Convex equations and differential inclusions in hybrid systems. In *43rd IEEE Conference on Decision and Control*, Nassau Bahamas, 2004. to be published.

[47] D. A. van Beek and J. E. Rooda. Languages and applications in hybrid modelling and simulation: Positioning of Chi. *Control Engineering Practice*, 8(1):81–91, 2000.

[48] D. A. van Beek, A. van den Ham, and J. E. Rooda. Modelling and control of process industry batch production systems. In *15th Triennial World Congress of the International Federation of Automatic Control*, Barcelona, 2002. CD-ROM.

[49] P. van de Brand, M. A. Reniers, and P. J. L. Cuijpers. Linearization of hybrid processes. Technical Report CS-Report 04-29, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2004.

[50] A. J. van der Schaft and J. M. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Springer Lecture Notes in Control and Information Sciences*. Springer, 2000.

[51] J. Vereijken. A process algebra for hybrid systems. In Bouajjani and Maler, editors, *The Second European Workshop on Real-Time and Hybrid Systems*, Grenoble, France, 1995.

# Appendix A

# Proofs of properties of the Chi semantics

## A.1 The proof of Lemma 1

*Let $p$ and $p'$ be closed process terms, $\sigma, \sigma'$ be valuations, $\xi, \xi'$ be extended valuations, $E$ and $E'$ be environments, $a$ be an action, $\rho$ be a trajectory, and $t \in T$. Then*

$$\begin{aligned}
\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle _, \sigma', E'\rangle \quad &\Rightarrow \quad \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma' \\
&\quad \wedge E = E', \\
\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle \quad &\Rightarrow \quad \mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma' \\
&\quad \wedge E = E', \\
\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \quad &\Rightarrow \quad \xi_\sigma = \sigma,
\end{aligned}$$

*where $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma', E'\rangle$ for some $p'$.*

*Proof.* We prove this lemma by induction on the depth of the proof of the transition in the left-hand-side of the implication and case distinction on the deduction rule applied last in such a proof. The proof for the equality $E = E'$ in the right-hand-side of the implication is trivial, because the equality $E = E'$ holds necessarily according to the result of each $\chi$ deduction rule. Therefore, we do not give the proof of this equality for each rule. In what follows, we write $E'$ as $E$.

Firstly, we give the proofs for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle \Rightarrow \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 1. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$ and $\sigma' = \xi'_\sigma$, where $\xi'_\sigma$ is an abbreviation for $\xi' \restriction \mathrm{dom}(\sigma)$. The domain of the extended valuation $\xi'$ is given by $\mathrm{dom}(\sigma) \cup \dot{C} \cup L$, and the domain of $\xi' \restriction \mathrm{dom}(\sigma)$ is $\mathrm{dom}(\xi') \cap \mathrm{dom}(\sigma)$. Since $\mathrm{dom}(\sigma') = \mathrm{dom}(\xi'_\sigma)$, it is not hard to see that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. For $\xi = \sigma \cup \xi^{\dot{C}L}$, we obtain $\xi_\sigma = \sigma$. We also have $\sigma' = \xi'_{\sigma'}$, because $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$.

- Rules 5 and 6 are similar to the previous case.

- Rule 10. Then, $p = [q]$ for some $q$ and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 13. Then $p \equiv u \curvearrowright q$ for some $u$ and $q$, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\xi \models u$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 16. Then $p \equiv q_1 ; q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 17. Then $p \equiv q_1 ; q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 20. Then $p \equiv b \to q$ for some $b$ and $q$, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\xi \models b$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 25. Then $p \equiv q_1 \,[]\, q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 28. Then $p \equiv q_1 \parallel q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E_a\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E_a\rangle$ and $\langle q_2, \sigma, E_b\rangle \xrightarrow{\xi,b,\xi'} \langle \_, \sigma', E_b\rangle$ for some (unimportant) actions $a$ and $b$, and some (unimportant) environments $E_a$ and $E_b$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 29. Then $p \equiv q_1 \parallel q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$ and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 32. Then $p \equiv \partial_{\mathcal{A}}(q)$ for some $\mathcal{A}$ and $q$, $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$, and $a \notin \mathcal{A}$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 35. Then $p \equiv \upsilon_{\mathcal{H}}(q)$ for some $\mathcal{H}$ and $q$, and $\langle q, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 38. Then $p \equiv X$ for some $X$, $E = (C, J, L, R)$ and $\langle R(x), \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction, we have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 41. Then $E = (C, J, L, R)$ and $p \equiv \iota_{J^+}(q)$ for some $J^+$ and $q$, and $(C, J \cup J^+, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

- Rule 44. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi_x,a,\xi_y} \langle \_, \sigma', E\rangle$ for some $\xi_x$ and $\xi_y$. Then, $E = (C, J, L, R)$, $p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q \,]\!]$ for some $q, \sigma_{\mathrm{dx}_\perp}, \mathbf{x}, \mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma''\rangle$ for some $\mathbf{d}, \mathbf{d}', \mathbf{x}', \mathbf{g}', \sigma_{\mathrm{d}'\mathrm{x}'}, \sigma'', \sigma' = \sigma''_\sigma; \xi, \xi'$such that $\xi_x = \xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi_y = \xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. Note that the syntactical equality of $p'$ is not given, because it is irrelevant for this proof.

  - Firstly, we have to show that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma''_\sigma)$. By induction, we know that $\mathrm{dom}(\sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}) = \mathrm{dom}(\sigma) \cup (\sigma_{\mathrm{d}'\mathrm{x}'}) = \mathrm{dom}(\sigma'')$. On the other hand, $\mathrm{dom}(\sigma''_\sigma) = \mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma) = (\mathrm{dom}(\sigma) \cup (\sigma_{\mathrm{d}'\mathrm{x}'})) \cap \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma)$, i.e. $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma''_\sigma)$.
  - Secondly, we have to show that $\xi_x \restriction \mathrm{dom}(\sigma) = \sigma$. By induction, we know that $\xi \restriction \mathrm{dom}(\sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}) = \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}$, then $\xi \restriction \mathrm{dom}(\sigma) = \sigma$ and $\xi \restriction \mathrm{dom}(\sigma_{\mathrm{d}'\mathrm{x}'}) = \sigma_{\mathrm{d}'\mathrm{x}'}$. On the other hand, $\xi_x \restriction \mathrm{dom}(\sigma) = (\xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \restriction \mathrm{dom}(\sigma) = \xi \restriction \mathrm{dom}(\sigma) = \sigma$, i.e. $\xi_x \restriction \mathrm{dom}(\sigma) = \sigma$.
  - Thirdly, we have to show that $\xi_y \restriction \mathrm{dom}(\sigma''_\sigma) = \sigma''_\sigma$. By induction, we know that $\xi \restriction \mathrm{dom}(\sigma'') = \sigma''$. On the other hand, $\xi_y \restriction \mathrm{dom}(\sigma''_\sigma) = (\xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \restriction \mathrm{dom}(\sigma''_\sigma) = (\xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \restriction (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma)) = \xi' \restriction (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma))$. From $\sigma'' = \xi \restriction \mathrm{dom}(\sigma'')$, we obtain $\sigma''_\sigma = \sigma'' \restriction \mathrm{dom}(\sigma) = (\xi \restriction \mathrm{dom}(\sigma'')) \restriction \mathrm{dom}(\sigma)$. It is not hard to see that $\xi' \restriction (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma)) = (\xi \restriction \mathrm{dom}(\sigma'')) \restriction \mathrm{dom}(\sigma)$, which also means $\xi_y \restriction \mathrm{dom}(\sigma''_\sigma) = \sigma''_\sigma$.

- Rules 47, 48 and 51. The proofs are similar. We only give the proof for Rule 47. Then $p \equiv [\![_{\mathrm{H}}\, H_0 \mid q \,]\!]$ for some $q, H_0$, $\langle q, \sigma, E\rangle \xrightarrow{\xi,b,\xi'} \langle \_, \sigma', E\rangle$ for some unimportant action $b$ for this proof, and $h \in H_0$. By induction we then have $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') \wedge \xi_\sigma = \sigma \wedge \xi'_{\sigma'} = \sigma'$.

The rules that have not been considered could not have been applied last since they conclude a time transition or a consistency predicate.

Secondly, we give the proofs for $\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E\rangle \Rightarrow \mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 3. Then, $p \equiv u \equiv p'$ for some $u$, $E = (C, J, L, R)$, $\rho \in \Omega(\sigma, C, L, u, t)$, and $\sigma' = \rho_\sigma(t)$. Then, by the definition of $\Omega$, $\mathrm{dom}(\rho) = [0, t]$, and $\rho(0) \restriction \mathrm{dom}(\sigma) = \rho_\sigma(0) = \sigma$ necessarily. From $\sigma' = \rho_\sigma(t)$, we know that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. Therefore, we also have $\sigma' = \rho_{\sigma'}(t)$.

- Rule 11. Then $p \equiv [q] \equiv p'$ for some $q$, $\rho \in \Omega_{\sigma E t}$ and $\sigma' = \rho_\sigma(t)$. Then, by the definition of $\Omega$, $\mathrm{dom}(\rho) = [0, t]$, and $\rho(0) \restriction \mathrm{dom}(\sigma) = \rho_\sigma(0) = \sigma$ necessarily. From $\sigma' = \rho_\sigma(t)$, we know that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. Therefore, we have also $\sigma' = \rho_{\sigma'}(t)$.

- Rule 14. Then $p \equiv u \curvearrowright q$ for some $u$ and $q$, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$ and $\rho(0) \models u$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 18. Then $p \equiv q_1;\ q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E\rangle$ for some $q_1'$ and $p' \equiv q_1';\ q_2$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 21. Then $p \equiv b \to q$ for some $b$ and $q$, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some $q'$ such that $p' \equiv b \to q'$, and $\forall_{s\in[0,t]}\rho(s) \models b$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 22. Then $p \equiv b \to q \equiv p'$ for some $b$ and $q$, $\rho \in \Omega_{\sigma E t}$ and $\sigma' = \rho_\sigma(t)$ (some irrelevant information for the proof is omitted). By the definition of $\Omega$, $\mathrm{dom}(\rho) = [0, t]$, and $\rho(0) \restriction \mathrm{dom}(\sigma) = \rho_\sigma(0) = \sigma$ necessarily. From $\sigma' = \rho_\sigma(t)$, we know that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$. Therefore, we have also $\sigma' = \rho_{\sigma'}(t)$.

- Rule 26. Then $p \equiv q_1 \,[]\, q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E\rangle$ for some $q_1'$ and $q_2'$, and $p' \equiv q_1' \,[]\, q_2'$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 30. Then $p \equiv q_1 \parallel q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E\rangle$ and $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E\rangle$, for some $q_1'$ and $q_2'$, and $p' \equiv q_1' \parallel q_2'$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 33. Then $p \equiv \partial_{\mathcal{A}}(q)$ for some $\mathcal{A}$ and $q$, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some $q'$, and $p' \equiv \partial_{\mathcal{A}}(q')$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 36. Then $p \equiv \upsilon_{\mathcal{H}}(q)$ for some $\mathcal{H}$ and $q$, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some $q'$, and $p' \equiv \upsilon_{\mathcal{H}}(q')$ (some irrelevant information for this proof is omitted). By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 39. Then $p \equiv X$ for some $X$, $E = (C, J, L, R)$ and $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E\rangle$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 42. Then $p \equiv \iota_{J^+}(q)$ for some $q$ and $J^+$, $E = (C, J, L, R)$, $(C, J \cup J^+, L, R) \Vdash \langle q, \sigma, \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some $q'$, and $p' \equiv \iota_{J^+}(q')$. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

- Rule 45. We assume $\langle p, \sigma, E\rangle \overset{t,\rho'}{\longmapsto} \langle p', \sigma', E\rangle$ for some $\rho'$. Then $E = (C, J, L, R)$, $p \equiv [\![_{\mathrm{V}}\ \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\ ]\!]$ for some $q$, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d'x'}}\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma''\rangle$ for some $q'$, $\mathbf{d}$, $\mathbf{d}'$, $\mathbf{x}'$, $\mathbf{g}'$, $\sigma_{\mathrm{d'x'}}$, $\sigma''$, $\sigma' = \sigma''_\sigma$, and $\rho' = \rho_{\sigma \dot{C} L} = \rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. Note that the syntactical equality of $p'$ is not given, because it is irrelevant for this proof.

  - Firstly, we have to show that $\mathrm{dom}(\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) = [0, t]$. By induction we know that $\mathrm{dom}(\rho) = [0, t]$. On the other hand, we have $\mathrm{dom}(\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) = \mathrm{dom}(\rho) = [0, t]$.

  - Secondly, we have to show that $\rho' \downarrow \mathrm{dom}(\sigma)(0) = (\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma)(0) = \sigma$. By induction we know that $\rho \downarrow (\mathrm{dom}(\sigma \cup \sigma_{\mathrm{d'x'}}))(0) = \sigma \cup \sigma_{\mathrm{d'x'}}$. Then, we have also $\rho \downarrow \mathrm{dom}(\sigma)(0) = \sigma$ and $\rho \downarrow \mathrm{dom}(\sigma_{\mathrm{d'x'}})(0) = \sigma_{\mathrm{d'x'}}$. On the other hand, $\rho' \downarrow \mathrm{dom}(\sigma)(0) = (\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma)(0) = \rho \downarrow \mathrm{dom}(\sigma)(0) = \sigma$.
  - Thirdly, we have to show that $\rho' \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = (\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = \sigma''_\sigma$. By induction we know that $\rho \downarrow \mathrm{dom}(\sigma'')(t) = \sigma''$. Then, we have $(\rho \downarrow \mathrm{dom}(\sigma'')) \downarrow \mathrm{dom}(\sigma)(t) = \sigma'' \downarrow \mathrm{dom}(\sigma) = \sigma''_\sigma$. On the other hand, $\rho' \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = ((\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow \mathrm{dom}(\sigma''_\sigma)(t) = ((\rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)) \downarrow (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma)))(t) = \rho \downarrow (\mathrm{dom}(\sigma'') \cap \mathrm{dom}(\sigma))(t) = (\rho \downarrow \mathrm{dom}(\sigma'')) \downarrow \mathrm{dom}(\sigma)(t) = \sigma''_\sigma$.

- Rules 49 and 52. The proofs are similar. We only give the proof for Rule 49. Then $p \equiv [\![_{\mathrm{H}}\ H_0 \mid q\ ]\!]$ for some $H_0$ and $q$, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some $q'$. Note that the syntactical equality of $p'$ is not given, because it is irrelevant for this proof. By induction we then have $\mathrm{dom}(\rho) = [0, t] \wedge \rho_\sigma(0) = \sigma \wedge \rho_{\sigma'}(t) = \sigma'$.

The rules that have not been considered could not have been applied last since they conclude an action transition or a consistency predicate.

The proof for $\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow} \Rightarrow \xi_\sigma = \sigma$ is trivial. According to all $\chi$ deduction rules for consistency transitions, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$ necessarily. Then we have $\xi_\sigma = \sigma$. ⊠

## A.2 The proof of Lemma 2

*Let $p$ and $p'$ be closed process terms, $\sigma$ and $\sigma'$ be valuations, $E$ and $E'$ be environments, $\xi$ and $\xi'$ be extended valuations and $a$ be an action. Then*

$$\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \implies \langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow},$$

*where $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma', E'\rangle$ for some $p'$, $\sigma'$, and $E'$.*

*Proof.* We prove this lemma by induction on the depth of the proof of $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'}$ using case distinction based on the deduction rule applied last. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 1. Then $p \equiv W : r \gg l_{\mathrm{a}}$ for some $W, r, l_{\mathrm{a}}$, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$, and $a = l_{\mathrm{a}}$. Therefore, by Rule 2, we have $\langle W : r \gg l_{\mathrm{a}}, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 5. Then $p \equiv h \,!!\, \mathbf{e}_n$ for some $h$ and $\mathbf{e}_n$, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$, and $a = \mathrm{isa}(h, [\xi(\mathbf{e}_n)])$. Therefore, by Rule 7, we have $\langle h \,!!\, \mathbf{e}_n, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 6. Then $p \equiv h \,??\, \mathbf{x}_n$ for some $h$ and $\mathbf{x}_n$, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$, and $a = \mathrm{ira}(h, [\mathbf{c}_n], \{\mathbf{x}_n\})$ for some $\mathbf{c}_n$. Then, by Rule 8, we have $\langle h \,??\, \mathbf{x}_n, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 10. Then, $p = [q]$ for some $q$, $E = (C, J, L, R)$ and $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$. By induction we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 12, we have $\langle [q], \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$, and $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$.

- Rule 13. Then $p \equiv u \curvearrowright q$ for some $u$ and $q$, $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$ and $\xi \models u$. By induction $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 15, we have $\langle u \curvearrowright q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 16. Then $p \equiv q_1 ; q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma', E'\rangle$ and $\langle q_2, \sigma', E\rangle \overset{\xi'}{\rightsquigarrow}$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 19, we have $\langle q_1 ; q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 17. Then $p \equiv q_1 ; q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi, a, \xi'} \langle q_1', \sigma', E'\rangle$ for some $q_1'$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 19, we have $\langle q_1 ; q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 20. Then $p \equiv b \to q$ for some $b$ and $q$, $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$ and $\xi \models b$. By induction $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 23, we have $\langle b \to q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 25. Then $p \equiv q_1 \,[]\, q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 27, we have $\langle q_1 \,[]\, q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 28. Then $p \equiv q_1 \parallel q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E_a\rangle \xrightarrow{\xi, a, \xi'}$ and $\langle q_2, \sigma, E_b\rangle \xrightarrow{\xi, b, \xi'}$ for some (unimportant) actions $a$ and $b$, and some (unimportant) environments $E_a$ and $E_b$. By induction $\langle q_1, \sigma, E_a\rangle \overset{\xi}{\rightsquigarrow}$ and $\langle q_2, \sigma, E_b\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 31 and by Lemma 4, we have $\langle q_1 \,[]\, q_2, \sigma, E_a\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 29. Then $p \equiv q_1 \parallel q_2$ for some $q_1$ and $q_2$, and $\langle q_1, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$ and $\langle q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. By induction $\langle q_1, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 31, we have $\langle q_1 \,[]\, q_2, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 32. Then $p \equiv \partial_{\mathcal{A}}(q)$ for some $\mathcal{A}$ and $q$, $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$, and $a \notin \mathcal{A}$. By induction we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34, we obtain $\langle \partial_{\mathcal{A}}(q), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 35. Then $p \equiv \upsilon_{\mathcal{H}}(q)$ for some $\mathcal{H}$ and $q$, and $\langle q, \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$. By induction we then have $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 37, we obtain $\langle \upsilon_{\mathcal{H}}(q), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 38. Then $p \equiv X$ for some $X$ and $E = (C, J, L, R)$ and $\langle R(x), \sigma, E\rangle \xrightarrow{\xi, a, \xi'}$. By induction, we have $\langle R(x), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Then, by Rule 40, $\langle X, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 41. Then $E = (C, J, L, R)$ and $p \equiv \iota_{J^+}(q)$ for some $J^+$ and $q$, and $(C, J \cup J^+, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi, a, \xi'}$ . By induction we have $(C, J \cup J^+, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. By Rule 43, we have $\langle \iota_{J^+}(q), \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

- Rule 44. We assume $\langle p, \sigma, E\rangle \xrightarrow{\xi_x, a, \xi_y}$ for some $\xi_x$, and $\xi_y$. Then, $E = (C, J, L, R)$, $p \equiv [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\, ]\!]$ for some $q, \sigma_{\mathrm{dx}_\perp}, \mathbf{x}, \mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \xrightarrow{\xi, a, \xi'}$ for some $\mathbf{d}, \mathbf{d}', \mathbf{x}', \mathbf{g}', \sigma_{\mathrm{d}'\mathrm{x}'}; \xi, \xi'$ such that $\xi_x = \xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$ and $\xi_y = \xi' \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. By induction we have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 46, we obtain $(C, J, L, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p\, ]\!], \sigma\rangle \overset{\xi \restriction (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)}{\rightsquigarrow}$.

- Rules 47, 48 and 51. The proofs are similar. We only give the proof for Rule 47. Then $p \equiv [\![_{\mathrm{H}}\, H_0 \mid q\, ]\!]$ for some $q, H_0$, $\langle q, \sigma, E\rangle \xrightarrow{\xi, b, \xi'}$ for some unimportant action $b$ for this proof, and $h \in H_0$. By induction we then $\langle q, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 50, we obtain $\langle [\![_{\mathrm{H}}\, H_0 \mid q\, ]\!], \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$.

The rules that have not been considered could not have been applied last since they conclude a time transition or a consistency predicate. ⊠

## A.3 The proof of Lemma 3

*Let $p$ and $p'$ be closed process terms, $\sigma$ and $\sigma'$ be valuations, $E$ and $E'$ be environments, $t \in T$, and $\rho$ be a trajectory. Then,*

$$\langle p, \sigma, E\rangle \xmapsto{t, \rho} \langle p', \sigma', E'\rangle \quad \Rightarrow \quad \langle p, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow} .$$

*Proof.* We prove this lemma by induction on the depth of the proof of $\langle p, \sigma, E\rangle \xmapsto{t, \rho} \langle p', \sigma', E'\rangle$ using case distinction based on the deduction rule applied last. We do not explicitly separate the base cases and the inductive steps.

The rule applied last is

- Rule 3. Then, $p \equiv u \equiv p'$ for some $u$, $E = (C, J, L, R)$, $\rho \in \Omega(\sigma, C, L, u, t)$. Then, by definition, $\rho(0) \models u$ and $\rho(0) \restriction \mathrm{dom}(\sigma) = \sigma$. Thus $\rho(0) = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Therefore, by Rule 4, we have $\langle u, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 11. Then $p \equiv [q]$ for some $q$ and $\rho(0) \in \Omega_{\sigma E t}$. Then, by definition, $\rho(0) \restriction \mathrm{dom}(\sigma) = \sigma$. Thus $\rho(0) = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Therefore, by Rule 12, $\langle [q], \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 14. Then $p \equiv u \curvearrowright q$ for some $u$ and $p$, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$ and $\rho(0) \models u$. Therefore, by induction, $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. Then, by Rule 15, $\langle u \curvearrowright q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 18. Then $p \equiv q_1;\ q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E'\rangle$ for some $q_1'$, and $p' \equiv q_1';\ q_2$. By induction we have $\langle q_1, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$, and thus by application of Rule 19 we have $\langle q_1;\ q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 21. Then $p \equiv b \to q$ for some $b$ and $q$, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E'\rangle$ for some $q'$ such that $p' \equiv b \to q'$, and $\forall_{s\in[0,t]}\rho(s) \models b$. By induction we have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. Since we also have $\rho(0) \models b$, we have, by Rule 23, $\langle b \to q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 22. Then $p \equiv b \to q$ for some $b$ and $q$, $\rho \in \Omega_{\sigma Et}$, $\forall_{s\in(0,t)}\rho(s) \models \neg b$, $\rho(0) \models b \implies \langle q, \sigma, E\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q', \sigma'', E''\rangle$ for some $q'$, $\sigma''$ and $E''$. In case $\rho(0) \models \neg b$, we also have $\sigma \cup \xi^{\dot{C}L} \models \neg b$ for some $\xi^{\dot{C}L} \in (\dot{C} \cup L) \mapsto \Lambda$. Then, by Rule 24, $\langle b \to q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. In case $\rho(0) \models b$, we have $\langle q, \sigma, E\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q', \sigma'', E''\rangle$. By induction we then have $\langle q, \sigma, E\rangle \overset{\rho\restriction\{0\}(0)}{\rightsquigarrow}$, which gives $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 23 we then have $\langle b \to q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 26. Then $p \equiv q_1 \,[]\, q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E'\rangle$ for some $q_1'$, $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E'\rangle$ for some $q_2'$, and $p' \equiv q_1' \,[]\, q_2'$. By induction we have $\langle q_1, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$ and $\langle q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$, and thus by application of Rule 27 we have $\langle q_1 \,[]\, q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 30. Then $p \equiv q_1 \parallel q_2$ for some $q_1$ and $q_2$, $\langle q_1, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_1', \sigma', E'\rangle$ for some $q_1'$, $\langle q_2, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q_2', \sigma', E'\rangle$ for some $q_2'$, and $p' \equiv q_1' \parallel q_2'$. By induction we have $\langle q_1, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$ and $\langle q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$, and thus by application of Rule 31 we have $\langle q_1 \parallel q_2, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 33. Then $p \equiv \partial_{\mathcal{A}}(q)$ for some $\mathcal{A}$ and $q$, $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some $q'$, and $p' \equiv \partial_{\mathcal{A}}(q')$. By induction we then have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 34, we obtain $\langle \partial_{\mathcal{A}}(q), \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 36. Then $p \equiv \upsilon_{\mathcal{H}}(q)$ for some $\mathcal{H}$ and $q$, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma, E\rangle$ for some $q'$, and $p' \equiv \upsilon_{\mathcal{H}}(q')$ (some irrelevant information for the proof is omitted). By induction we then have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 37, we obtain $\langle \upsilon_{\mathcal{H}}(q), \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 39. Then $p \equiv X$ for some $X$, $E = (C, J, L, R)$ and $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$. As the proof for $\langle R(X), \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', E'\rangle$ has smaller depth, by induction we have $\langle R(X), \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. Then, by Rule 40, we have $\langle X, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$ as well.

- Rule 42. Then $p \equiv \iota_{J^+}(q)$ for some $q$ and $J^+$, $E = (C, J, L, R)$, $(C, J \cup J^+, L, R) \Vdash \langle q, \sigma, \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma'\rangle$ for some $q'$, and $p' \equiv \iota_{J^+}(q')$. By induction we then have $(C, J \cup J^+, L, R) \Vdash \langle q, \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$. From Rule 43, we deduce $\langle \iota_{J^+}(q), \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$.

- Rule 45. We assume $\langle p, \sigma, E\rangle \overset{t,\rho'}{\longmapsto} \langle p', \sigma', E'\rangle$ for some $\rho'$. Then $E = (C, J, L, R)$, $p \equiv [\![_{\mathrm{V}}\ \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\ ]\!]$ for some $q$, $\sigma_{\mathrm{dx}_\perp}$, $\mathbf{x}$, $\mathbf{g}$, $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma''\rangle$ for some $\rho, q', \mathbf{d}, \mathbf{d}', \mathbf{x}', \mathbf{g}', \sigma_{\mathrm{d}'\mathrm{x}'}, \sigma'', \sigma' = \sigma''_\sigma$, and $\rho' = \rho_{\sigma\dot{C}L} = \rho \downarrow (\mathrm{dom}(\sigma) \cup \dot{C} \cup L)$. Note that the syntactical equality of $p'$ is not given, because it is irrelevant for this proof. By induction we then have $(C \cup \{\mathbf{x}'\}, J, L \cup \{\mathbf{g}'\}, R) \Vdash \langle q[\mathbf{d}', \mathbf{x}', \mathbf{g}'/\mathbf{d}, \mathbf{x}, \mathbf{g}], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'}\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 46, we obtain $(C, J, L, R) \Vdash \langle [\![_{\mathrm{V}}\ \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid q\ ]\!], \sigma\rangle \overset{\rho\downarrow(\mathrm{dom}(\sigma)\cup\dot{C}\cup L)(0)}{\rightsquigarrow}$.
- Rules 49 and 52. The proofs are similar. We only give the proof for Rule 49. Then $p \equiv [\![_{\mathrm{H}}\ H_0 \mid q\ ]\!]$ for some $H_0$ and $q$, and $\langle q, \sigma, E\rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma', E\rangle$ for some $q'$. Note that the syntactical equality of $p'$ is not given, because it is irrelevant for this proof. By induction we then have $\langle q, \sigma, E\rangle \overset{\rho(0)}{\rightsquigarrow}$. By Rule 50, we obtain $\langle [\![_{\mathrm{H}}\ H_0 \mid p\ ]\!], \sigma\rangle \overset{\rho(0)}{\rightsquigarrow}$.

The rules that have not been considered could not have been applied last since they conclude an action transition or a consistency predicate. ⊠

# Appendix B

# Proofs of properties of the Chi operators

In this section, the outline of the proofs for the properties in Section 6.4 is given. For all of these properties, the proofs follow the same lines. A relation $R$ is defined containing at least all closed instantiations of the property to be proved. Then, it must be shown that this relation is a stateless bisimulation. For this in principle for each pair of closed process terms $(p, q) \in R$, it has to be shown that it satisfies the six conditions of Definition 1. Often, the relation $R$ contains pairs of the form $(i_d, i_d)$. Since the proofs are trivial for such pairs these are omitted. As the deduction rules of hybrid $\chi$ are such that the environment does not change in a transition, we only consider those cases in the proofs. As a consequence we use the notation $E \Vdash$ as much as possible.

## B.1 Properties of signal emission operator

**Lemma 5** *For arbitrary closed process term $p$ we have*

$$\text{true} \curvearrowright p \leftrightarrows p.$$

*Proof.* Let $R = \{(\text{true} \curvearrowright p, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of conditions 1 – 3 are similar to the proofs of conditions 1 – 3 of Lemma 12 (except the premise $\xi \models b$ is replaced by $\xi \models u$). The proofs of conditions 4 and 5 are similar to the proofs of conditions 2 and 3 (notice that the premise $\xi \models u$ is replaced by $\rho(0) \models u$ in the proofs). The proofs of condition 6 are similar to the proofs of condition 6 of Lemma 12 (except Rule 24 has not been applied, and the premise $\xi \models b$ is replaced by $\xi \models u$ in the proofs). ⊠

**Lemma 6** *For arbitrary closed process term $p$ we have*

$$\text{false} \curvearrowright p \leftrightarrow \bot.$$

*Proof.* Since there are no action transition rules, time transition rules and consistency transition rules defined for $\curvearrowright$ in which the initialization predicate is not satisfied, also indicate that $\text{false} \curvearrowright p$ cannot perform any transition. Therefore, the conditions 1 – 6 hold trivially. ☒

**Lemma 7** *For arbitrary predicate $u$ we have*

$$u \curvearrowright u \leftrightarrow u.$$

*Proof.* Let $R = \{(u \curvearrowright u, u) \mid \text{predicate } u\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since there are no action transition rules defined for $u$, also indicate that $u \curvearrowright u$ has no action transitions, the conditions 1 – 3 hold trivially.

*Condition 4*: We assume $(C, J, L, R) \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $(C, J, L, R), \sigma, t, \rho, k_1, \sigma'$, which means that Rule 14 has been applied necessarily. Then, $(C, J, L, R) \Vdash \langle u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u$. For $(C, J, L, R) \Vdash \langle u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$, Rule 3 has been applied necessarily. Then, $\rho \in \Omega(\sigma, C, L, u, t)$, $\sigma' = \rho_\sigma(t)$ and $k_1 \equiv u$. Using Rule 3, we obtain $(C, J, L, R) \Vdash \langle u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle u, \rho_\sigma(t)\rangle$ and observe that $(u, u) \in R$.

*Condition 5*: We assume $(C, J, L, R) \Vdash \langle u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $(C, J, L, R), \sigma, t, \rho, k_1, \sigma'$, which means that Rule 3 has been applied necessarily. Then, $\rho \in \Omega(\sigma, C, L, u, t)$, $\sigma' = \rho_\sigma(t)$ and $k_1 \equiv u$. We know that $\forall s \in [0, t] : \rho(s) \models u$ (from the definition of the function $\Omega$). Hence, we also have $\rho(0) \models u$. Using Rule 14, we obtain $(C, J, L, R) \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle u, \rho_\sigma(t)\rangle$ and observe that $(u, u) \in R$.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means that Rule 15 has been applied necessarily. Then, $(C, J, L, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u$. Second, we assume $(C, J, L, R) \Vdash \langle u, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ for some $\xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models u$. According to Rule 15, we get $(C, J, L, R) \Vdash \langle u \curvearrowright u, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. ☒

**Lemma 8** *For arbitrary closed process term $p$ and arbitrary predicates $u, u'$ we have*

$$u \curvearrowright (u' \curvearrowright p) \leftrightarrow (u \wedge u') \curvearrowright p.$$

*Proof.* Let $R = \{(u \curvearrowright (u' \curvearrowright p), (u \wedge u') \curvearrowright p) \mid p \in P, \text{predicates } u, u'\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: First, we assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 13.1 has been applied necessarily. Then, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $\xi \models u$. Again, Rule 13.1 has been applied necessarily. Therefore, we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $\xi \models u'$. From $\xi \models u$ and $\xi \models u'$ we get $\xi \models u \wedge u'$. Using Rule 13.1, we obtain $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. Second, we assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 13.1 has been applied necessarily. Thus, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$ and $\xi \models u \wedge u'$. From $\xi \models u \wedge u'$ we obtain $\xi \models u$ and $\xi \models u'$. Using Rule 13.1, we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. Again using Rule 13.1, we obtain $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$.

*Condition 2*: We assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 13.2 has been applied necessarily. Thus, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models u$. Again, Rule 13.2 has been applied necessarily. Therefore, we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models u'$. From $\xi \models u$ and $\xi \models u'$, we obtain $\xi \models u \wedge u'$. Using Rule 13.2, we get $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 3*: We assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 13.2 has been applied necessarily. Therefore, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models u \wedge u'$. From $\xi \models u \wedge u'$, we also have $\xi \models u$ and $\xi \models u'$. Using Rule 13.2 we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. Again using Rule 13.2 we obtain $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 4*: We assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that Rule 14 has been applied necessarily. Then, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u$. For $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$, Rule 14 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u'$. From $\rho(0) \models u$ and $\rho(0) \models u'$, we obtain $\rho(0) \models u \wedge u'$. Using Rule 14, we get $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle (u \wedge u') \curvearrowright k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 5*: We assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that Rule 14 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and $\rho(0) \models u \wedge u'$. From $\rho(0) \models u \wedge u'$, we can also have $\rho(0) \models u$ and $\rho(0) \models u'$. Using Rule 14, we obtain $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$. Again, using Rule 14 we get $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means that Rule 15 has been applied necessarily. Then, $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u$. For $E \Vdash \langle u' \curvearrowright p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, Rule 15 has been applied necessarily. Then $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models u'$. From $\xi \models u$ and $\xi \models u'$, we can have $\xi \models u \wedge u'$. Using Rule 15, we obtain

$E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle (u \wedge u') \curvearrowright p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means Rule 15 has been applied necessarily. Then, $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models (u \wedge u')$. From $\xi \models (u \wedge u')$, we get $\xi \models u$ and $\xi \models u'$. According to Rule 15, we obtain $E \Vdash \langle u' \curvearrowright p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 15, we get $E \Vdash \langle u \curvearrowright (u' \curvearrowright p), \sigma \rangle \overset{\xi}{\rightsquigarrow}$. ⊠

## B.2 Properties of alternative composition

**Lemma 9 (Idempotency of alternative composition)** *For arbitrary closed term $p$ we have*

$$p \,[]\, p \leftrightarrows p.$$

*Proof.* Let $R = \{(p \,[]\, p, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: First, we assume $E \Vdash \langle p \,[]\, p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma' \rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. Since the left and right argument of the [] are the same, we only give the proofs in which Rule 25.1.l has been applied. Then, $E \Vdash \langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma' \rangle$. Second, we assume $E \Vdash \langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma' \rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$. We know that $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ (see also Lemma 2). Using Rule 25.1.l, we obtain $E \Vdash \langle p \,[]\, p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma' \rangle$.

*Condition 2*: We assume $E \Vdash \langle p \,[]\, p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma' \rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that Rule 25.2.l or Rule 25.2.r has been applied necessarily. Since the left and right argument of the [] are the same, we only the proofs in which Rule 25.1.l has been applied. Then, $E \Vdash \langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma' \rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 3*: We assume $E \Vdash \langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma' \rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$. We also know that $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ (see also Lemma 2). Using Rule 25.2.l, we obtain $E \Vdash \langle p \,[]\, p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma' \rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 4*: We assume $E \Vdash \langle p \,[]\, p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma' \rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 26 has been applied necessarily. Then, we get $E \Vdash \langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma' \rangle$ for some $k_p$ such that $k_1 \equiv k_p \,[]\, k_p$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $E \Vdash \langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma' \rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$. Using Rule 26, we obtain $E \Vdash \langle p \,[]\, p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle k_1 \,[]\, k_1, \sigma' \rangle$. Take $k_2 \equiv k_1 \,[]\, k_1$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle p \,[]\, p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means Rule 27 has been applied necessarily. Then, we get $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$. Using Rule 27, we obtain $E \Vdash \langle p \,[]\, p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$.

$\boxtimes$

**Lemma 10 (Commutativity of alternative composition)** *For arbitrary closed process terms $p$ and $q$ we have*

$$p \mathbin{[]} q \leftrightarrows q \mathbin{[]} p.$$

*Proof.* Let $R = \{(p \mathbin{[]} q, q \mathbin{[]} p) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since the deduction rules for $[]$ are symmetrical w.r.t. the left and right argument, obviously all conditions are met. $\boxtimes$

**Lemma 11 (Associativity of alternative composition)** *For arbitrary closed process terms $p$, $q$ and $r$ we have*

$$(p \mathbin{[]} q) \mathbin{[]} r \leftrightarrows p \mathbin{[]} (q \mathbin{[]} r).$$

*Proof.* Let $R = \{((p \mathbin{[]} q) \mathbin{[]} r, p \mathbin{[]} (q \mathbin{[]} r)) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similar to the proof of the right implication. The proofs of conditions 3 and 5 are similar to the proofs of conditions 2 and 4. The proof of the left implication of condition 6 is similar to the proof of the right implication.

*Condition 1*: We assume $E \Vdash \langle (p \mathbin{[]} q) \mathbin{[]} r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. Hence, we distinguish two cases:

1. Rule 25.1.l has been applied. Then, $E \Vdash \langle p \mathbin{[]} q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle p \mathbin{[]} q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, this means that again either Rule 25.1.l or Rule 25.1.r has been applied necessarily. Hence, we can further distinguish two cases:
   (a) Rule 25.1.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \overset{\xi,a,\xi'}{\rightsquigarrow} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we obtain $E \Vdash \langle q \mathbin{[]} r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We further get $E \Vdash \langle p \mathbin{[]} (q \mathbin{[]} r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ using Rule 25.1.l.
   (b) Rule 25.1.r has been applied. Then, $E \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 25.1.l, we obtain $E \Vdash \langle q \mathbin{[]} r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Applying Rule 25.1.r, we obtain $E \Vdash \langle p \mathbin{[]} (q \mathbin{[]} r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.
2. Rule 25.1.r has been applied. The proof is similar to the previous case.

*Condition 2*: We assume $E \Vdash \langle (p \mathbin{[]} q) \mathbin{[]} r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that either Rule 25.2.l or Rule 25.2.r has been applied necessarily. Hence, we distinguish two cases:

1. Rule 25.2.l has been applied. Then $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, this means that again either Rule 25.2.l or Rule 25.2.r has been applied necessarily. Hence, we again distinguish two cases:

   (a) Rule 25.2.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we obtain $E \Vdash \langle q \,[]\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We further get $E \Vdash \langle p \,[]\, (q \,[]\, r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ using Rule 25.2.l and observe that $(k_1, k_1) \in R$.

   (b) Rule 25.2.r has been applied. Then $E \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. We get $E \Vdash \langle q \,[]\, r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ using Rule 25.2.l. Applying Rule 25.2.r, we obtain $E \Vdash \langle p \,[]\, (q \,[]\, r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

2. Rule 25.2.r has been applied. The proof is similar to the previous case.

*Condition 4*: We assume $E \Vdash \langle (p \,[]\, q) \,[]\, r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 26 has been applied necessarily. Then $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ and $E \Vdash \langle r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_r, \sigma'\rangle$ for some $k_{pq}$ and $k_r$ such that $k_1 \equiv k_{pq} \,[]\, k_r$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, we obtain $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some $k_p, k_q$ such that $k_{pq} \equiv k_p \,[]\, k_q$ (using Rule 26). Applying Rule 26, we get $E \Vdash \langle q \,[]\, r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q \,[]\, k_r, \sigma'\rangle$. Again, due to Rule 26, we can have $E \Vdash \langle p \,[]\, (q \,[]\, r), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p \,[]\, (k_q \,[]\, k_r), \sigma'\rangle$. Note that $k_1 \equiv (k_p \,[]\, k_q) \,[]\, k_r$. Take $k_2 \equiv k_p \,[]\, (k_q \,[]\, k_r)$ and observe that $(k_1, k_2) \in R$.

*Condition 6*: We assume $E \Vdash \langle (p \,[]\, q) \,[]\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means Rule 27 has been applied necessarily. Then $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, we obtain $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ (see also Rule 27). Applying Rule 27, we get $E \Vdash \langle q \,[]\, r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 27, we can have $E \Vdash \langle p \,[]\, (q \,[]\, r), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. ⊠

## B.3 Properties of guard operator

**Lemma 12** *For arbitrary closed process term $p$ we have*

$$\text{true} \rightarrow p \leftrightarrow p.$$

*Proof.* Let $R = \{(\text{true} \rightarrow p, p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: First, we assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means that Rule 20.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

Second, we assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$. We also know that $\xi \models \text{true}$ by Lemma 2, and obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ using Rule 20.1.

*Condition 2*: We assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that Rule 20.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 3*: We assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$. We also know that $\xi \models \text{true}$ by Lemma 2. We obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ using Rule 20.2 and observe that $(k_1, k_1) \in R$.

*Condition 4*: We assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 21 has been applied necessarily. Notice that Rule 22 cannot be applied, because the premise $\forall_{s\in(0,t)}\ \rho(s) \models \neg\text{true}$ does not hold. Then $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \text{true} \rightarrow k_p$ and $\forall_{s\in[0,t]}\ \rho(s) \models \text{true}$. Take $k_2 \equiv k_p$ and observe that $(k_2, k_1) \in R$.

*Condition 5*: We assume $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$. We also know that $\forall_{s\in[0,t]}\ \rho(s) \models \text{true}$. We obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{true} \rightarrow k_1, \sigma'\rangle$ using Rule 21. Take $k_2 \equiv \text{true} \rightarrow k_1$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means Rule 23 has been applied necessarily. Notice that Rule 24 cannot have applied, because the premise $\sigma \cup \xi^{\dot{C}L} \models \neg\text{true}$ does not hold. Then $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$. We also know $\xi \models \text{true}$. We obtain $E \Vdash \langle \text{true} \rightarrow p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23. ⊠

**Lemma 13** *For arbitrary closed process term $p$ we have*

$$\text{false} \rightarrow p \leftrightarrows \text{true}.$$

*Proof.* Let $R = \{(\text{false} \rightarrow p, \text{true}) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since there no are action transition rules defined for guard that evaluates to false in the extended valuation (i.e. $\xi \models \text{false}$), and for the process term true also no action transition rules are defined, the conditions 1 – 3 hold trivially.

*Condition 4*: We assume $(C, J, L, R) \Vdash \langle \text{false} \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $(C, J, L, R), \sigma, t, \rho, k_1, \sigma'$ which means Rule 22 has been applied necessarily. Notice that Rule 21 cannot be applied, because the premise $\forall_{s\in[0,t]}\ \rho(s) \models \text{false}$ does not hold. Then $k_1 \equiv \text{false} \rightarrow p$, $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma Et}$ and $\forall_{s\in(0,t)}\ \rho(s) \models \neg\text{false}$. For $\rho \in \Omega_{\sigma Et}$, we can have $(C, J, L, R) \Vdash \langle \text{true}, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{true}, \rho_\sigma(t)\rangle$ (see also Rule 3). Take $k_2 \equiv \text{true}$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $(C, J, L, R) \Vdash \langle \text{true}, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $(C, J, L, R), \sigma, t, \rho, k_1, \sigma'$, which means Rule 3 has been applied necessarily. Then $k_1 \equiv \text{true}$, $\sigma' = \rho_\sigma(t)$, and $\rho \in \Omega(\sigma, C, J, L, \text{true}, t)$. We know that $\forall_{s\in(0,t)}\ \rho(s) \models \neg\text{false}$, $\rho(0) \models \text{false} \Rightarrow (C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$ for some $p', \sigma''$, and $\rho(t) \models \text{false} \Rightarrow (C, J, L, R) \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (since the left-hand sides of the implications are false, these two implications hold trivially). Using Rule 22, we obtain $(C, J, L, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \text{false} \to p, \sigma'\rangle$. Take $k_2 \equiv \text{false} \to p$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means Rule 24 has been applied necessarily. Notice that Rule 23 cannot have been applied, because $\xi \models \text{false}$ does not hold. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$. We know that $\sigma \cup \xi^{\dot{C}L} \models \text{true}$. Using Rule 4, we obtain $(C, J, L, R) \Vdash \langle \text{true}, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Second, we assume $(C, J, L, R) \Vdash \langle \text{true}, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means Rule 4 has been applied necessarily. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \text{true}$. We also know that $\sigma \cup \xi^{\dot{C}L} \models \neg\text{false}$. Using Rule 24 we get $(C, J, L, R) \Vdash \langle \text{false} \to p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. ⊠

**Lemma 14** *For arbitrary closed process term $p$ and arbitrary guard $b$ we have*

$$b \to \bot \leftrightarrow \neg b.$$

*Proof.* Let $R = \{(b \to \bot, \neg b) \mid p \in P, \text{guard } b\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since there are no action transition rules defined for $\bot$, also $b \to \bot$ has no action transition rules defined, and there are no action transition rules defined for delay predicates, the conditions 1 – 3 hold trivially.

*Condition 4*: We assume $(C, J, L, R) \Vdash \langle b \to \bot, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $(C, J, L, R), \sigma, t, \rho, k_1, \sigma'$, which means that either Rule 21 or Rule 22 has been applied necessarily. Then we can distinguish two cases:

1. Rule 21 has been applied. Then, $(C, J, L, R) \Vdash \langle \bot, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ for some $k_p$. This leads a contradiction, because $\bot$ cannot perform any action transitions. Thus, Rule 21 cannot have been applied.
2. Rule 22 has been applied. Then, $k_1 \equiv b \to \bot$ and $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma Et}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow (C, J, L, R) \Vdash \langle \bot, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ for some $z, \sigma''$ and $\rho(t) \models b (C, J, L, R) \Vdash\!\Rightarrow \langle \bot, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow (C, J, L, R) \Vdash \langle \bot, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ and $\rho(t) \models b \Rightarrow (C, J, L, R) \Vdash \langle \bot, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(0) \models \neg b$ and $\rho(t) \models \neg b$, since the right-hand side of these implications are false (due to $\bot$ cannot perform any transition). Thus, we have $\forall_{s\in[0,t]}\ \rho(s) \models \neg b$. Hence, we can also obtain the following transition $(C, J, L, R) \Vdash \langle \neg b, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle \neg b, \rho_\sigma(t)\rangle$ (see also Rule 3). Take $k_2 \equiv \neg b$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $(C, J, L, R) \Vdash \langle \neg b, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $(C, J, L, R), \sigma, t, \rho, k_1, \sigma'$, which means that Rule 3 has been applied necessarily. Then, $k_1 \equiv \neg b$, $\rho \in \Omega(\sigma, C, J, L, \neg b, t)$ and $\sigma' = \rho_\sigma(t)$. From $\rho \in \Omega(\sigma, C, J, L, \neg b, t)$, we know that $\forall_{s\in[0,t]}\ \rho(s) \models \neg b$ and $\bot$ also cannot perform any transition. Then the following premises $\rho(0) \models b \Rightarrow (C, J, L, R) \Vdash \langle \bot, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ for some $z, \sigma''$, and $\rho(t) \models b \Rightarrow (C, J, L, R) \Vdash \langle \bot, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ hold (because the left-hand side of the implications are false). Using Rule 22, we obtain $(C, J, L, R) \Vdash \langle b \to \bot, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to \bot, \rho_\sigma(t)\rangle$. Take $k_2 \equiv b \to \bot$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle b \to \bot, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means that Rule 24 has been applied necessarily. Notice that Rule 23 cannot be applied, because the premise $(C, J, L, R) \Vdash \langle \bot, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ does not hold. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Applying Rule 4, we get $(C, J, L, R) \Vdash \langle \neg b, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Second, we assume $(C, J, L, R) \Vdash \langle \neg b, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means Rule 4 has been applied necessarily. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24, we obtain $(C, J, L, R) \Vdash \langle b \to \bot, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. ⊠

**Lemma 15 (Distributivity of guard operator over alternative composition)** *For arbitrary closed process terms $p$ and $q$ and arbitrary guard $b$ we have*

$$b \to (p \,[]\, q) \leftrightarrows b \to p \,[]\, b \to q.$$

*Proof.* Let $R = \{(b \to (p \,[]\, q), b \to p \,[]\, b \to q) \mid p, q \in P, \text{guard } b\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: First, we assume $E \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means that Rule 20.1 has been applied necessarily. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $\xi \models b$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, we distinguish two cases:

1. Rule 25.1.l has been applied. Then $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 20.1, we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. We also obtain $E \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23. Applying Rule 25.1.l, we get $E \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.
2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

Second, we assume $(C, J, L, R) \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $(C, J, L, R), \sigma, \xi, a, \xi', \sigma'$, which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. We distinguish two cases:

1. Rule 25.1.l has been applied. Then $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 20.1, we obtain $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$ and $\xi \models b$. For $(C, J, L, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means that either Rule 23 or Rule 24 has been applied necessarily. We distinguish two cases:
   (a) Rule 23 has been applied. Then $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 25.1.l, we can have $(C, J, L, R) \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$. Due to Rule 20.1, we finally get $(C, J, L, R) \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$.
   (b) Rule 24 has been applied. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. This leads to a contradiction. Therefore this case cannot occur.
2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

*Condition 2*: We assume $E \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1\sigma'$, which means that Rule 20.2 has been applied necessarily. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models b$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, we distinguish two cases:

1. Rule 25.2.l has been applied. Then $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 20.1, we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. We also obtain $E \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23. Applying Rule 25.2.l, we get $E \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.
2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

*Condition 3*: We assume $(C, J, L, R) \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that Rule 25.1.l or Rule 25.1.r has been applied necessarily. We distinguish two cases:

1. Rule 25.2.l has been applied. Then $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 20.2, we obtain $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $\xi \models b$. For $(C, J, L, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means that either Rule 23 or Rule 24 has been applied necessarily. We distinguish two cases:
   (a) Rule 23 has been applied. We have $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 23. Applying Rule 25.2.l, we can have $(C, J, L, R) \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$. Due to Rule 20.2, we finally get $(C, J, L, R) \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.
   (b) Rule 24 has been applied. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. This leads to a contradiction. Therefore this case cannot occur.

2. Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.

*Condition 4*: We assume $E \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that either Rule 21 or Rule 22 has been applied necessarily. Then we can distinguish two cases:

1. Rule 21 has been applied. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ for some $k_{pq}$ such that $k_1 \equiv b \to k_{pq}$ and $\forall_{s\in[0,t]}\ \rho(s) \models b$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, we get $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some $k_p, k_q$ such that $k_{pq} \equiv k_p \,[]\, k_q$ (using Rule 26). Applying Rule 21, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p, \sigma'\rangle$ and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_q, \sigma'\rangle$. According to Rule 26, we have $E \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p \,[]\, b \to k_q, \sigma'\rangle$. Note that $k_1 \equiv b \to k_p \,[]\, k_q$. Take $k_2 \equiv b \to k_p \,[]\, b \to k_q$ and observe that $(k_1, k_2) \in R$.

2. Rule 22 has been applied. Then, $k_1 \equiv b \to (p \,[]\, q)$ and $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}, \forall_{s\in(0,t)}\ \rho(s) \models \neg b, \rho(0) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$ for some $z, \sigma''$ and $\rho(t) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z, \sigma''\rangle$, we can also have $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p_z, \sigma''\rangle$ for some $p_z$, and $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma''\rangle$ for some $q_z$ (see also Rule 26). From $\rho(t) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we also get $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (see also Rule 27). Using Rule 22, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p, \rho_\sigma(t)\rangle$ and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to q, \rho_\sigma(t)\rangle$. According to Rule 26, we obtain $E \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p \,[]\, b \to q, \rho_\sigma(t)\rangle$. Take $k_2 \equiv b \to p \,[]\, b \to q$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $E \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that Rule 26 has been applied necessarily. Then $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$, and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some $k_p, k_q$ such that $k_1 \equiv k_p \,[]\, k_q$. For $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$, and $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$, four cases can be distinguished:

1. Rule 21 has been applied for both. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_p, \sigma'\rangle$, $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_q, \sigma'\rangle$ for some $k'_p, k'_q$ such that $k_p \equiv b \to k'_p$, $k_q \equiv b \to k'_q$, and $\forall_{s\in[0,t]}\ \rho(s) \models b$. Using Rule 26, we obtain $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_p \,[]\, k'_q, \sigma'\rangle$. Applying Rule 21, we get $E \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k'_p \,[]\, k'_q, \sigma'\rangle$. Note that $k_1 \equiv b \to k'_p \,[]\, b \to k'_q$. Take $k_2 \equiv b \to k'_p \,[]\, k'_q$ and observe that $(k_2, k_1) \in R$.

2. Rule 22 has been applied for both. Then, $k_p \equiv b \to p$, $k_q \equiv b \to q$ and $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p_z, \sigma''\rangle$, $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma'''\rangle$, for some $p_z, q_z, \sigma'', \sigma'''$, $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p_z, \sigma''\rangle$, and

$\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma'''\rangle$, by Lemma 1 we know that $\sigma'' = \sigma''' = \rho_\sigma(0)$, we get $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle z', \sigma''\rangle$ for some $z'$ (see also Rule 26). From $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ and $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(t) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (see also Rule 27). Using Rule 22, we obtain $E \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to (p \,[]\, q), \rho_\sigma(t)\rangle$. Notice that $k_1 \equiv b \to p \,[]\, b \to q$. Take $k_2 \equiv b \to (p \,[]\, q)$ and observe that $(k_2, k_1) \in R$.

3. Rule 21 has been applied for $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$, and Rule 22 has been applied for $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k'_p, \sigma'\rangle$ for some $k'_p$ such that $k_p \equiv b \to k'_p$, $\sigma' = \rho_\sigma(t)$, $\forall_{s\in[0,t]}\ \rho(s) \models b$, $\rho \in \Omega_{\sigma Et}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma''\rangle$, for some $q_z, \sigma''$, $\rho(t) \models b \Rightarrow E \Vdash \langle q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, $k_q \equiv b \to q$, and $k_1 \equiv b \to k'_p \,[]\, b \to q$. From $\forall_{s\in[0,t]}\ \rho(s) \models b$, and $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, this leads to a contradiction, unless $t = 0$. Hence, $t = 0$. Then we consider only the case in which $t = 0$. From $E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k'_p, \sigma'\rangle$, $\rho(0) \models b$, and $\rho(0) \models b \Rightarrow E \Vdash \langle q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle q_z, \sigma''\rangle$, it is not hard to see that we get $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k_z, \sigma'''\rangle$ for some $k_z, \sigma'''$. We know that $\sigma' = \sigma'' = \sigma''' = \rho_\sigma(0)$ (see also Rule 26 and Lemma 1). Since $\rho \in \Omega_{\sigma Et}$, we have $\sigma = \rho_\sigma(0)$. Also, from $E \Vdash \langle p, \rho_\sigma(0)\rangle \overset{0,\rho}{\longmapsto} \langle k'_p, \sigma'\rangle$, we have $E \Vdash \langle p, \rho_\sigma(0)\rangle \overset{\rho(0)}{\rightsquigarrow}$ (by Lemma 3). Using Rule 26, we $\rho(0) \models b \Rightarrow E \Vdash \langle p \,[]\, q, \rho_\sigma(0)\rangle \overset{\rho(0)}{\rightsquigarrow}$. Applying Rule 22, we obtain $E \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \overset{0,\rho}{\longmapsto} \langle b \to (p \,[]\, q), \rho_\sigma(0)\rangle$. Take $k_2 \equiv b \to (p \,[]\, q)$ and observe that $(k_2, k_1) \in R$.

4. Rule 21 has been applied for $E \Vdash \langle b \to q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$, and Rule 22 has been applied for $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$. The proof is similar to the previous case.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle b \to (p \,[]\, q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means that Rule 23 or Rule 24 has been applied necessarily. Then, we distinguish two cases:

1. Rule 23 has been applied. Then $(C, J, L, R) \Vdash \langle p \,[]\, q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. Using Rule 27, we have $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 23, we obtain $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 27, we get $(C, J, L, R) \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.

2. Rule 24 has been applied. Then $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24, we can have $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle b \to q, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Applying Rule 27, we get $(C, J, L, R) \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Second, we assume $(C, J, L, R) \Vdash \langle b \to p \,[]\, b \to q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means Rule 27 has been applied necessarily. Then, $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$

and $(C, J, L, R) \Vdash \langle b \to q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. For $(C, J, L, R) \Vdash \langle b \to p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle b \to q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$, four cases can be distinguished:

1. Rule 23 has been applied for both. Then, $(C, J, L, R) \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. According to Rule 27, we obtain $(C, J, L, R) \Vdash \langle p \,[]\, q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 23, we get $(C, J, L, R) \Vdash \langle b \to (p \,[]\, q), \sigma \rangle \overset{\xi}{\rightsquigarrow}$.
2. Rule 24 has been applied for both. Then $\xi \equiv \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. According to Rule 24, we can have $(C, J, L, R) \Vdash \langle b \to (p \,[]\, q), \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}$.
3. Rule 23 has been applied for $(C, J, L, R) \Vdash \langle b \to p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ and Rule 24 has been applied for $(C, J, L, R) \Vdash \langle b \to q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Then, $(C, J, L, R) \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$, $\xi \models b$, and $\xi = \sigma \cup \xi^{\dot{C}L}$, and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. This leads to a contradiction. Therefore, Rule 23 cannot have been applied for $(C, J, L, R) \Vdash \langle b \to p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ and Rule 24 cannot have been applied for $(C, J, L, R) \Vdash \langle b \to q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$.
4. Rule 23 has been applied for $(C, J, L, R) \Vdash \langle b \to q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ and Rule 24 has been applied for $(C, J, L, R) \Vdash \langle b \to p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. The proof is similar to the previous case.

⊠

## B.4 Properties of sequential composition

**Lemma 16 (Left-zero element for sequential composition)** *For arbitrary closed process term $p$ we have*

$$\delta;\ p \leftrightarrow \delta.$$

*Proof.* Let $R = \{(\delta;\ p, \delta) \mid p \in P\}$. Since there are no action transition rules and time transition rules defined for $\delta$, and therefore also not for $\delta;\ p$, the conditions 1 – 5 hold trivially.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle \delta;\ p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means that Rule 19 has been applied necessarily. Then, $(C, J, L, R) \Vdash \langle \delta, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $(C, J, L, R) \Vdash \langle \delta, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $(C, J, L, R) \Vdash \langle \delta;\ p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. ⊠

**Lemma 17 (Associativity of sequential composition)** *For arbitrary closed process terms $p$, $q$ and $r$ we have*

$$(p;\ q);\ r \leftrightarrow p;\ (q;\ r).$$

*Proof.* Let $R = \{((p;\ q);\ r,\ p;\ (q;\ r)) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proofs of conditions 4 and 5 are similar to the proofs of conditions 2 and 3 (except Rule 16 has not been applied, because no $\chi$ process can transform to terminated process by means of time transitions) since the deduction rules for non-terminating action transitions and time transitions of ; are similar.

*Condition 1*: Since there are no termination transitions defined for $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, condition 1 holds trivially.

*Condition 2*: We assume $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we distinguish two cases:

1. Rule 16 has been applied. Then $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. This leads to a contradiction as there is no deduction rule that allows a sequential composition to perform a termination transition. Hence, this case cannot occur.

2. Rule 17 has been applied. Then, $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$ for some $k_1'$ such that $k_1 \equiv k_1';\ r$. We distinguish two cases for $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1', \sigma'\rangle$:

   (a) Rule 16 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1' \equiv q$. According to Rule 19, we have $E \Vdash \langle q;\ r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 16, we have $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q;\ r, \sigma'\rangle$. Note that $k_1 \equiv q;\ r$. Take $k_2 \equiv q;\ r$ and observe that $(k_1, k_2) \in R$.

   (b) Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1' \equiv k_p;\ q$. Using Rule 17 we obtain $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ (q;\ r), \sigma'\rangle$. Note that $k_1 \equiv (k_p;\ q);\ r$. Take $k_2 \equiv k_p;\ (q;\ r)$ and observe that $(k_1, k_2) \in R$.

*Condition 3*: We assume $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we distinguish two cases:

1. Rule 16 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$, $E \Vdash \langle q;\ r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1 \equiv q;\ r$. According to Rule 19, we have $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 16, we obtain $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Using Rule 17, we obtain $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q;\ r, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

2. Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv k_p;\ (q;\ r)$. Using Rule 17, we obtain $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$. Again, using Rule 17, we obtain $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle (k_p;\ q);\ r, \sigma'\rangle$. Take $k_2 \equiv (k_p;\ q);\ r$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means that Rule 19 has applied necessarily. Then $E \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 19, we get $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p;\ (q;\ r), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means that Rule 19 has applied necessarily. Then $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 19, we get $E \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 19, we obtain $E \Vdash \langle (p;\ q);\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. ⊠

**Lemma 18 (Distribution of sequential over alternative composition)** *For arbitrary closed process terms p, q and r we have*

$$(p \,[]\, q);\ r \leftrightarrows p;\ r \,[]\, q;\ r.$$

*Proof.* Let $R = \{((p \,[]\, q);\ r,\ p;\ r \,[]\, q;\ r) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: Since there no action transition rules defined for any closed process term $k_1$ and $k_2$ such that $E \Vdash \langle k_1;\ k_2, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma'\rangle$, condition 1 holds trivially.

*Condition 2*: We assume $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$; which means that either Rule 16 or Rule 17 has been applied necessarily. Hence, we can distinguish two cases:

1. Rule 16 has been applied. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma'\rangle$, and $E \Vdash \langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma'\rangle$ we again distinguish two cases:
   (a) Rule 25.1.l has been applied. Then, $k_1 \equiv r$, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle \surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 16, we get $E \Vdash \langle p;\ r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle r, \sigma'\rangle$. Due to Rule 19, we have $E \Vdash \langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 25.2.l, we have $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle r, \sigma'\rangle$ and observe that $(r, r) \in R$.
   (b) Rule 25.1.r has been applied. The proof is similar to the proof of the previous case.
2. Rule 17 has been applied. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1', \sigma'\rangle$ for some $k_1'$ such that $k_1 \equiv k_1';\ r$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1', \sigma'\rangle$ we can further distinguish two cases:
   (a) Rule 25.2.l has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1', \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 17, we get $E \Vdash \langle p;\ r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1';\ r, \sigma'\rangle$. Using Rule 19, we get $\langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 25.2.l, we have $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \xrightarrow{\xi, a, \xi'} \langle k_1';\ r, \sigma'\rangle$. Take $k_2 \equiv k_1';\ r$ and observe that $(k_1, k_2) \in R$.
   (b) Rule 25.2.r has been applied. The proof is similar to the proof of the previous case.

*Condition 3*: We assume $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$; which means that either Rule 25.2.l or Rule 25.2.r has been applied necessarily. Hence, we can distinguish two cases:

1. Rule 25.2.l has been applied. Then, $E \Vdash \langle p;\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ and $E \Vdash \langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For $E \Vdash \langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, we also get $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ using Rule 19. For $E \Vdash \langle p;\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ we again distinguish two cases:

   (a) Rule 16 has been applied. Then, $k_1 \equiv r$, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, and $\langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Applying Rule 25.1.l, we get $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$. According to Rule 16, we have $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle r, \sigma'\rangle$ and observe that $(r, r) \in R$.

   (b) Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv k_p;\ r$. Using Rule 25.2.l, we get $E \Vdash \langle p \,[]\, q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$. According to Rule 17, we have $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ r, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

2. Rule 25.2.r has been applied. The proof is similar to the proof of the previous case.

*Condition 4*: We assume $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$; which means Rule 18 has been applied necessarily. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$ for some $k_{pq}$ such that $k_1 \equiv k_{pq};\ r$. For $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pq}, \sigma'\rangle$, Rule 26 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some $k_p, k_q$ such that $k_{pq} \equiv k_p \,[]\, k_q$. Using Rule 18, we obtain $E \Vdash \langle p;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p;\ r, \sigma'\rangle$ and $E \Vdash \langle q;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q;\ r, \sigma'\rangle$. According to Rule 26 we obtain $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p;\ r \,[]\, k_q;\ r, \sigma'\rangle$. Note that $k_1 \equiv (k_p \,[]\, k_q);\ r$. Take $k_2 \equiv k_p;\ r \,[]\, k_q;\ r$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$; which means Rule 26 has been applied necessarily. Then, $E \Vdash \langle p;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pr}, \sigma'\rangle$ and $E \Vdash \langle q;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{qr}, \sigma'\rangle$ for some $k_{pr}, k_{qr}$ such that $k_1 \equiv k_{pr} \,[]\, k_{qr}$. For $E \Vdash \langle p;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{pr}, \sigma'\rangle$ and $E \Vdash \langle q;\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_{qr}, \sigma'\rangle$, Rule 18 has been applied to both. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $E \Vdash \langle q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_q, \sigma'\rangle$ for some $k_p, k_q$ such that $k_{pr} \equiv k_p;\ r$ and $k_{qr} \equiv k_q;\ r$. Using Rule 26 we then obtain $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p \,[]\, k_q, \sigma'\rangle$. Applying Rule 19, we get $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle (k_p \,[]\, k_q);\ r, \sigma'\rangle$. Note that $k_1 \equiv k_p;\ r \,[]\, k_q;\ r$. Take $k_2 \equiv (k_p \,[]\, k_q);\ r$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$; which means Rule 19 has been applied necessarily. Then, $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we have $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 19, we get $E \Vdash \langle p;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. According to Rule 27, we get $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p;\ r \,[]\, q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for

some $E, \sigma, \xi$; which means Rule 27 has been applied necessarily. Then, $E \Vdash \langle p;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q;\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Due to Rule 19, we obtain $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $E \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 27, we get $E \Vdash \langle p \,[]\, q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 19, we have $E \Vdash \langle (p \,[]\, q);\ r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. ⊠

**Lemma 19** *For arbitrary closed process terms $p$ and $q$ and arbitrary guard $b$ we have*

$$b \to (p;\ q) \leftrightarrows b \to p;\ q.$$

*Proof.* Let $R = \{(b \to (p;\ q), b \to p;\ q) \mid p, q \in P, \text{guard } b\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: We assume $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means that Rule 20.1 has been applied necessarily. Then, $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$ and $\xi \models b$. This leads to a contradiction as there is no deduction rule that allows a sequential composition to perform a termination transition. Second, we assume $E \Vdash \langle b \to p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$. This also leads to a contradiction as there is no deduction rule that allows a sequential composition to perform a termination transition. Thus, condition 1 holds trivially.

*Condition 2*: We assume $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that Rule 20.2 has been applied necessarily. Then, $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, and $\xi \models b$. For $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$, two cases can be distinguished:

1. Rule 16 has been applied. Then, $k_1 \equiv q$, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 20.1 we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$. Applying Rule 16 we have $E \Vdash \langle b \to p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$ and observe that $(q, q) \in R$.

2. Rule 17 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_q$ such that $k_1 \equiv k_p;\ q$. Using Rule 20.2 we have $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$, and using Rule 17 we have $E \Vdash \langle b \to p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 3*: We assume $E \Vdash \langle b \to p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means that Rule 16 or Rule 17 has been applied necessarily. Then, we distinguish two cases:

1. Rule 16 has been applied. Then, $k_1 \equiv q$, $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$ and $E \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. According to Rule 20.1 we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$ and $\xi \models b$. Applying Rule 16 we have $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Using Rule 20.2 we get $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$ and observe that $(q, q) \in R$.

2. Rule 17 has been applied. Then $E \Vdash \langle b \to p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv k_p;\ q$. Using Rule 20.2, we obtain $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ and $\xi \models b$. Using Rule 17, we get $E \Vdash \langle p;\ q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$. Applying Rule 20.2, we have $E \Vdash \langle b \to (p;\ q), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p;\ q, \sigma'\rangle$ and observe that $(k_1, k_1) \in R$.

*Condition 4*: We assume $E \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that either Rule 21 or Rule 22 has been applied necessarily. Then we can distinguish two cases:

1. Rule 21 has been applied. Then, $E \Vdash \langle p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$ and $\forall_{s\in[0,t]}\ \rho(t) \models b$ such that $k_1 \equiv b \to k_1'$. For $E \Vdash \langle p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$, we get $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1' \equiv k_p;\ q$ using Rule 18. Applying Rule 21, we get $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to k_p, \sigma'\rangle$. According to Rule 18, we obtain $E \Vdash \langle (b \to p);\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle (b \to k_p);\ q, \sigma'\rangle$. Note that $k_1 \equiv b \to (k_p;\ q)$. Take $k_2 \equiv (b \to k_p);\ q$ and observe that $(k_1, k_2) \in R$.
2. Rule 22 has been applied. Then, $k_1 \equiv b \to (p;\ q)$, $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle p;\ q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, for some $p', \sigma''$, $\rho(t) \models b \Rightarrow E \Vdash \langle p;\ q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. For $\rho(0) \models b \Rightarrow E \Vdash \langle p;\ q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, we get $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k_p', \sigma''\rangle$, for some $k_p'$ (see also Rule 18). For $\rho(t) \models b \Rightarrow E \Vdash \langle p;\ q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ (see also Rule 19). Using Rule 22, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to p, \rho_\sigma(t)\rangle$. Using Rule 18, we obtain $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle (b \to p);\ q, \rho_\sigma(t)\rangle$. Take $k_2 \equiv (b \to p);\ q$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $E \Vdash \langle (b \to p);\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means that Rule 18 has been applied necessarily. Then $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$ for some $k_1'$ such that $k_1 \equiv k_1';\ q$. For $E \Vdash \langle b \to p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_1', \sigma'\rangle$, two cases can be distinguished:

1. Rule 21 has been applied. Then, $E \Vdash \langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p, \sigma'\rangle$ and $\forall_{s\in[0,t]}\ \rho(t) \models b$ for some $k_p$ such that $k_1' \equiv b \to k_p$. Using Rule 18, we get $E \Vdash \langle p;\ q, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle k_p;\ q, \sigma'\rangle$. According to Rule 21, we have $E \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to (k_p;\ q), \sigma'\rangle$. Note that $k_1 \equiv (b \to k_p);\ q$. Take $k_2 \equiv b \to (k_p;\ q)$ and observe that $(k_2, k_1) \in R$.
2. Rule 22 has been applied. Then, $k_1' \equiv b \to p$, $\sigma' = \rho_\sigma(t)$, $\rho \in \Omega_{\sigma E t}$, $\forall_{s\in(0,t)}\ \rho(s) \models \neg b$, $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, for some $p', \sigma''$, and $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$. From $\rho(0) \models b \Rightarrow E \Vdash \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma''\rangle$, we get $\rho(0) \models b \Rightarrow E \Vdash \langle p;\ q, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle k_p', \sigma''\rangle$ for some $k_p'$. From $\rho(t) \models b \Rightarrow E \Vdash \langle p, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$, we get $\rho(t) \models b \Rightarrow E \Vdash \langle p;\ q, \rho_\sigma(t)\rangle \overset{\rho(t)}{\rightsquigarrow}$ using Rule 19. Applying Rule 22, we obtain $E \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \to (p;\ q), \rho_\sigma(t)\rangle$. Note that $k_1 \equiv (b \to p);\ q$. Take $k_2 \equiv b \to (p;\ q)$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means that either Rule 23 or Rule 24 has been applied necessarily. Then we can distinguish two cases:

1. Rule 23 has been applied. Then, $(C, J, L, R) \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. Rule 19, we obtain $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 23, we get $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 19, we obtain $(C, J, L, R) \Vdash \langle (b \to p);\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$.
2. Rule 24 has been applied. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24 we get $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. Applying Rule 19, we obtain $(C, J, L, R) \Vdash \langle (b \to p);\ q, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

Second, we assume $(C, J, L, R) \Vdash \langle (b \to p);\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R), \sigma, \xi$, which means that Rule 19 has been applied necessarily. Then $(C, J, L, R) \Vdash \langle b \to p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. For this, we can distinguish two cases:

1. Rule 23 has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $\xi \models b$. Using Rule 19, we obtain $(C, J, L, R) \Vdash \langle p;\ q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Applying Rule 23, we get $(C, J, L, R) \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{\xi}{\rightsquigarrow}$.
2. Rule 24 has been applied. Then, $\xi = \sigma \cup \xi^{\dot{C}L}$ and $\sigma \cup \xi^{\dot{C}L} \models \neg b$. Using Rule 24 we get $(C, J, L, R) \Vdash \langle b \to (p;\ q), \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$.

⊠

## B.5 Properties of parallel composition

**Lemma 20 (Commutativity of parallel composition)** *For arbitrary closed process terms $p$ and $q$ we have*

$$p \parallel q \leftrightarrows q \parallel p.$$

*Proof.* Let $R = \{(p \parallel q, q \parallel p) \mid p, q \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. Since the deduction rules for $\parallel$ are symmetrical w.r.t. the left and right argument, obviously all conditions are met. ⊠

**Lemma 21 (Associativity of parallel composition)** *For arbitrary closed process terms $p$, $q$ and $r$ we have*

$$(p \parallel q) \parallel r \leftrightarrows p \parallel (q \parallel r).$$

*Proof.* Let $R = \{((p \parallel q) \parallel r, p \parallel (q \parallel r)) \mid p, q, r \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$. The proof of the left implication of condition 1 is similar to the proof of the right implication of condition 1. The proof of conditions 3 is similar to the proof of conditions 2. The proofs of conditions 4 – 6 are the same as the proofs of conditions 4 – 6 of Lemma 11 (apart from the operator that has been used), because the time and consistency transition rules defined for the [] and $\parallel$ are the same. To increase the readability of this proof, we often apply Lemma 4 to obtain $(C, J \cup W, L, R) \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ from $(C, J, L, R) \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$ or the other way around without mentioning explicitly the use of the Lemma 4.

*Condition 1*: We assume $(C, J, L, R) \Vdash \langle (p \parallel q) \parallel r, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \checkmark, \sigma' \rangle$ for some $(C, J, L, R), \sigma, \xi, a, \xi', \sigma'$, which means that either Rule 28.1.l or Rule 28.1.r has been applied necessarily. Hence, we distinguish two cases:

1. Rule 28.1.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma \rangle$, $(C, J, L, R) \Vdash \langle r, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma' \rangle$ for some $W, h, cs$ and $a = ca(h, cs)$. Since we do not have a rule for $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma' \rangle$, we obtain a contradiction and the right implication of condition 1 holds trivially.

2. Rule 28.1.r has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle r, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma' \rangle$, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma' \rangle$ for some $W, h, cs$, and $a = ca(h, cs)$. Since we do not have a rule for $(C, J, L, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma' \rangle$, we obtain a contradiction and the right implication of condition 1 holds trivially.

*Condition 2*: We assume $(C, J, L, R) \Vdash \langle (p \parallel q) \parallel r, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle k_1, \sigma' \rangle$ for some $(C, J, L, R), \sigma, \xi, a, \xi', k_1, \sigma'$. Based on the deduction rule that has been applied we can distinguish ten cases:

1. Rule 28.2.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle k_1, \sigma' \rangle$ and $(C, J, L, R) \Vdash \langle r, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma' \rangle$ for some $W, h, cs$, and $a = ca(h, cs)$. For $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle k_1, \sigma' \rangle$ we can distinguish four more cases:

   (a) Rule 29.1.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma \rangle$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma' \rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv q$. Using Rule 29.1.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \langle q, \sigma' \rangle$. Using Rule 28.3.l we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma \rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle q, \sigma' \rangle$, and observe that $(q, q) \in R$.

   (b) Rule 29.1.r has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \langle \checkmark, \sigma' \rangle$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma' \rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv p$. Using Rule 28.1.l we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma \rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle \checkmark, \sigma' \rangle$.

Using Rule 29.1.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle p, \sigma'\rangle$ and observe that $(p, p) \in R$.

(c) Rule 29.2.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv k_p \parallel q$, and $(C, J \cup W, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.1.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle q, \sigma'\rangle$. Using Rule 28.4.l we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle k_p \parallel q, \sigma'\rangle$. Take $k_2 \equiv k_p \parallel q$ and observe that $(k_1, k_2) \in R$.

(d) Rule 29.2.r has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle k_q, \sigma'\rangle$ for some $k_q$ such that $k_1 \equiv p \parallel k_q$, and $(C, J \cup W, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.2.l we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle k_q, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle p \parallel k_q, \sigma'\rangle$. Take $k_2 \equiv p \parallel k_q$ and observe that $(k_1, k_2) \in R$.

2. Rule 28.2.r has been applied. Then, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$ and $(C, J \cup W, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle k_1, \sigma'\rangle$ for some $W, h, cs$, and $a = ca(h, cs)$. The conclusion $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$ cannot be obtained from the deduction rules. Hence, this case cannot occur.

3. Rule 28.3.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle \checkmark, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle k_1, \sigma'\rangle$ for some $W, h, cs$, and $a = ca(h, cs)$. The conclusion $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle \checkmark, \sigma'\rangle$ cannot be obtained from the deduction rules. Hence, this case cannot occur.

4. Rule 28.3.r has been applied. Then, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle k_1, \sigma'\rangle$ and $(C, J \cup W, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle \checkmark, \sigma'\rangle$ for some $W, h, cs$, and $a = ca(h, cs)$. For $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle k_1, \sigma'\rangle$ we can distinguish four more cases:

(a) Rule 29.1.l has been applied. Then, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_1 \equiv q$. Using Rule 29.1.r we obtain $(C, J \cup W, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi, \text{isa}(h,cs), \xi'} \langle q, \sigma'\rangle$. Using Rule 28.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle q, \sigma'\rangle$ and observe that $(q, q) \in R$.

(b) Rule 29.1.r has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi, \text{ira}(h,cs,W), \xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv p$. Using Rule 28.1.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle \checkmark, \sigma'\rangle$. Using Rule 29.1.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi, ca(h,cs), \xi'} \langle p, \sigma'\rangle$ and observe that $(p, p) \in R$.

(c) Rule 29.2.l has been applied. Then, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv k_p \parallel q$, and $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.1.r we obtain $(C, J \cup W, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q, \sigma'\rangle$. Using Rule 28.4.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_p \parallel q, \sigma'\rangle$. Take $k_2 \equiv k_p \parallel q$ and observe that $(k_1, k_2) \in R$.

(d) Rule 29.2.r has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_q, \sigma'\rangle$ for some $k_q$ such that $k_1 \equiv p \parallel k_q$, and $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.3.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_q, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle p \parallel k_q, \sigma'\rangle$. Take $k_2 \equiv p \parallel k_q$ and observe that $(k_1, k_2) \in R$.

5. Rule 28.4.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_r, \sigma'\rangle$ for some $W, h, cs, k_{pq}, k_r$ such that $k_1 \equiv k_{pq} \parallel k_r$, and $a = ca(h, cs)$. For $(C, J \cup W, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$ four cases can be distinguished:

(a) Rule 29.1.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv q$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.3.l we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel k_r$. Take $k_2 \equiv q \parallel k_r$ and observe that $(k_1, k_2) \in R$.

(b) Rule 29.1.r has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv p$. Using Rule 28.3.l we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv p \parallel k_r$. Take $k_2 \equiv p \parallel k_r$ and observe that $(k_1, k_2) \in R$.

(c) Rule 29.2.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_{pq} \equiv k_p \parallel q$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.4.l we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_p \parallel (q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel q) \parallel k_r$. Take $k_2 \equiv k_p \parallel (q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

(d) Rule 29.2.r has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_q, \sigma'\rangle$ for some $k_q$ such that $k_{pq} \equiv p \parallel k_q$, and $(C, J \cup W, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.4.l we obtain

$(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_q \parallel k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel (k_q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (p \parallel k_q) \parallel k_r$. Take $k_2 \equiv p \parallel (k_q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

6. Rule 28.4.r has been applied. Then, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$, $(C, J \cup W, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_r, \sigma'\rangle$ for some $W, h, cs, k_{pq}, k_r$ such that $k_1 \equiv k_{pq} \parallel k_r$, and $a = ca(h, cs)$. For $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$ we can distinguish four more cases:

   (a) Rule 29.1.l has been applied. Then, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\checkmark, \sigma'\rangle$, $\langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_{pq} \equiv q$. Using Rule 29.2.r we obtain $(C, J \cup W, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel k_r$. Take $k_2 \equiv q \parallel k_r$ and observe that $(k_1, k_2) \in R$.

   (b) Rule 29.1.r has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_{pq} \equiv p$. Using Rule 28.2.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel k_r, \sigma'\rangle$. Notice that $k_1 \equiv p \parallel k_r$. Take $k_2 \equiv p \parallel k_r$ and observe that $(k_1, k_2) \in R$.

   (c) Rule 29.2.l has been applied. Then, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_{pq} \equiv k_p \parallel q$, and $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.r we obtain $(C, J \cup W, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 28.4.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_p \parallel (q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel q) \parallel k_r$. Take $k_2 \equiv k_p \parallel (q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

   (d) Rule 29.2.r has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_q, \sigma'\rangle$ for some $k_q$ such that $k_{pq} \equiv p \parallel k_q$, and $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 28.4.r we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle k_q \parallel k_r, \sigma'\rangle$. Using Rule 29.2.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle p \parallel (k_q \parallel k_r), \sigma'\rangle$. Notice that $k_1 \equiv (p \parallel k_q) \parallel k_r$. Take $k_2 \equiv p \parallel (k_q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

7. Rule 29.1.l has been applied. Then, $(C, J, L, R) \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$,$(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark, \sigma'\rangle$, $k_1 \equiv r$, and $(C, J, L, R) \Vdash \langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Then two cases can be considered:

   (a) Rule 28.1.l has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\checkmark, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\checkmark, \sigma'\rangle$ for some $W, h, cs$ and $a = ca(h, cs)$.

Using Rule 29.1.l we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle r, \sigma'\rangle$. Using Rule 28.3.l we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle r, \sigma'\rangle$, and observe that $(r, r) \in R$.

(b) Rule 28.1.r has been applied. Then, $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\surd, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\surd, \sigma'\rangle$ for some $W$, $h$, $cs$, and $a = ca(h, cs)$. Using Rule 29.1.l we obtain $(C, J \cup W, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle r, \sigma'\rangle$. Using Rule 28.3.r we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle r, \sigma'\rangle$, and observe that $(r, r) \in R$.

8. Rule 29.1.r has been applied. Then, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\surd, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_1 \equiv p \parallel q$. According to Rule 31, we have $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.1.r, we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$. Using Rule 29.2.r, we obtain $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel q, \sigma'\rangle$. Take $k_2 \equiv p \parallel q$ and observe that $(k_1, k_2) \in R$.

9. Rule 29.2.l has been applied. Then, $(C, J, L, R) \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_{pq}, \sigma'\rangle$ for some $k_{pq}$ such that $k_1 \equiv k_{pq} \parallel r$, and $(C, J, L, R) \Vdash \langle r, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. For $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_{pq}, \sigma'\rangle$, ten cases can be distinguished.

(a) Rule 28.2.l has been applied. Then, $(C, J, L, R) \Vdash \langle r, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\surd, \sigma'\rangle$ for some $W$, $h$, $cs$, and $a = ca(h, cs)$. Then applying Rule 29.1.l followed by Rule 28.4.l gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

(b) Rule 28.2.r has been applied. Then $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\surd, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$ for some $W$, $h$, $cs$, and $a = ca(h, cs)$. Then applying Rule 29.1.l followed by Rule 28.4.r gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

(c) Rule 28.3.l has been applied. Then $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle\surd, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_{pq}, \sigma'\rangle$ for some $W$, $h$, $cs$, and $a = ca(h, cs)$. Then applying Rule 29.2.l followed by Rule 28.3.l gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

(d) Rule 28.3.r has been applied. Then $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_{pq}, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle\surd, \sigma'\rangle$ for some $W$, $h$, $cs$, $a = ca(h, cs)$. Then applying Rule 29.1.l followed by Rule 28.4.r gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_{pq} \parallel r, \sigma'\rangle$. Take $k_2 \equiv k_{pq} \parallel r$ and observe that $(k_1, k_2) \in R$.

(e) Rule 28.4.l has been applied. Then $(C, J \cup W, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_p, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_q, \sigma'\rangle$ for some $W, h, cs, k_p, k_q$ such that $k_{pq} \equiv k_p \parallel k_q$, and $a = ca(h, cs)$. Then applying Rule 29.2.l followed by Rule 28.4.l gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,ca(h,cs),\xi'} \langle k_p \parallel (k_q \parallel r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel k_q) \parallel r$. Take $k_2 \equiv k_p \parallel (k_q \parallel r)$ and observe that $(k_1, k_2) \in R$.

(f) Rule 28.4.r has been applied. Then $(C, J \cup W, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \langle k_q, \sigma'\rangle$ and $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \langle k_p, \sigma'\rangle$ for some $W, h, cs, k_p, k_q$ such that $k_{pq} \equiv k_p \parallel k_q$, and $a = ca(h, cs)$. Then applying Rule 29.2.l followed by Rule 28.4.r gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p \parallel (k_q \parallel r), \sigma'\rangle$. Notice that $k_1 \equiv (k_p \parallel k_q) \parallel r$. Take $k_2 \equiv k_p \parallel (k_q \parallel r)$ and observe that $(k_1, k_2) \in R$.

(g) Rule 29.1.l has been applied. Then $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$, and $k_{pq} \equiv q$. We have $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ (see Rule 31). Applying Rule 29.1.l gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q \parallel r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel r$. Take $k_2 \equiv q \parallel r$ and observe that $(k_1, k_2) \in R$.

(h) Rule 29.1.r has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma'\rangle$, $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $k_{pq} \equiv p$. Applying Rule 29.1.l and then Rule 29.2.r gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel r, \sigma'\rangle$. Notice that $k_1 \equiv q \parallel r$. Take $k_2 \equiv p \parallel r$ and observe that $(k_1, k_2) \in R$.

(i) Rule 29.2.l has been applied. Then, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_{pq} \equiv k_p \parallel q$, and $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. We have $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ (see Rule 31). Applying Rule 29.2.l gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p \parallel (q \parallel r), \sigma'\rangle$. Take $k_2 \equiv k_p \parallel (q \parallel r)$ and observe that $(k_1, k_2) \in R$.

(j) Rule 29.2.r has been applied. Then, $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_q, \sigma'\rangle$ for some $k_q$ such that $k_{pq} \equiv p \parallel k_q$, and $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Applying Rule 29.2.l and then Rule 29.2.r gives $(C, J, L, R) \Vdash \langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel (k_q \parallel r), \sigma'\rangle$. Notice $k_1 \equiv (p \parallel k_q) \parallel r$. Take $k_2 \equiv p \parallel (k_q \parallel r)$ and observe that $(k_1, k_2) \in R$.

10. Rule 29.2.r has been applied. Then, $(C, J, L, R) \Vdash \langle p \parallel q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_r, \sigma'\rangle$ for some $k_r$ such that $k_1 \equiv (p \parallel q) \parallel k_r$, and $(C, J, L, R) \Vdash \langle p \parallel q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. According to Rule 31, we have $(C, J, L, R) \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle q, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, $(C, J, L, R) \Vdash \langle p, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$ and $(C, J, L, R) \Vdash \langle q, \sigma'\rangle \overset{\xi'}{\rightsquigarrow}$. Using Rule 29.2.r, we obtain $(C, J, L, R) \Vdash \langle q \parallel r, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q \parallel k_r, \sigma'\rangle$. Using Rule 29.2.r, we obtain $(C, J, L, R) \Vdash$

$\langle p \parallel (q \parallel r), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle p \parallel (q \parallel k_r), \sigma'\rangle$. Take $k_2 \equiv p \parallel (q \parallel k_r)$ and observe that $(k_1, k_2) \in R$.

$\boxtimes$

## B.6 Properties of action encapsulation operator

**Lemma 22** *For arbitrary closed process terms $p$ we have*

$$\partial_\emptyset(p) \leftrightarrows p.$$

*Proof.* Let $R = \{(\partial_\emptyset(p), p) \mid p \in P\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: First, we assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$. We know that $a \notin \emptyset$. Using Rule 32.1, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

*Condition 2*: We assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \partial_\emptyset(k_p)$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.

*Condition 3*: We assume $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$. We know that $a \notin \emptyset$. Using Rule 32.2, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_\emptyset(k_1), \sigma'\rangle$. Take $k_2 \equiv \partial_\emptyset(k_1)$ and observe that $(k_2, k_1) \in R$.

*Condition 4*: We assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \partial_\emptyset(k_p)$. Take $k_2 \equiv k_p$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$. Using Rule 33, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \xmapsto{t,\rho} \langle \partial_\emptyset(k_1), \sigma'\rangle$. Take $k_2 \equiv \partial_\emptyset(k_1)$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$. Using Rule 34, we obtain $E \Vdash \langle \partial_\emptyset(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. $\boxtimes$

**Lemma 23** *For arbitrary closed process terms $p$, sets of actions $\mathcal{A}$ and $\mathcal{A}'$ we have*

$$\partial_\mathcal{A}(\partial_{\mathcal{A}'}(p)) \leftrightarrows \partial_{\mathcal{A}\cup\mathcal{A}'}(p).$$

*Proof.* Let $R = \{(\partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \partial_{\mathcal{A}\cup\mathcal{A}'}(p)) \mid p \in P, \text{sets of actions } \mathcal{A}, \mathcal{A}'\} \cup \{(i_d, i_d) \mid i_d \in P\}$.

*Condition 1*: First, we assume $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $a \notin \mathcal{A}$. Again, due to Rule 32.1, we have $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $a \notin \mathcal{A}'$. From $a \notin \mathcal{A}$ and $a \notin \mathcal{A}'$, we know that $a \notin \mathcal{A}\cup\mathcal{A}'$. Using Rule 32.1, we obtain $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Second, we assume $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', \sigma'$, which means Rule 32.1 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$ and $a \notin \mathcal{A}\cup\mathcal{A}'$. From $a \notin \mathcal{A}\cup\mathcal{A}'$, we know that $a \notin \mathcal{A}$ and $a \notin \mathcal{A}'$. Using Rule 32.1, we get $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$. Again, using Rule 32.1, we obtain $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \surd, \sigma'\rangle$.

*Condition 2*: We assume $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \partial_{\mathcal{A}}(k_p)$ and $a \notin \mathcal{A}$. Using Rule 32.2, we get $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k'_p, \sigma'\rangle$ for some $k'_p$ such that $k_p \equiv \partial_{\mathcal{A}'}(k'_p)$ and $a \notin \mathcal{A}'$. From $a \notin \mathcal{A}$ and $a \notin \mathcal{A}'$, we know that $a \notin \mathcal{A}\cup\mathcal{A}'$. Using Rule 32.2, we get $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(k'_p), \sigma'\rangle$. Note that $k_1 \equiv \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(k'_p))$. Take $k_2 \equiv \partial_{\mathcal{A}\cup\mathcal{A}'}(k'_p)$ and observe that $(k_1, k_2) \in R$.

*Condition 3*: We assume $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_1, \sigma'\rangle$ for some $E, \sigma, \xi, a, \xi', k_1, \sigma'$, which means Rule 32.2 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \partial_{\mathcal{A}\cup\mathcal{A}'}(k_p)$ and $a \notin \mathcal{A}\cup\mathcal{A}'$. From $a \notin \mathcal{A}\cup\mathcal{A}'$, we know that $a \notin \mathcal{A}$ and $a \notin \mathcal{A}'$. Using Rule 32.2, we get $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_{\mathcal{A}'}(k_p), \sigma'\rangle$. Again, due to Rule 32.2, we have $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(k_p)), \sigma'\rangle$. Take $k_2 \equiv \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(k_p))$ and observe that $(k_2, k_1) \in R$.

*Condition 4*: We assume $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \partial_{\mathcal{A}}(k_p)$. Again, due to Rule 33, we get $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k'_p, \sigma'\rangle$ for some $k'_p$ such that $k_p \equiv \partial_{\mathcal{A}'}(k'_p)$. Using Rule 33, we obtain $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma\rangle \xmapsto{t,\rho} \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(k'_p), \sigma'\rangle$. Note that $k_1 \equiv \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(k'_p))$. Take $k_2 \equiv \partial_{\mathcal{A}\cup\mathcal{A}'}(k'_p)$ and observe that $(k_1, k_2) \in R$.

*Condition 5*: We assume $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma\rangle \xmapsto{t,\rho} \langle k_1, \sigma'\rangle$ for some $E, \sigma, t, \rho, k_1, \sigma'$, which means Rule 33 has been applied necessarily. Then, $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle k_p, \sigma'\rangle$ for some $k_p$ such that $k_1 \equiv \partial_{\mathcal{A}\cup\mathcal{A}'}(k_p)$. Using Rule 33, we get $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \xmapsto{t,\rho} \langle \partial_{\mathcal{A}'}(k_p), \sigma'\rangle$. Again, due to Rule 33, we obtain $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \xmapsto{t,\rho} \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(k_p)), \sigma'\rangle$. Take $k_2 \equiv \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(k_p))$ and observe that $(k_2, k_1) \in R$.

*Condition 6*: First, we assume $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$, which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle \partial_{\mathcal{A}'}(p), \sigma\rangle \overset{\xi}{\rightsquigarrow}$. Again, due to Rule 34, we

have $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34 we obtain $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Second, we assume $E \Vdash \langle \partial_{\mathcal{A}\cup\mathcal{A}'}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}$ for some $E, \sigma, \xi$. which means Rule 34 has been applied necessarily. Then, $E \Vdash \langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34, we obtain $\langle E \Vdash \partial_{\mathcal{A}'}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}$. Using Rule 34 again, we obtain $E \Vdash \langle \partial_{\mathcal{A}}(\partial_{\mathcal{A}'}(p)), \sigma \rangle \overset{\xi}{\rightsquigarrow}$. ⊠

## B.7 Inconsistent process

**Lemma 24** *For arbitrary predicate $u$, using the previous properties gives,*

$$u \curvearrowright \bot \underline{\leftrightarrow} \bot.$$

*Proof.* $u \curvearrowright \bot \underline{\leftrightarrow} u \curvearrowright (\text{false} \curvearrowright p) \underline{\leftrightarrow} (u \wedge \text{false}) \curvearrowright p \underline{\leftrightarrow} \text{false} \curvearrowright p \underline{\leftrightarrow} \bot$.

⊠

**Lemma 25** *For arbitrary closed term $p$ we have*

$$p \,[]\, \bot \underline{\leftrightarrow} \bot.$$

*Proof.* Since there are no transition rules defined for $\bot$, also note that $p \,[]\, \bot$ has no transitions, the conditions 1 – 6 hold trivially. ⊠

**Lemma 26** *For arbitrary closed process term $p$ we have*

$$p \parallel \bot \underline{\leftrightarrow} \bot.$$

*Proof.* Since there are no transition rules defined for $\bot$, also note that $p \parallel \bot$ has no transitions, the conditions 1 – 6 hold trivially. ⊠

**Lemma 27** *For arbitrary set of actions $\mathcal{A}$ we have*

$$\partial_{\mathcal{A}}(\bot) \underline{\leftrightarrow} \bot.$$

*Proof.* Since there are no transition rules defined for $\bot$, also note that $\partial_{\mathcal{A}}(\bot)$ has no transitions, the conditions 1 – 6 hold trivially. ⊠

**Lemma 28** *For arbitrary closed process term $p$ we have*

$$\bot;\ p \underline{\leftrightarrow} \bot.$$

*Proof.* Since there are no transition rules defined for $\bot$, also note that $p;\ \bot$ has no transitions, the conditions 1 – 6 hold trivially. ⊠

**Lemma 29** *We have*

$$\text{skip};\ \bot \underline{\leftrightarrow} \delta.$$

*Proof.* We know that skip $\equiv \emptyset : \text{true} \gg \tau$. Let $R = \{(\emptyset : \text{true} \gg \tau;\ \bot, \delta)\}$. Since there are no action transition rules and time transition rules defined for $\delta$ and $\bot$, also $\emptyset : \text{true} \gg \tau;\ \bot$ cannot perform any action transitions (because $\bot$ is not consistent) and time transitions (because no time transition rules defined for $\emptyset : \text{true} \gg \tau$), the conditions 1 – 5 hold trivially.

*Condition 6*: First, we assume $(C, J, L, R) \Vdash \langle \emptyset : \text{true} \gg \tau;\ \bot, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ for some $(C, J, L, R)$, $\sigma$, $\xi$, which means that Rule 19 has been applied necessarily. Then, $(C, J, L, R) \Vdash \langle \emptyset : \text{true} \gg \tau, \sigma\rangle \overset{\xi}{\rightsquigarrow}$ such that $\xi = \sigma \cup \xi^{\dot{C}L}$ (see also Rule 2). Then, we get $(C, J, L, R) \Vdash \langle \delta, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$ using Rule 9. Second, we assume $(C, J, L, R) \Vdash \langle \delta, \sigma\rangle \overset{\xi}{\rightsquigarrow}$, which means that Rule 9 has been applied necessarily. Then $\xi = \sigma \cup \xi^{\dot{C}L}$. Using Rule 2, we get $(C, J, L, R) \Vdash \langle \emptyset : \text{true} \gg \tau, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. According to Rule 19, we obtain $(C, J, L, R) \Vdash \langle \emptyset : \text{true} \gg \tau;\ \bot, \sigma\rangle \overset{\sigma\cup\xi^{\dot{C}L}}{\rightsquigarrow}$. ⊠

**Lemma 30** *Using the previous properties gives,*

$$\bot \underline{\leftrightarrow} \text{false}.$$

*Proof.* $\bot \underline{\leftrightarrow} \text{false} \curvearrowright \text{false} \underline{\leftrightarrow} \text{false}$. ⊠