Syntax and Semantics of Timed 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

Contents

Chapter 1

Introduction

In this document, the timed χ (Chi) language is described. The timed χ language is obtained by means of simplification of hybrid χ (see [17]). The intended use of χ is for modeling, simulation, verification, and real-time control. Its application domain consists of large and complex manufacturing systems. Although the semantics is formally defined, the straightforward and elegant syntax and semantics is also highly suited to non-computer scientists. In the remainder of this report, we usually refer to timed χ as χ.

The most important concepts in χ are summarized below:

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

    - Strong time deterministic alternative composition operator. Where in the previous version of discrete-event χ [5] 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 χ semantics, the passage of time can never result in such a choice. In fact, the passage of time can only result in changes to the value of the predefined variable time. In the previous versions of χ, alternative composition $\Delta 5\ [] \ x := 1$ could non-deterministically choose between doing a delay of $t \leq 5$ to $\Delta(5-t)$, 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 $\Delta 5\ [] \ x := 1$ can only do the (non-delayable) action $x := 1$, and then terminate. Timed automata have a comparable choice mechanism, apart from initialization. In a timed automaton, action transitions cannot disappear as a result of time passing. They can only be disabled for the period of time that the associated 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 timed automaton as a result of time passing are changes in the values of the clocks. 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 discrete-event χ [5] had non-delayable guards, such as found in many process algebras, the current χ 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 α that must be switched on when the time becomes bigger then $t_{\max}$. Using a delayable guard, this can be modeled simply by $\mathsf{time} \geq t_{\max} \rightarrow \alpha := \mathrm{true}$.

  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 a 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:

  $$((\mathsf{time} \leq 3 \rightarrow h\,!\,\mathsf{time})\ []\ \Delta 10) \parallel \Delta 5;\ h\,?\,y.$$

  Using non-delayable guards, the process term can perform a delay of at most 5, and after performing an internal action transforms into

  $$(h\,!\,\mathsf{time}\ []\ \Delta 5) \parallel h\,?\,y.$$

  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 $\mathsf{time} \leq 3$.

  Using delayable guards on the other hand, the process term can do the delay of at most 3, and transforms into:

  $$(\mathsf{time} \leq 3 \rightarrow h\,!\,\mathsf{time}\ []\ \Delta 7) \parallel \Delta 2;\ h\,?\,y,$$

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

  $$(\mathsf{time} \leq 3 \rightarrow h\,!\,\mathsf{time}\ []\ \Delta 5) \parallel h\,?\,y,$$

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

  $$(\mathsf{time} \leq 3 \rightarrow h\,!\,\mathsf{time}\ []\ \Delta 0) \parallel h\,?\,y.$$

  The time-out takes place, leading to: $h\,?\,y$. 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. The χ formalism has both urgent and non-urgent actions. 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 [5] 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

  - $\Delta 5 \parallel x := 1$
  - $\Delta 5 \,[]\, x := 1$
  - $\mathsf{time} = 0 \curvearrowright (\mathsf{time} \leq 0 \rightarrow x := 1)$

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

  - $\Delta 5 \parallel [x := 1]$
  - $\Delta 5 \,[]\, [x := 1]$
  - $\mathsf{time} = 0 \curvearrowright (\mathsf{time} \leq 0 \rightarrow [x := 1])$

  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 [8].

  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.

- Syntactic extensions. Ease of modeling is further supported in χ by extension of the small set of orthogonal core process terms with additional process terms for ease of modeling. These additional process terms are defined by means of a straightforward mapping into the core process terms.

2. Concepts for complex system specification.

   - 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.

The history of the χ 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 [13] 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 [19]. Later, the hybrid language and simulator were developed [7, 18]. For the purpose of verification, the discrete-event part of the language was mapped onto the process algebra χ_σ by means of a syntactical translation. The semantics of χ_σ was defined using a structured operational semantics style (SOS), bisimulation relations were derived, and a model checker was built [5]. In this way, verification of discrete-event χ models was made possible [4]. In [17], the hybrid χ language

was formally defined. The timed χ language described in this report is obtained as a simplification of hybrid χ. It is suited for discrete-event modeling. Where in χ_σ, it was not possible to refer to the current (model) time, in timed χ, there is a predefined variable time, that denotes the current time.

This report is organized as follows. Chapter 2 describes the syntax and informal semantics of the timed χ language. In Chapter 3, the semantics of timed χ is formally specified. An example in Chapter 4 illustrates the use of the language. A general translation scheme for translating timed automata to timed χ is given in Chapter 5. The derivation of timed χ from hybrid χ is described in Chapter 6. In Chapter 7, a notion of equivalence is defined, which is shown to be a congruence for all timed χ operators. Furthermore, some useful properties of closed timed χ process terms are given.

Chapter 2

Syntax and informal semantics of the timed Chi language

This chapter presents a concise definition of the syntax and informal semantics of timed χ. The syntax definition is incomplete in the sense that the syntax of predicates, expressions, etc, is not defined. In the remainder of this report, we usually refer to timed χ as χ.

2.1 Processes

A χ process is a triple $\langle p, \sigma, E \rangle$, where p denotes a process term, σ denotes a valuation, and E denotes an environment. The syntax of process terms is introduced in Section 2.2. Variables in χ 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 σ and the environment E, together define the variables that exist in the χ process and the variable classes to which they belong.

Discrete behavior (instantaneous changes) of a χ process is represented by means of action transitions, and delay behavior (time passing) is represented by means of time transitions.

The variables are grouped into different classes with respect to the delay behavior and the 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 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 during action transitions. Such changes need to be explicitly specified. This is the normal behavior of the χ variables. The predefined variable time is by definition non-jumping.
- 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. Note that in principle, jumping variables occur only 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 variable, see Sections 3.3.2 and 3.4.6.

In χ, an environment is a tuple (J, R), where J denotes the set of jumping variables, and R denotes a recursive process definition. It is required that $J \subseteq (\mathrm{dom}(\sigma) \setminus \{\mathsf{time}\})$, and $\mathrm{dom}(\sigma) \cap \mathrm{dom}(R) = \emptyset$. 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 domain of the valuation σ in a χ process $\langle p, \sigma, E\rangle$ consists of the discrete variables and the predefined variable time.

For a χ process $\langle p, \sigma, (J, R)\rangle$, 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 $\mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$.
  - the set of non-jumping discrete variables is $D \setminus J$,
  - the set of jumping discrete variables is $D \cap J$.
- The predefined (non-jumping) variable denoting the current time is time.

A χ process $\langle p, \sigma, E\rangle$ is consistent if valuation σ is consistent with p in environment E. In timed χ, there are two process terms which can introduce inconsistencies: the inconsistent process term $\bot$ that is inconsistent with all valuations, and the signal emission operator $u \curvearrowright p$ that is inconsistent with all valuations in which predicate u does not hold. In χ, only consistent processes can perform action or delay transitions, and the result of an action or delay transition is always a consistent process.

2.2 Process terms

Process terms P (without P_{ext}, see the table below) are the 'core' elements of the χ language. In Section 2.3, the syntax of χ process terms is extended with process terms P_{ext} to ensure better readability of χ models. The semantics of those process terms is defined in terms of the core process terms given in this section.

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

An informal, concise explanation of the core syntax, together with some additional (informal) definitions, is given below. Chapter 3 gives a more detailed account of the meaning. The core 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 χ process term p cannot perform actions nor delays. Only the χ 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 actually refer to the process term together with a valuation and environment.

2.2.1 Action predicates

An instantaneous change of variables in χ 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_{a}, such that predicate r is satisfied. The predefined global variable time cannot be assigned. The action label l_{a} is taken from a given set A_{label} which at least contains the special action label τ representing the internal or silent step. The non-jumping variables that are not mentioned in W remain unchanged, and the jumping variables may obtain arbitrary values.

In this report, we do not explicitly give a syntax for such predicates r. In r, variables and '$-$' superscripted variables 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 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 valuation that results from the execution of the action predicate. A predicate r is satisfied if evaluating the '$-$' superscripted variables in the original valuation and evaluating the normal occurrences of the variables in the obtained valuation means that the predicate is true. Note that it can be the case that different instantaneous changes satisfy the predicate, this may result in non-determinism.

Note that the (multi-)assignment is not a primitive in χ, as for example in [5]. This is because action predicates are more expressive than assignments. An assignment can be expressed as an action predicate (see Section 2.3.2), but not the other way around. Consider for example the action predicate $\{x\} : x \in [0, 1] \gg \tau$ that changes the value of x to a value in the interval $[0, 1]$,

such as used in the example model in Chapter 4. Also, the predicate of an action predicate may consist of a conjunction of implicit equations, e.g. $\{\mathbf{x}\} : f_1(\mathbf{x}^-, \mathbf{x}) = 0 \wedge \ldots \wedge f_n(\mathbf{x}^-, \mathbf{x}) = 0 \gg \tau$. The solution of such a system of equations, if present, need not always be expressible in an explicit form. The system may also have multiple solutions.

Deadlock and inconsistency

In χ, only consistent processes can perform 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 signal emission process term $x \geq 0 \curvearrowright p$ is consistent for the valuations in which the value of x is greater or equal to zero, and inconsistent for all other valuations. Inconsistent process term $\perp$ is inconsistent for all valuations, and it cannot perform any transition. Process term $\perp$ originates from the process algebra with propositional signals ACP_{ps} ([1]). The deadlock process term δ cannot perform actions or delays. It is however consistent with arbitrary valuations. Both process terms are needed for the specification of properties only.

2.2.2 Delay enabling operator

By means of the *delay enabling operator* $[p]$, delay behavior of arbitrary duration can be specified. The resulting behavior is such that arbitrary delays are allowed. As a consequence, any delay behavior of p is neglected. The action behavior of p remains unchanged.

2.2.3 Signal emission

Signal emission operator $u \curvearrowright p$, where u denotes a predicate over variables, behaves as p for those valuations where u holds. The process term is inconsistent with valuations for which u does not hold.

2.2.4 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.

2.2.5 Conditional

The *guarded process term* $b \rightarrow p$ can perform whatever actions p can perform under the condition that the guard b evaluates to true using the current valuation. All variables are allowed to occur in b. The guarded process term can delay according to p under the condition that for the intermediate 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.

2.2.6 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.

2.2.7 Parallelism

Parallelism can be specified by means of the *parallel composition operator* $\parallel$. 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 $p \parallel q$ 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 current 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 χ 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 τ 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 χ 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 (H is the set of all possible channel labels), ensures that p can only delay in case no communication via a channel from $\mathcal{H}$ is possible. Such urgent channels correspond to urgent channels defined in some versions of timed automata, such as UPPAAL [8].

2.2.8 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.

2.2.9 Hierarchical modeling

Thus far, it has been assumed that all variables that are allowed to occur in a χ process term are declared in the valuation. 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 $[\![_{\mathrm{V}}\, \sigma_{\perp} \mid p \,]\!]$ is introduced, where $\sigma_{\perp}$ denotes a valuation of local discrete variables, where values may be undefined ($\perp$). The set of local discrete variables is $\mathrm{dom}(\sigma_{\perp})$. It is allowed that the local variables have been declared on a more global level already. Any occurrence of a variable from $\mathrm{dom}(\sigma_{\perp})$ 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 τ) 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 Syntactic extensions

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

2.3.1 Processes

A χ model is of the following form:

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

where

- $s_1, \ldots, s_k$ denote the discrete variables,
- $h_1, \ldots, h_l$ 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.

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 χ model is an abbreviation for the set of χ processes defined by:

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

namely for each valuation σ_{st}, with $\mathrm{dom}(\sigma_{st}) = \{s_1, \ldots, s_k, \mathsf{time}\}$, a separate χ process. In the χ process, A_{ia} represents the internal send and receive actions via channels $h_1, \ldots, h_l$.

As a shorthand, the keyword disc is omitted when there are no discrete variable declarations, and the keyword chan is omitted when there are no 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

The syntactic extensions for process terms are defined as follows:

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

The operators of P and P_{ext} 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 (τ) 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 τ, 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.

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

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 Δd denotes a process term that first delays for d time units, and then terminates by means of an internal action τ.

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

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 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.

Scope operator

The modeling scope operator process term

$$[\![\ \mathsf{disc}\ \mathbf{s}_k, \mathsf{chan}\ \mathbf{h}_m, i, L_R\ \text{`}|\text{'}\ p\ ]\!]$$

is used to declare a scope consisting of local discrete variables $s_1, \ldots, s_k$, local channels $h_1, \ldots, h_m$, initialization predicate i, and local recursion definition list L_R. The variables all have to be different.

$$
\begin{array}{lcl}
[\![\ \mathsf{disc}\ \mathbf{s}_k & & [\![_{\mathrm{V}}\ \sigma_s \\
,\ \mathsf{chan}\ \mathbf{h}_m & & |\ \ [\![_{\mathrm{H}}\ \{h_1, \ldots, h_m\} \\
,\ i & \triangleq & \quad |\ \upsilon_{\{h_1,\ldots,h_m\}}([\![_{\mathrm{R}}\ \{L_R\} \mid i \curvearrowright p\ ]\!]) \\
,\ L_R & & \quad]\!] \\
|\ p & &]\!] \\
]\!] & &
\end{array}
$$

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

In a similar way as defined for χ processes, the keyword disc is omitted when there are no discrete variable declarations, 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_{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:

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

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

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

must be present among the j process definitions $pd_1 \ldots pd_j$. Here l_{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, $\mathbf{z}_d$ denotes a comma separated list of local discrete variables.

In process term p_{body}, apart from the local variables $\mathbf{z}_d$ 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 $\mathbf{z}_d$ 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_{body} must be declared.

Formally, the syntactic translation of process instantiation

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

with corresponding process definition

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

is given by

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

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_{body}.

The variable w is assumed to be fresh with respect to $\mathbf{x}'_k$, $\mathbf{v}_n$, $\mathbf{z}_d$. 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_{body} normally does not change the values of these variables.

2.4 Data types

The χ 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 χ 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 χ type system is strictly enforced in the χ tools. However, since the type system is not formalized, it is omitted from the specifications in this report.

Chapter 3

Semantics of the timed Chi language

This chapter presents the structured operational semantics (SOS [16]) of timed χ. It associates a hybrid transition system [6] with a χ process. The semantics is defined only for a subset of the syntactically allowed χ processes. E.g. the semantics of the χ 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 χ 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 an SOS is to define the behavior of hybrid χ processes at a certain chosen level of abstraction. The meaning of a χ 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 Λ (containing at least the booleans $\mathbb{B}$ and the reals $\mathbb{R}$). The set of all valuations is denoted Σ: $\Sigma = V \mapsto \Lambda$, and we assume $\sigma \in \Sigma$ and $\mathsf{time} \in \mathrm{dom}(\sigma)$ for all χ processes $\langle p, \sigma, E \rangle$. The set T is used to represent points in time; usually $T = \mathbb{R}_{\geq 0}$. The set of environments ES is defined as $ES = \mathcal{P}(V) \times RS$, 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) $\_ \xrightarrow{\_} \_ \subseteq (P \times \Sigma \times ES) \times (\Sigma \times A \times \Sigma) \times (P \times \Sigma \times ES)$, where A denotes the set of actions, and is defined as $A = A_{\text{label}} \cup A_{\text{com}}$. The set of action labels A_{label} includes at least the pre-defined internal action τ. The set of communication actions A_{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 valuations ξ and ξ' and thereby transforms into the process $\langle p', \sigma', E'\rangle$, where σ' and E' denote the accompanying valuation and environment of the process term p', respectively, after the discrete action a is executed.

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

2. Delay behavior (time-passing) by means of time transitions: $\_ \overset{-}{\longmapsto} \_ \subseteq (P \times \Sigma \times ES) \times (T \times (T \mapsto \Sigma)) \times (P \times \Sigma \times ES)$. The intuition of a time transition $\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p', \sigma', E'\rangle$ is that during the time transition, the 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: $\_ \overset{-}{\rightsquigarrow} \subseteq (P \times \Sigma \times ES) \times \Sigma$. The intuition of a consistency transition $\langle p, \sigma, E\rangle \overset{\xi}{\rightsquigarrow}$ is that process term p is consistent with valuation ξ in environment E.

In this report, for all transitions, the domain of the valuation σ equals the domain of valuation σ', and environment E equals environment E'.

For all action transitions $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \checkmark, \sigma', E'\rangle$ and $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle p', \sigma', E'\rangle$, $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$, $\xi = \sigma$, $\xi' = \sigma'$, and $E = E'$.

For all time transitions $\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle p, \sigma', E'\rangle$, $\mathrm{dom}(\rho) = [0, t]$, $\rho(0) = \sigma$, $\rho(t) = \sigma'$, and $E = E'$. These properties of the semantics can be found in Chapter 7.

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 χ deduction rules, some additional abbreviations are used. Notation $E \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma'\rangle$, where $q \in P \cup \{\checkmark\}$ is an abbreviation for $\langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle q, \sigma', E\rangle$, notation $E \Vdash \langle p, \sigma\rangle \xmapsto{t,\rho} \langle q, \sigma'\rangle$ is an abbreviation for $\langle p, \sigma, E\rangle \xmapsto{t,\rho} \langle q, \sigma', E\rangle$, 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 $\langle p, \sigma\rangle \xmapsto{t,\rho} \langle q, \sigma'\rangle$ 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 ξ, a, and ξ'.

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 [9], 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 $g(c) = f(c) \restriction S$ 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 $c \in \mathrm{dom}(g)$.

3.2.2 Notations

Let $x \in V$ be a variable, $S \subseteq V$ be a set 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 σ. We use the similar notation $\sigma(e)$. to denote the value of expression e for valuation σ.
- Function $\Xi \in \Sigma \times \mathcal{P}(V) \to \mathcal{P}(\Sigma)$ returns a set of valuations, given a valuation, and the set of jumping variables. Formally, function Ξ is defined as:

  $$\Xi(\sigma, J) = \{\sigma' \mid \mathrm{dom}(\sigma') = \mathrm{dom}(\sigma),\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\ \sigma'(x) = \sigma(x)\}.$$

  The domain of the valuations is given by $\mathrm{dom}(\sigma)$. The values of the variables in $\mathrm{dom}(\sigma) \setminus J$ are given by σ. The jumping variables J are allowed to change arbitrarily.
- Function $\Omega \in \Sigma \times T \to \mathcal{P}(T \mapsto \Sigma)$ returns a set of trajectories for the model variables, given a valuation and the duration of the time transition. For $t \geq 0$, the set contains exactly one trajectory. For $t < 0$, the set is empty. Formally, function Ω is defined as:

  $$\begin{array}{ll}
  \Omega(\sigma, t) = & \\
  \{\rho & \\
  \mid\ \rho \in [0, t] \mapsto (\mathrm{dom}(\sigma) \to \Lambda) & \\
  ,\ t \geq 0 & \\
  ,\ \forall x \in \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\} : & \rho \downarrow x \text{ is a constant function.} \\
  ,\ \forall x \in \mathrm{dom}(\sigma) : & (\rho \downarrow x)(0) = \sigma(x) \\
  ,\ \forall s \in [0, t] : & \rho(s)(\mathsf{time}) = \sigma(\mathsf{time}) + s \\
  \} &
  \end{array}$$

  In some SOS rules describing delay behavior, $\Omega(\sigma, t)$ is used as a hypothesis. It does not restrict t and the trajectory ρ other than by means of the default restrictions. Among others, the discrete variables remain constant, and the duration t of the time transition must be positive or zero.

3.3 Deduction rules for atomic process terms

3.3.1 Action predicate

Action predicate process term $W : r \gg l_a$ denotes instantaneous changes to the variables from set $W \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, by means of an action labeled $l_a \in A_{\mathrm{label}}$, such that predicate r over variables from $\mathrm{dom}(\sigma^-)$ and $\mathrm{dom}(\sigma')$ is satisfied, see Rule 1.

Variables occurring with a '$^-$' superscript in r are evaluated in σ^-, which denotes a valuation with $\mathrm{dom}(\sigma^-) = \{x^- \mid x \in \mathrm{dom}(\sigma)\}$, and $\sigma^-(x^-) = \sigma(x)$. For valuation σ', the values of the non-jumping variables $(\mathrm{dom}(\sigma) \setminus (J \cup W))$ are given by σ. The jumping variables J and the variables from set W are allowed to change such that the action predicate is satisfied.

Rule 2 states that action predicates are consistent with any valuations σ in any environment E.

$$\frac{\sigma' \in \Xi(\sigma, J \cup W),\ \sigma^- \cup \sigma' \models r}{(J, R) \Vdash \langle W : r \gg l_a, \sigma\rangle \xrightarrow{\sigma,\ l_a,\ \sigma'} \langle \surd, \sigma'\rangle} \text{T-1}$$

$$\frac{}{E \Vdash \langle W : r \gg l_a, \sigma\rangle \overset{\sigma}{\rightsquigarrow}} \text{T-2}$$

3.3.2 Send and receive

Send and receive process terms $h \mathbin{!!} \mathbf{e}_n$ and $h \mathbin{??} \mathbf{x}_n$ 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 valuation σ, see Rule 3, where $\mathbf{e}_n$ denotes $e_1, \ldots, e_n$, $[\sigma(\mathbf{e}_n)]$ denotes the list of values $[\sigma(e_1), \ldots, \sigma(e_n)]$ for $n \geq 1$, and $\sigma(e)$ denotes the value of expression e for valuation σ. The case that n equals 0, represents the case where nothing is sent via the channel, and $\mathbf{e}_0$ and $[\sigma(\mathbf{e}_0)]$ denote an empty expression and an empty list, respectively. For $n \geq 1$, the receive process term $h \mathbin{??} x_1, \ldots, x_n$ can receive the list of values $[c_1, \ldots, c_n]$, see Rule 4, where $\mathbf{x}_n$ denotes $x_1, \ldots, x_n$, $\{\mathbf{x}_n\}$ denotes the set $\{x_1, \ldots, x_n\}$ ($\{\mathbf{x}_n\} \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$), $[\mathbf{c}_n]$ denotes the list of values $[c_1, \ldots, c_n]$, and $\sigma'(\mathbf{x}_n) = \mathbf{c}_n$ is an abbreviation for $\sigma'(x_1) = c_1,\ \ldots,\ \sigma'(x_n) = c_n$. 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 $\sigma'(\mathbf{x}_0) = \mathbf{c}_0$ always holds. Furthermore, we assume $\{\mathbf{x}_n\} \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$. Rules 5 and 6 state that the send and receive process terms are consistent with any valuation σ in any environment E.

$$\frac{\sigma' \in \Xi(\sigma, J)}{(J, R) \Vdash \langle h \mathbin{!!} \mathbf{e}_n, \sigma\rangle \xrightarrow{\sigma,\ \mathrm{isa}(h, [\sigma(\mathbf{e}_n)]),\ \sigma'} \langle \surd, \sigma'\rangle} \text{T-3}$$

$$\frac{\sigma' \in \Xi(\sigma,\, J \cup \{\mathbf{x}_n\}),\ \sigma'(\mathbf{x}_n) = \mathbf{c}_n}{(J,\, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \xrightarrow{\sigma,\ \mathrm{ira}(h,[\mathbf{c}_n],\{\mathbf{x}_n\}),\ \sigma'} \langle \surd, \sigma' \rangle} \text{ T-4}$$

$$\frac{}{E \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \overset{\sigma}{\rightsquigarrow}} \text{ T-5} \qquad \frac{}{E \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \overset{\sigma}{\rightsquigarrow}} \text{ T-6}$$

3.3.3 Consistent deadlock

Process term δ cannot perform any action transition, nor time transition. It is, however, consistent with any valuation σ in any environment E.

$$\frac{}{E \Vdash \langle \delta, \sigma \rangle \overset{\sigma}{\rightsquigarrow}} \text{ T-7}$$

3.3.4 Inconsistent process term

Inconsistent process term $\bot$ is considered to be in an inconsistent state from its start. Like process term δ, process term $\bot$ cannot perform any action transitions, nor time transitions. Process term $\bot$ originates from the process algebra with propositional signals $\mathrm{ACP}_{\mathrm{ps}}$ ([1]).

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 9). Time transitions of p itself are ignored. The delay enabling operator does not affect the action behavior of p (see Rule 8). Process term $[p]$ is consistent with any valuation σ in any environment E (see Rule 10).

$$E\ \frac{\langle p, \sigma \rangle \xrightarrow{\alpha} \langle \overset{\surd}{p'}, \sigma' \rangle}{\langle [p], \sigma \rangle \xrightarrow{\alpha} \langle \overset{\surd}{p'}, \sigma' \rangle} \text{ T-8} \qquad E\ \frac{\rho \in \Omega(\sigma, t)}{\langle [p], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [p], \rho(t) \rangle} \text{ T-9}$$

$$\frac{}{E \Vdash \langle [p], \sigma \rangle \overset{\sigma}{\rightsquigarrow}} \text{ T-10}$$

3.4.2 Signal emission operator

The signal emission operator $u \curvearrowright p$ ensures that p starts its behavior from a valuation ξ in which predicate u is satisfied. This operator was inspired by the signal emission operator from the process algebra with propositional signals $\mathrm{ACP}_{\mathrm{ps}}$ [1], which was also used in [3]. Rule 13 states that the signal emission operator restricts consistency of process term $u \curvearrowright p$ to those valuations ξ that satisfy predicate u in environment E.

$$E \;\frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix},\sigma'\rangle,\ \xi \models u}{\langle u \curvearrowright p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix},\sigma'\rangle}\ \text{T-11} \qquad E \;\frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle,\ \rho(0) \models u}{\langle\, u \curvearrowright p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}\ \text{T-12}$$

$$E \;\frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow},\ \xi \models u}{\langle\, u \curvearrowright p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ \text{T-13}$$

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 valuation ξ' must be consistent with q (see Rule 14).

$$E \;\frac{\langle p,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle\checkmark,\sigma'\rangle,\ \langle q,\sigma'\rangle \overset{\xi'}{\rightsquigarrow}}{\langle p;\ q,\sigma\rangle \xrightarrow{\xi,a,\xi'} \langle q,\sigma'\rangle}\ \text{T-14} \qquad E \;\frac{\langle p,\sigma\rangle \xrightarrow{\alpha} \langle p',\sigma'\rangle}{\langle p;\ q,\sigma\rangle \xrightarrow{\alpha} \langle p';\ q,\sigma'\rangle}\ \text{T-15}$$

$$E \;\frac{\langle p,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma'\rangle}{\langle p;\ q,\sigma\rangle \overset{t,\rho}{\longmapsto} \langle p';\ q,\sigma'\rangle}\ \text{T-16} \qquad E \;\frac{\langle p,\sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle p;\ q,\sigma\rangle \overset{\xi}{\rightsquigarrow}}\ \text{T-17}$$

3.4.4 Guard operator

The guarded process term $b \to p$ can perform whatever actions p can perform under the condition that the guard evaluates to true using valuation ξ (we assume b to be defined in ξ in Rule 18). Evaluating the guard in ξ ensures that when guard operators are nested with signal emission operators, actions can be executed only if all emitted predicates and all guards hold, independently of the order.

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 perform a time transition of 1, such that the value of the current time time becomes 1, and thereafter it can perform a τ action to the terminated process. If the condition in Rule 20 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 then 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 20, 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 20 requires consistency if the guard holds in the end-point of the trajectory. This ensures that it is impossible to delay to an inconsistent state.

Rule 21 and 22 define that $b \rightarrow p$ is consistent with (21) valuations for which b holds and with which p is consistent, and with (22) valuations for which b does not hold.

$$E\ \frac{\langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \xi \models b}{\langle b \rightarrow p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle}\ \text{T-18}$$

$$E\ \frac{\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle,\ \forall_{s\in[0,t]}\ \rho(s) \models b}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p', \sigma'\rangle}\ \text{T-19}$$

$$E\ \frac{\begin{array}{c}\rho \in \Omega(\sigma, t),\ \forall_{s\in(0,t)}\ \rho(s) \models \neg b,\\ \rho(0) \models b \Rightarrow \langle p, \sigma\rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma'\rangle\\ \rho(t) \models b \Rightarrow \langle p, \rho(t)\rangle \overset{\rho(t)}{\rightsquigarrow}\end{array}}{\langle b \rightarrow p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle b \rightarrow p, \rho(t)\rangle}\ \text{T-20}$$

$$E\ \frac{\langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow},\ \xi \models b}{\langle b \rightarrow p, \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ \text{T-21} \qquad \frac{\sigma \models \neg b}{E \Vdash \langle b \rightarrow p, \sigma\rangle \overset{\sigma}{\rightsquigarrow}}\ \text{T-22}$$

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 valuation ξ is consistent with q, as specified in Rule 23.

The passage of time by itself cannot result in making a choice (see Rule 24). This is called strong time-determinism, as defined in [14]. Consider for example the χ 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 χ_σ process algebra [5]. There, $\langle h\,!\,[]\,(\Delta 10; \Delta 1), \sigma \rangle$ does not make a choice after expiration of $\Delta 10$.

Rule 25 states that an alternative composition is consistent with a valuation if both alternatives are consistent with that valuation.

$$E \quad \frac{\langle p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle,\ \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle p\ []\ q, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle,\ \langle q\ []\ p, \sigma \rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle} \text{ T-23}$$

$$E \quad \frac{\langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle,\ \langle q, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma' \rangle}{\langle p\ []\ q, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p'\ []\ q', \sigma' \rangle} \text{ T-24}$$

$$E \quad \frac{\langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow},\ \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle p\ []\ q, \sigma \rangle \overset{\xi}{\rightsquigarrow}} \text{ T-25}$$

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 27). Process term p can only perform action transitions from a valuation ξ which is consistent with q. Furthermore, q must be consistent with the resulting valuation ξ' (see Rule 27).

The parallel composition allows the synchronization of matching send and receive actions. A send action isa(h, cs) and a receive action 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 valuations ξ' and σ' 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 26). The result of the synchronization is a communication action that is represented by ca(h, cs) as defined by Rule 26.

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 ρ) of their parallel composition as defined by Rule 28.

A parallel composition of two process terms is consistent with a valuation if both process terms are consistent with that valuation (see Rule 29).

$$\frac{\begin{array}{c}(J \cup W, R) \Vdash \langle p, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \left\langle \begin{array}{c} \checkmark \\ p' \\ \checkmark \\ p' \end{array}, \sigma' \right\rangle, \\ (J, R) \Vdash \langle q, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \left\langle \begin{array}{c} \checkmark \\ \checkmark \\ q' \\ q' \end{array}, \sigma' \right\rangle \end{array}}{\begin{array}{c}(J, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \left\langle \begin{array}{c} \checkmark \\ p' \\ q' \\ p' \parallel q' \end{array}, \sigma' \right\rangle, \\ \langle q \parallel p, \sigma \rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \left\langle \begin{array}{c} \checkmark \\ p' \\ q' \\ q' \parallel p' \end{array}, \sigma' \right\rangle \end{array}} \text{T-26}$$

$$E \; \frac{\langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}, \langle p, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{array}{c} \checkmark \\ p' \end{array}, \sigma' \rangle, \langle q, \sigma' \rangle \overset{\xi'}{\rightsquigarrow}}{\langle p \parallel q, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{array}{c} q \\ p' \parallel q \end{array}, \sigma' \rangle, \langle q \parallel p, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{array}{c} q \\ q \parallel p' \end{array}, \sigma' \rangle} \text{T-27}$$

$$E \; \frac{\langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle, \langle q, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p' \parallel q', \sigma' \rangle} \text{T-28}$$

$$E \; \frac{\langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}, \langle q, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle p \parallel q, \sigma \rangle \overset{\xi}{\rightsquigarrow}} \text{T-29}$$

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 30). Action encapsulation has no effect on time transitions and consistency, as defined by Rules 31 and 32.

$$E \quad \frac{\langle p, \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle, \ a \notin \mathcal{A}}{\langle \partial_{\mathcal{A}}(p), \sigma \rangle \xrightarrow{\xi, a, \xi'} \langle \begin{smallmatrix}\checkmark\\ \partial_{\mathcal{A}}(p')\end{smallmatrix}, \sigma' \rangle} \text{ T-30}$$

$$E \quad \frac{\langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle}{\langle \partial_{\mathcal{A}}(p), \sigma \rangle \overset{t,\rho}{\longmapsto} \langle \partial_{\mathcal{A}}(p'), \sigma' \rangle} \text{ T-31} \qquad E \quad \frac{\langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle \partial_{\mathcal{A}}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}} \text{ T-32}$$

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 33 and 35. Time transitions are allowed only if at each intermediate state while delaying no communication actions via channels from $\mathcal{H}$ are possible.

$$E \quad \frac{\langle p, \sigma \rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma' \rangle}{\langle \upsilon_{\mathcal{H}}(p), \sigma \rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\checkmark\\ \upsilon_{\mathcal{H}}(p')\end{smallmatrix}, \sigma' \rangle} \text{ T-33}$$

$$E \quad \frac{\begin{array}{c}\langle p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle, \\ \forall_{s \in [0,t)} \left(\langle p, \sigma \rangle \overset{s,\rho\restriction[0,s]}{\longmapsto} \langle p_s, \sigma_s \rangle, \ \langle p_s, \sigma_s \rangle \overset{t-s,\rho_{-s}}{\longmapsto} \langle p', \sigma' \rangle, \ \forall_{h \in \mathcal{H}} \langle p_s, \sigma_s, E \rangle \overset{\mathrm{ca}(h,*)}{\nrightarrow} \right)\end{array}}{\langle \upsilon_{\mathcal{H}}(p), \sigma \rangle \overset{t,\rho}{\longmapsto} \langle \upsilon_{\mathcal{H}}(p'), \sigma' \rangle} \text{ T-34}$$

$$E \quad \frac{\langle p, \sigma \rangle \overset{\xi}{\rightsquigarrow}}{\langle \upsilon_{\mathcal{H}}(p), \sigma \rangle \overset{\xi}{\rightsquigarrow}} \text{ T-35}$$

where ρ_{-s} denotes the trajectory ρ 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 ρ: $\mathrm{dom}(\rho) = [0, t]$.

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 χ process directly, or by means of the recursion scope operator, see Section 3.4.12.

$$(J, R)\ \frac{\langle R(X), \sigma\rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma'\rangle}{\langle X, \sigma\rangle \xrightarrow{\alpha} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma'\rangle}\ \text{T-36} \qquad (J, R)\ \frac{\langle R(X), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{\langle X, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}\ \text{T-37}$$

$$(J, R)\ \frac{\langle R(X), \sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle X, \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ \text{T-38}$$

3.4.10 Variable scope operator

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

$$[\![\mathrm{V}\, \sigma_{\mathrm{d}_\perp} \mid p]\!],$$

that is used in an environment (J, R), with valuation σ, and where $\sigma_{\mathrm{d}_\perp}$ denotes a local valuation with domain $\{\mathbf{d}\}$, and $\mathbf{d}$ denotes the local discrete variables $d_1, \ldots, d_k$, behaves as p after 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 $\mathbf{d}'$, in the rules below, denotes fresh variables $d'_1, \ldots, d'_k$ with respect to $\mathrm{dom}(\sigma)$. The local variables $d_1, \ldots, d_k$ are assumed to be all different. Notation $p[\mathbf{d}'/\mathbf{d}]$ denotes the process term that is obtained by substitution of the variables $\mathbf{d}$ in p by $\mathbf{d}'$. 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 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 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 $\kappa \in \Sigma \times \Sigma \times A \times \Sigma \to \Sigma \times A \times \Sigma$ is defined as follows. For arbitrary receive actions $\mathrm{ira}(h, cs, W)$:

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

and for all other actions:

$$\kappa_\sigma(\xi, a, \xi') = \xi_\sigma, a, \xi'_\sigma,$$

where valuations ξ_σ, ξ'_σ denote $\xi \restriction \mathrm{dom}(\sigma)$, $\xi' \restriction \mathrm{dom}(\sigma)$, respectively. Furthermore, in the rules below, the following abbreviations are used: valuation σ'_σ denotes $\sigma' \restriction \mathrm{dom}(\sigma)$, and trajectory ρ_σ denotes $\rho \downarrow \sigma$.

Valuation $\sigma_{\mathrm{d}_\perp} \in \{\mathbf{d}\} \mapsto (\Lambda \cup \{\perp\})$ and valuation $\sigma_{\mathrm{d}'} \in \{\mathbf{d}'\} \mapsto \Lambda$ define the same values for all (renamed) variables for which $\sigma_{\mathrm{d}_\perp}$ is defined. For the undefined variables in $\sigma_{\mathrm{d}_\perp}$, $\sigma_{\mathrm{d}'}$ has an arbitrary value: $\forall\, v \in \mathrm{dom}(\sigma_{\mathrm{d}_\perp}) : \sigma_{\mathrm{d}_\perp}(v) \neq \perp \Rightarrow \sigma_{\mathrm{d}'}(v[\mathbf{d}'/\mathbf{d}]) = \sigma_{\mathrm{d}_\perp}(v)$, where $v[\mathbf{d}'/\mathbf{d}]$ denotes the renamed version of variable v.

$$E \quad \frac{\langle p[\mathbf{d}'/\mathbf{d}], \sigma \cup \sigma_{\mathrm{d}'}\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle}{\langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{d}_\perp} \mid p\,]\!], \sigma\rangle \xrightarrow{\kappa_\sigma(\xi,a,\xi')} \langle \overset{\checkmark}{[\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}'\})[\mathbf{d}/\mathbf{d}'] \mid p'[\mathbf{d}/\mathbf{d}']\,]\!]}, \sigma'_\sigma\rangle} \text{ T-39}$$

$$E \quad \frac{\langle p[\mathbf{d}'/\mathbf{d}], \sigma \cup \sigma_{\mathrm{d}'}\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{\langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{d}_\perp} \mid p\,]\!], \sigma\rangle \overset{t,\rho_\sigma}{\longmapsto} \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}'\})[\mathbf{d}/\mathbf{d}'] \mid p'[\mathbf{d}/\mathbf{d}']\,]\!], \sigma'_\sigma\rangle} \text{ T-40}$$

$$E \quad \frac{\langle p[\mathbf{d}'/\mathbf{d}], \sigma \cup \sigma_{\mathrm{d}'}\rangle \overset{\xi}{\rightsquigarrow}}{\langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{d}_\perp} \mid p\,]\!], \sigma\rangle \overset{\xi_\sigma}{\rightsquigarrow}} \text{ T-41}$$

3.4.11 Channel scope operator

By means of the channel scope operator, local channels can be introduced in a χ 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 τ actions (see Rule 42). 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 42 only specifies behavior for communication actions $\mathrm{ca}(h, cs)$. Therefore, these 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 $l_{\mathrm{a}} \in A_{\mathrm{label}}$.

$$E \quad \frac{\langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ h \in H_0}{\langle [\![_{\mathrm{H}}\ H_0 \mid p\ ]\!], \sigma\rangle \xrightarrow{\xi,\tau,\xi'} \langle \overset{\checkmark}{[\![_{\mathrm{H}}\ H_0 \mid p'\ ]\!]}, \sigma'\rangle} \text{ T-42}$$

$$E \quad \frac{\langle p, \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma'\rangle,\ \mathrm{ch}(a) \notin H_0}{\langle [\![_{\mathrm{H}}\ H_0 \mid p\ ]\!], \sigma\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{[\![_{\mathrm{H}}\ H_0 \mid p'\ ]\!]}, \sigma'\rangle} \text{ T-43}$$

$$E \quad \frac{\langle p, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{\langle [\![_{\mathrm{H}}\ H_0 \mid p\ ]\!], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{H}}\ H_0 \mid p'\ ]\!], \sigma'\rangle} \text{ T-44}$$

$$E \quad \frac{\langle p, \sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle [\![_{\mathrm{H}}\ H_0 \mid p\ ]\!], \sigma\rangle \overset{\xi}{\rightsquigarrow}} \text{ T-45}$$

3.4.12 Recursion scope operator

By means of the recursion scope operator, local recursion definitions are introduced in a χ process. The application of the recursion scope operator to a process term p with a 'global' valuation σ and a 'global' environment (J, 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 $\mathbf{X}'$.

$$\frac{(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma\rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma'\rangle}{(J, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\ ]\!], \sigma\rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{[\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}']\ ]\!]}, \sigma'\rangle} \text{ T-46}$$

$$\frac{(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle,}{(J, R) \Vdash \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\ ]\!], \sigma\rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{R}}\ \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}']\ ]\!], \sigma'\rangle} \text{ T-47}$$

$$\frac{(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \overset{\xi}{\rightsquigarrow}}{(J, R) \Vdash \langle [\![_{\mathrm{R}} \{\mathbf{X} \mapsto \mathbf{q}\} \mid p \,]\!], \sigma \rangle \overset{\xi}{\rightsquigarrow}} \text{T-48}$$

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 $[\![_{\mathrm{R}}\ \{X \mapsto b \to \mathrm{skip};\ p;\ X\ [] \neg b \to \mathrm{skip}\} \mid X\ ]\!]$, respectively, cannot override existing recursion definitions using the same recursion variable X.

Chapter 4

Discrete-event model of a manufacturing line

A manufacturing line consists of a generator G, distributer D, two manufacturing cells C, and an assembling machine M_{A}. Figure 4.1 shows the iconic model of the manufacturing line. Processes R and E are added to obtain a closed system; they do not model actual behavior.

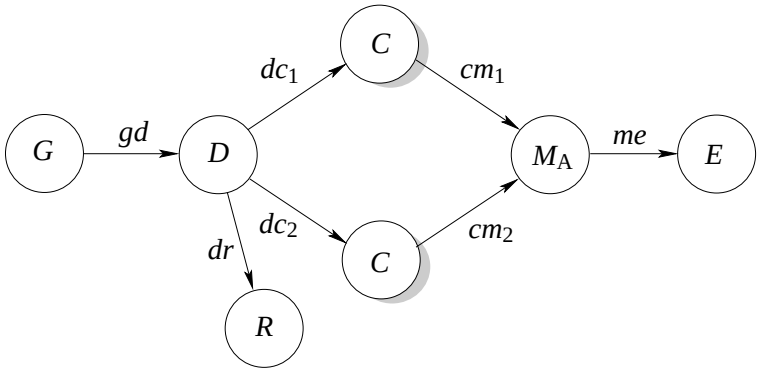

Figure 4.1: Iconic model of a manufacturing line.

The manufacturing line is modeled as follows, where t_{gen}, t_{out}, pt_{min1}, pt_{max1}, pt_{min2}, pt_{max2}, pt_{min3}, pt_{max3}, pt_{min4}, pt_{max4}, and pt denote constants.

$$
\begin{aligned}
&\langle\ G(gd, t_{\mathrm{gen}}) \\
&\|\ D(gd, dc_1, dc_2, dr, t_{\mathrm{out}}) \\
&\|\ R(dr) \\
&\|\ C(dc_1, cm_1, pt_{\mathrm{min1}}, pt_{\mathrm{max1}}, N_1, pt_{\mathrm{min2}}, pt_{\mathrm{max2}}) \\
&\|\ C(dc_2, cm_2, pt_{\mathrm{min3}}, pt_{\mathrm{max3}}, N_2, pt_{\mathrm{min4}}, pt_{\mathrm{max4}}) \\
&\|\ M_{\mathrm{A}}(cm_1, cm_2, me, pt)
\end{aligned}
$$

$$
\begin{array}{l}
\| \; E(me) \\
\rangle
\end{array}
$$

$$G(\text{chan } gd, \text{val } t_{\text{gen}}) = [\![\text{ disc } x,\ x = \text{false} \mid *(\Delta t_{\text{gen}};\ gd\,!\,x) \ ]\!]$$

$$
\begin{array}{l}
D(\text{chan } gd, dc_1, dc_2, dr, \text{val } t_{\text{out}}) = \\
[\![\text{ disc } x \\
\mid *(gd\,?\,x;\ (dc_1\,!\,x \ [] \ dc_2\,!\,x \ [] \ \Delta t_{\text{out}};\ dr\,!\,x)) \\
]\!]
\end{array}
$$

$$R(\text{chan } dr) = [\![\text{ disc } x \mid *(dr\,?\,x) \ ]\!]$$

$$
\begin{array}{l}
M_{\text{A}}(\text{chan } cm_1, cm_2, me, \text{ val } pt) = \\
[\![\text{ disc } x, y \\
\mid *((cm_1\,?\,x \ \| \ cm_2\,?\,y);\ \Delta pt;\ me\,!\,x) \\
]\!]
\end{array}
$$

$$E(\text{chan } me) = [\![\text{ disc } x \mid *(me\,?\,x) \ ]\!]$$

Every t_{gen} time units (assuming $t_{\text{gen}} \geq t_{\text{out}}$, see process D), a product is generated by generator G. A product is modeled by a boolean variable x that is initially false. The boolean indicates whether the product has done the second round in a manufacturing cell C. A product enters the manufacturing line via channel gd. The distributor tries to send a product either via channel dc_1 or channel dc_2 . In case this is not possible within t_{out} time units, the product is rejected and sent to reject process R via channel dr ($dc_1\,!\,x \ [] \ dc_2\,!\,x \ [] \ \Delta t_{\text{out}};\ dr\,!\,x$). Process R consumes the rejected products ($*(dr\,?\,x)$).

A manufacturing cell C, shown in Figure 4.2, consists of two machines (M_{rw}, M) and a N-place FIFO (first-in-first-out) buffer B.

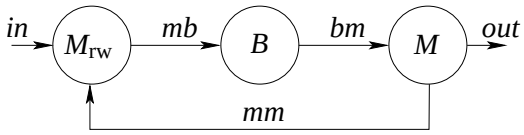

Figure 4.2: Iconic model of a manufacturing cell C.

The χ specification of the manufacturing cell is as follows:

$$
\begin{array}{l}
C(\text{chan } in, out, \text{val } pt_{\text{min1}}, pt_{\text{max1}}, N, pt_{\text{min2}}, pt_{\text{max2}}) \\
[\![\text{ chan } mb, bm, mm
\end{array}
$$

$$
\begin{array}{l}
\mid M_{\mathrm{rw}}(in, mb, mm, pt_{\mathrm{min1}}, pt_{\mathrm{max1}}) \\
\parallel B(mb, bm, N) \\
\parallel M(bm, out, mm, pt_{\mathrm{min2}}, pt_{\mathrm{max2}}) \\
]\!]
\end{array}
$$

Products enter the cell via channel *in*. The routing of a product in the manufacturing cell is as follows: M_{rw}, B, M, M_{rw}, B, M. Products leave the manufacturing cell via channel *out*. The process definitions of the buffer and the machines are given by

$$
\begin{array}{l}
B(\mathsf{chan}\ in, out, \mathsf{val}\ N) = \\
[\![\ \mathsf{disc}\ x \\
,\ xs = [\,] \\
\mid *(\ \mathrm{len}(xs) < N \rightarrow in\,?\,x;\ xs := xs +\!\!+ [x] \\
\quad [\,]\,\mathrm{len}(xs) > 0\ \rightarrow out\,!\,\mathrm{hd}(xs);\ xs :=\ \mathrm{tl}(xs) \\
\quad) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
M_{\mathrm{rw}}(\mathsf{chan}\ in, out, mm, \mathsf{val}\ pt_{\mathrm{min}}, pt_{\mathrm{max}}) = \\
[\![\ \mathsf{disc}\ x, pt \\
\mid *((in\,?\,x\ [\,]\ mm\,?\,x;\ x := \mathrm{true});\ pt : pt \in [pt_{\mathrm{min}}, pt_{\mathrm{max}}];\ \Delta pt;\ out\,!\,x) \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
M(\mathsf{chan}\ in, out, mm, \mathsf{val}\ pt_{\mathrm{min}}, pt_{\mathrm{max}}) = \\
[\![\ \mathsf{disc}\ x, pt \\
\mid *(in\,?\,x;\ pt : pt \in [pt_{\mathrm{min}}, pt_{\mathrm{max}}];\ \Delta pt;\ (x \rightarrow out\,!\,x\ [\,]\,\neg x \rightarrow mm\,!\,x)) \\
]\!]
\end{array}
$$

The buffer can store up to N products, which are stored in a list xs ($\mathrm{len}(xs) < N \rightarrow in\,?\,x;\ xs := xs +\!\!+ [x]$), where $[x]$ denotes a list with one element x, and $+\!\!+$ denotes list concatenation. The empty list is denoted by [ ]. If the buffer is not empty, the first product in the buffer can be sent to the machine via channel *out* ($\mathrm{len}(xs) > 0 \rightarrow out\,!\,\mathrm{hd}(xs);\ xs := \mathrm{tl}(xs)$), where $\mathrm{hd}(xs)$ denotes the first element (head) of list xs, and $\mathrm{tl}(xs)$ denotes the remainder (tail) of list xs without its first element. Machine M_{rw} receives products from channels *in* and *mm*. A product received from *mm* is assigned the value true, which indicates that this product is processed by machine M_{rw} for the second time ($in\,?\,x\ [\,]\ mm\,?\,x;\ x := \mathrm{true}$). The machine has a processing time between pt_{min} and pt_{max} time units ($pt : pt \in [pt_{\mathrm{min}}, pt_{\mathrm{max}}];\ \Delta pt$). Processed products are sent via channel *out* to the buffer B. Machine M receives products via channel *in*, and processes them for *pt* time units. Depending on the value of product variable x, the product is sent either via channel *mm* to machine M_{rw} (x equals false) or it leaves the manufacturing cell via channel *out* (x equals true) ($x \rightarrow out\,!\,x\ [\,]\,\neg x \rightarrow mm\,!\,x$).

After processing in one of the two manufacturing cells C, products are sent to machine M_{A}. Machine M_{A} waits to receive one product via channel cm_1, and one product via channel cm_2,

in a non-deterministic order ($cm_1 \,?\, x \parallel cm_2 \,?\, y$). After processing these two products (Δpt), the combination of them leaves the manufacturing line via channel *out* ($out \,!\, x$). Process E consumes the processed products.

Chapter 5

Translating timed automata to timed Chi

In this chapter, the timed automata model is translated to the timed χ formalism.

5.1 Definition of a timed automaton

A timed automaton (based on [11]) consists of the following components:

- A finite set of (real-valued) clocks $X = \{x_1, \ldots, x_n\}$, and a finite set $Y = \{y_1, \ldots, y_m\}$ of variables.
- A finite directed multi-graph (V, E), where V denotes a set of vertices (locations / control modes) and E denotes a set of edges (control switches). Vertex $v_0 \in V$ denotes the initial location.
- A vertex labeling function inv that assigns an invariant to each location $v \in V$, and a vertex labeling function init that assigns an initialization predicate to the initial location v_0. Invariants and initialization predicates are predicates over variables and clocks.
- Three edge labeling functions guard, reset, and assign that assign to each edge $e \in E$, a guard, clock resets, and assignments to variables, respectively. Guards are predicates over variables and clocks, and clock resets are of the form $x := 0$, where x denotes a clock.
- A finite set Σ of events, and an edge labeling function event $\in E \rightarrow \Sigma$ that assigns an event to each edge $e \in E$.

5.2 General translation scheme

Consider a timed automaton with n clock variables ($X = \{x_1, \ldots, x_n\}$), m discrete variables ($Y = \{y_1, \ldots, y_m\}$), k locations ($V = \{v_1, \ldots, v_k\}$), and initial location v_1, to be translated to a corresponding χ specification. The translation is defined as follows:

$$
\begin{array}{l}
\langle\ [\![_{\mathrm{R}}\ \{\ v_1 \mapsto \neg(\mathcal{T}_{\mathrm{C}}(\mathrm{inv}(v_1))) \rightarrow \bot \\
\qquad\qquad ([]_{e:e\in\mathrm{edges}(v_1)} \\
\qquad\qquad\ [\mathcal{T}_{\mathrm{C}}(\mathrm{guard}(e)) \rightarrow \mathcal{T}_{\mathrm{A}}(\mathcal{T}_{\mathrm{R}}(\mathrm{reset}(e)), \mathcal{T}_{\mathrm{C}}(\mathrm{assign}(e))) \gg \mathrm{event}(e)];\ \mathrm{target}(e) \\
\qquad\qquad) \\
\qquad \vdots \\
\qquad ,\ v_k \mapsto \neg(\mathcal{T}_{\mathrm{C}}(\mathrm{inv}(v_k))) \rightarrow \bot \\
\qquad\qquad ([]_{e:e\in\mathrm{edges}(v_k)} \\
\qquad\qquad\ [\mathcal{T}_{\mathrm{C}}(\mathrm{guard}(e)) \rightarrow \mathcal{T}_{\mathrm{A}}(\mathcal{T}_{\mathrm{R}}(\mathrm{reset}(e)), \mathcal{T}_{\mathrm{C}}(\mathrm{assign}(e))) \gg \mathrm{event}(e)];\ \mathrm{target}(e) \\
\qquad\qquad) \\
\qquad\ \}\ \mid\ v_1\ ]\!]\ , \mathcal{R}(\mathrm{init}(v_1)) \cup \{\mathsf{time} \mapsto 0\}, (\emptyset, \emptyset)\ \rangle
\end{array}
$$

The delay behavior in a vertex v_i is restricted by means of its invariant. This restriction is translated to the process term $\bot$, guarded with the negation of its invariant ($\neg(\mathcal{T}_{\mathrm{C}}(\mathrm{inv}(v_i))) \rightarrow \bot$). Function $\mathcal{T}_{\mathrm{C}}$ replaces all clock variables x by expressions $\mathsf{time} - x$. E.g. invariant $\mathcal{T}_{\mathrm{C}}(x \leq 2)$ becomes $\mathsf{time} - x \leq 2$.

Each outgoing edge is translated to a process term of the form $[b \rightarrow W : r \gg l_{\mathrm{a}};\ X]$. In this process term, the guard b is defined as $\mathcal{T}_{\mathrm{C}}(\mathrm{guard}(e))$.

The label of the action predicate is defined as $\mathrm{event}(e)$, and the predicate r is defined as $\mathcal{T}_{\mathrm{A}}(\mathcal{T}_{\mathrm{R}}(\mathrm{reset}(e)), \mathcal{T}_{\mathrm{C}}(\mathrm{assign}(e)))$, where function $\mathcal{T}_{\mathrm{R}}$ replaces a clock reset $x := 0$ by $x := \mathsf{time}$, and function $\mathcal{T}_{\mathrm{A}}$ combines clock resets and assignments into one action predicate. For example $\mathcal{T}_{\mathrm{A}}(x := \mathsf{time}, y := 2)$ becomes $\{x, y\} : x = \mathsf{time} \wedge y = 2$.

Function $\mathcal{R}$ translates an init predicate $x_0 = c_0 \wedge \cdots \wedge x_n = c_n$ to a valuation $\{x_0 \mapsto c_0, \ldots, x_n \mapsto c_n\}$. E.g. $\mathcal{R}(x = 1 \wedge y = 2) = \{x \mapsto 1, y \mapsto 2\}$.

In the translation scheme, the δ process term cannot be used instead of the inconsistent process term $\bot$. Consider for instance a timed automaton with one clock variable x, one location v_0, function $\mathrm{init}(v_0) = (x = 0)$ and function $\mathrm{inv}(v_0) = (x < 1)$. This automaton can perform time transitions with duration $t \in [0, 1)$. Using the translation scheme, the following χ process is obtained: $\langle [\![_{\mathrm{R}}\ v_0 \mapsto \neg(\mathsf{time} - x < 1) \rightarrow \bot \mid v_0\ ]\!], \{x \mapsto 0, \mathsf{time} \mapsto 0\}, (\emptyset, \emptyset)\rangle$. Like the timed automaton model, this process can perform time transitions of duration $t \in [0, 1)$. Translating the timed automaton model to χ, using δ instead of $\bot$, the following χ process is obtained: $\langle [\![_{\mathrm{R}}\ v_0 \mapsto \neg(\mathsf{time} - x < 1) \rightarrow \delta \mid v_0\ ]\!], \{x \mapsto 0, \mathsf{time} \mapsto 0\}, (\emptyset, \emptyset)\rangle$. This process can perform time transitions of duration $t \in [0, 1]$, including $t = 1$, and is therefore incorrect.

5.3 Example: a coffee vendor machine

5.3.1 Timed automaton model of the coffee vendor machine

After receiving a coin, modeled by means of action *coin* in the timed automaton model of Figure 5.1, the coffee machine allows the user to push a button (action *button*) within 5 time units. If the user pushes the button within this time interval, the machine produces the coffee after 2 to 4 time units (*coffee*). Otherwise, the machine refunds after 0.1 to 1 time units (*refund*).

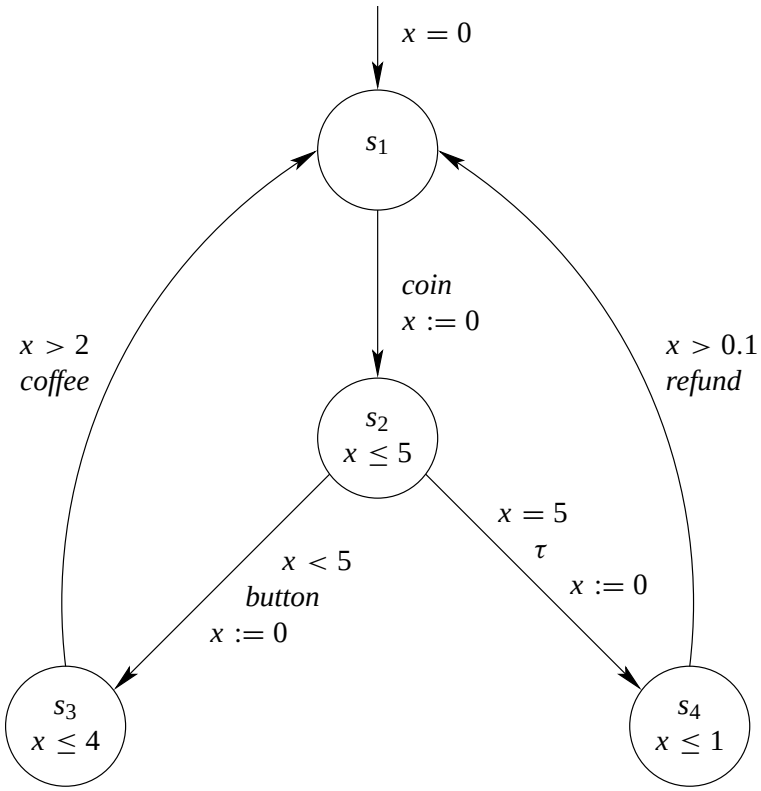

Figure 5.1: Timed automaton of the coffee vendor machine.

5.3.2 A χ model of the coffee vendor machine

Translation of the timed automaton model of the coffee vendor machine described in the previous section results in the following timed χ specification.

$$
\begin{array}{l}
\langle\ [\![_{\mathrm{R}}\ \{\ s_1 \mapsto \neg(\mathrm{true}) \rightarrow \bot \\
\qquad\qquad []\,[\mathrm{true} \rightarrow \{x\} : x = \mathsf{time} \gg \mathit{coin}];\ s_2 \\
\qquad ,\ s_2 \mapsto \neg(\mathsf{time} - x \leq 5) \rightarrow \bot \\
\qquad\qquad []\,[\mathsf{time} - x < 5 \rightarrow \{x\} : x = \mathsf{time} \gg \mathit{button}];\ s_3 \\
\qquad\qquad []\,[\mathsf{time} - x = 5 \rightarrow \{x\} : x = \mathsf{time} \gg \tau];\ s_4 \\
\qquad ,\ s_3 \mapsto \neg(\mathsf{time} - x \leq 4) \rightarrow \bot \\
\qquad\qquad []\,[\mathsf{time} - x > 2 \rightarrow \emptyset : \mathrm{true} \gg \mathit{coffee}];\ s_1 \\
\qquad ,\ s_4 \mapsto \neg(\mathsf{time} - x \leq 1) \rightarrow \bot \\
\qquad\qquad []\,[\mathsf{time} - x > 0.1 \rightarrow \emptyset : \mathrm{true} \gg \mathit{refund}];\ s_1 \\
\qquad \}\\
\quad |\quad s_1 \\
\quad]\!]\\
,\ \{\mathsf{time} \mapsto 0, x \mapsto 0, (\emptyset, \emptyset)\\
\rangle
\end{array}
$$

Chapter 6

Derivation of timed Chi from hybrid Chi

In this chapter, the timed χ language is derived from the hybrid χ language defined in [17], such that hybrid χ is an operational conservative extension of timed χ. This derivation is divided into two main steps: 1) the syntax of hybrid χ is restricted (Section 6.1), 2) the continuous and algebraic variables are removed (Section 6.2).

6.1 The $\mathcal{L}_1$ language

Let $\mathcal{L}_0(C, L)$ be the hybrid χ language, where C and L denote the set of continuous variables and the set of algebraic variables, respectively.

Restricting the syntax of the $\mathcal{L}_0$ language leads to the $\mathcal{L}_1$ language. The syntactic restrictions are listed below:

- remove delay predicate u,
- remove jump enabling operator ι_{J^+},
- remove variable scope operators $[\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \{\mathbf{x}\}, \{\mathbf{g}\} \mid p\,]\!]$, where $\{\mathbf{x}\} \neq \emptyset$ or $\{\mathbf{g}\} \neq \emptyset$. Therefore, the remaining variable scope operators in the $\mathcal{L}_1$ language are of the form $[\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_\perp}, \emptyset, \emptyset \mid p\,]\!]$.

This results in the following syntax for the process terms $p \in P_{\mathcal{L}_1}$ of the $\mathcal{L}_1$ language:

$$
\begin{aligned}
P_{\mathcal{L}_1} ::= \;& W : r \gg l_{\mathrm{a}} \;\mid\; \delta \;\mid\; \perp \\
\mid\; & [P] \;\mid\; u \curvearrowright P \;\mid\; P;\, P \;\mid\; b \rightarrow P \;\mid\; P \,[]\, P \\
\mid\; & P \parallel P \;\mid\; h \,!!\, \mathbf{e}_n \;\mid\; h \,??\, \mathbf{x}_n \;\mid\; \partial_A(P) \;\mid\; \upsilon_{\mathcal{H}}(P) \\
\mid\; & X \;\mid\; [\![_{\mathrm{V}}\, \sigma_\perp, \emptyset, \emptyset \text{ `}|\text{'} P\, ]\!] \;\mid\; [\![_{\mathrm{H}}\, H_0 \text{ `}|\text{'} P\,]\!] \;\mid\; [\![_{\mathrm{R}}\, R \text{ `}|\text{'} P\,]\!] \\
\mid\; & P_{\mathrm{ext}_{\mathcal{L}_1}}
\end{aligned}
$$

$$
\begin{array}{rcl}
P_{\mathrm{ext}_{\mathcal{L}_1}} & ::= & \mathrm{skip} \mid \mathbf{x}_n := \mathbf{e}_n \mid \mathbf{x}_n : r \mid h\,!\,\mathbf{e}_n \mid h\,?\,\mathbf{x}_n \\
& \mid & \Delta_d(P) \mid \Delta d \mid *P \mid *b : P \\
& \mid & [\![\ \mathsf{disc}\ \mathbf{s}_k, \mathsf{chan}\ \mathbf{h}_m, i, L_R\ \text{`}|\text{'}\ P\ ]\!] \\
& \mid & l_{\mathrm{p}}(\mathbf{x}_k,\ \mathbf{h}_m,\ \mathbf{e}_n)
\end{array}
$$

The deduction system of $\mathcal{L}_1$, denoted by $\mathcal{D}_{\mathcal{L}_1}$, consists of all deduction rules of the deduction system of $\mathcal{L}_0$ in which only syntax from $\mathcal{L}_1$ is used, i.e., the deduction rules defining atomic process terms and operators that are removed are omitted.

Lemma 1 $\mathcal{D}_{\mathcal{L}_1} \subset \mathcal{D}_{\mathcal{L}_0}$.

Proof. Trivial ⊠

Lemma 2 *If all syntax used in the conclusion of a deduction rule from $\mathcal{D}_{\mathcal{L}_0}$ is contained in $\mathcal{L}_1$, then all syntax used in the hypothesis of this rule is also contained in $\mathcal{L}_1$.*

Proof. Trivial ⊠

Lemma 3 *Let p and p' be closed process terms from $P_{\mathcal{L}_1}$, σ, σ' be valuations, ξ, ξ' be extended valuations, (C, J, L, R) and (C', J', L', R') be environments, a be an action, ρ be a trajectory, and $t \in T$. Then*

$$
\begin{array}{ll}
\mathcal{L}_0(C,L) \models \langle p, \sigma, (C, J, L, R)\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', (C', J', L', R')\rangle & \Leftrightarrow \\
\mathcal{L}_1(C,L) \models \langle p, \sigma, (C, J, L, R)\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', (C', J', L', R')\rangle, & \\
\mathcal{L}_0(C,L) \models \langle p, \sigma, (C, J, L, R)\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', (C', J', L', R')\rangle & \Leftrightarrow \\
\mathcal{L}_1(C,L) \models \langle p, \sigma, (C, J, L, R)\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma', (C', J', L', R')\rangle, & \\
\mathcal{L}_0(C,L) \models \langle p, \sigma, (C, J, L, R)\rangle \overset{\xi}{\rightsquigarrow} & \Leftrightarrow \\
\mathcal{L}_1(C,L) \models \langle p, \sigma, (C, J, L, R)\rangle \overset{\xi}{\rightsquigarrow}, &
\end{array}
$$

where $\langle p, \sigma, (C, J, L, R)\rangle \xrightarrow{\xi,a,\xi'} \langle \_, \sigma', (C', J', L', R')\rangle$ *is an abbreviation for* $\langle p, \sigma, (C, J, L, R)\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'}, \sigma', (C', J', L', R')\rangle$ *for some* p'.

Proof. The proof for the three statements of the lemma are very similar. Therefore, we only consider the second statement. Proof of left implication ($\Leftarrow$): We prove this lemma by induction on the depth of the proof of a time transition derived in $\mathcal{L}_1(C, L)$ and use case distinction on

the deduction rule applied last. If a time transition can be derived from $\mathcal{D}_{\mathcal{L}_1}$, there exists a deduction rule from $\mathcal{D}_{\mathcal{L}_1}$ and a substitution such that this transition is obtained from applying the substitution to the deduction rule. Using Lemma 1, this deduction rule is also part of $\mathcal{D}_{\mathcal{L}_0}$. By induction (Lemma 2 guarantees that the substituted hypotheses are also $\mathcal{L}_1(C, L)$-terms) the substituted hypotheses of the deduction rule are also derivable in $\mathcal{L}_0(C, L)$. Therefore, using the same deduction rule and the same substitution, the transition can be derived in $\mathcal{L}_0(C, L)$ as well.

Proof of right implication ($\Rightarrow$): The proof follows the same lines with the additional observation that due to Lemma 2, a proof of a transition between $\mathcal{L}_1(C, L)$-terms will never use a deduction rule that is not also contained in $\mathcal{D}_{\mathcal{L}_1}$. ⊠

6.2 The $\mathcal{L}_2$ language

The $\mathcal{L}_2$ language is defined as $\mathcal{L}_2 \equiv \mathcal{L}_1(\emptyset, \emptyset)$. The syntax of $\mathcal{L}_2$ remains unchanged w.r.t. $\mathcal{L}_1$. The deduction system of $\mathcal{L}_2$ equals the deduction system $\mathcal{D}_{\mathcal{L}_1}$, where all occurrences of C and L are replaced by $\emptyset$ (this substitution is also applied to the functions Ω and Ξ). Furthermore, mathematical equalities of the form $\mathrm{dom}(\xi) = \mathrm{dom}(\sigma) \implies \xi_\sigma = \xi$, $\mathrm{dom}(\xi^{\emptyset\emptyset}) = \emptyset \implies \sigma \cup \xi^{\emptyset\emptyset} = \sigma$ are used. Some free variables are renamed to other free variables for a more intuitive representation. After the substitution, all environments are of the form $(\emptyset, J, \emptyset, R)$. Since, there are no deduction rules which can modify the empty sets of the environment, this information is redundant. Hence, the environment is simplified to (J, R).

To illustrate how $\mathcal{D}_{\mathcal{L}_2}$ is derived from $\mathcal{D}_{\mathcal{L}_1}$, the derivation is shown below for function Ξ, function Ω and the deduction rule of the action transition of the action predicate in $\mathcal{D}_{\mathcal{L}_2}$. The other derivations can be found in Appendix A.

Substitution $[\emptyset, \emptyset / C, L]$ on function definition Ξ,

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

results in the following definition:

$$\Xi(\sigma, J) = \{\sigma' \mid \mathrm{dom}(\sigma') = \mathrm{dom}(\sigma),\ \forall_{x \in \mathrm{dom}(\sigma) \setminus J}\ \sigma'(x) = \sigma(x)\},$$

using $C = \emptyset \implies \dot{C} = \emptyset$, and bound variable ξ is substituted by bound variable σ'. The signature of the function is simplified to $\Xi \in (\Sigma \times \mathcal{P}(V)) \to \mathcal{P}(\Sigma)$.

All occurrences of function Ω in the deduction rules of $\mathcal{D}_{\mathcal{L}_2}$ are of the form $\Omega(\sigma, C, L, u, t)$ (or in abbreviated notation $\Omega_{\sigma E t}$), where $u \equiv \mathrm{true}$. Therefore, in the function definition for Ω true can be substituted for u.

Substitution $[\emptyset, \emptyset, \mathrm{true} / C, L, u]$ on function definition Ω,

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

results in the following definition:

$$
\begin{array}{ll}
\Omega(\sigma, t) = & \\
\{\rho & \\
\mid\ \rho \in [0,t] \mapsto ((\mathrm{dom}(\sigma)) \to \Lambda) & \\
,\ t \geq 0 & \\
,\ \forall x \in \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}: & \rho \downarrow x \text{ is a constant function} \\
,\ \forall x \in \mathrm{dom}(\sigma): & (\rho \downarrow x)(0) = \sigma(x) \\
,\ \forall s \in [0,t]: & \rho(s)(\mathsf{time}) = \sigma(\mathsf{time}) + s \\
\} &
\end{array}
$$

using $C = \emptyset \implies \dot{C} = \emptyset$. The signature of the function is simplified to $\Omega \in \Sigma \times T \to \mathcal{P}(T \mapsto \Sigma)$.

Substitution of $[\emptyset, \emptyset / C, L]$ on the deduction rule for the action transition of the action predicate,

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

results in the following deduction rule:

$$
\frac{\sigma' \in \Xi(\sigma, J \cup W),\ \sigma^- \cup \sigma' \models r}{(J, R) \Vdash \langle W : r \gg l_a, \sigma \rangle \xrightarrow{\sigma,\ l_a,\ \sigma'} \langle \surd, \sigma' \rangle}\ \mathcal{L}_2\text{-}1
$$

using $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \xi = \sigma$, $\mathrm{dom}(\xi') = \mathrm{dom}(\sigma) \implies \xi'_\sigma = \xi'$, and free variables ξ' and ξ^- are substituted by σ' and σ^-, respectively.

Lemma 4 *Let p and p' be closed process terms from $P_{\mathcal{L}_2}$, σ, σ' be valuations, ξ, ξ' be extended valuations, J and J' be sets of jumping variables, R and R' be recursion definitions, a be an action, ρ be a trajectory, and $t \in T$. Then*

$$\begin{aligned}
\mathcal{L}_1(\emptyset,\emptyset) \models\ & \langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \xrightarrow{\xi,a,\xi'} \langle _,\sigma',(\emptyset,J',\emptyset,R')\rangle \quad \Leftrightarrow\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle _,\sigma',(J',R')\rangle\\
\mathcal{L}_1(\emptyset,\emptyset) \models\ & \langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',(\emptyset,J',\emptyset,R')\rangle \quad \Leftrightarrow\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',(J',R')\rangle\\
\mathcal{L}_1(\emptyset,\emptyset) \models\ & \langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \overset{\xi}{\rightsquigarrow} \quad \Leftrightarrow\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \overset{\xi}{\rightsquigarrow},
\end{aligned}$$

where $\langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \xrightarrow{\xi,a,\xi'} \langle \_,\sigma',(\emptyset,J',\emptyset,R')\rangle$ is an abbreviation for $\langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',(\emptyset,J',\emptyset,R')\rangle$ for some p', and $\langle p,\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'}$ $\langle \_,\sigma',(J',R')\rangle$ is an abbreviation for $\langle p,\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',(J',R')\rangle$ for some p'.

Proof. This proof follows the same lines as the proof of Lemma 3. There are two important differences. The first difference is that the deduction rules of $\mathcal{L}_1(\emptyset,\emptyset)$ and the deduction rules of $\mathcal{L}_2$ are syntactically different. Nevertheless, by construction of the deduction system of $\mathcal{L}_2$, there is a one-to-one mapping between these sets of deduction rules that explains precisely how the application of a deduction rules from one of these languages has to be mimicked by the application of a deduction rule from the other language.

The second difference is that, due to the renaming of some variables in the original deduction rule, the substitutions that are used in deriving in the two deduction systems are not identical anymore. Again, by construction, i.e. using the mathematical identities as defined in this section and in Appendix A, also here it is clear how a substitution used for deriving a transition in $\mathcal{L}_1(\emptyset,\emptyset)$ can be transformed into a substitution used for deriving the same transition in $\mathcal{L}_2$, and vice versa. ⊠

6.3 Relating $\mathcal{L}_0(\emptyset,\emptyset)$ and $\mathcal{L}_2$

From Lemma 3 and 4 we obtain the following relation between $\mathcal{L}_0(\emptyset,\emptyset)$ and $\mathcal{L}_2$:

Corrolary 1 *Let p and p' be closed process terms from $P_{\mathcal{L}_2}$, σ, σ' be valuations, ξ, ξ' be extended valuations, J and J' be sets of jumping variables, R and R' be recursion definitions,*

a be an action, ρ be a trajectory, and $t \in T$. Then

$$\begin{aligned}
\mathcal{L}_0(\emptyset,\emptyset) \models\ & \langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \xrightarrow{\xi,a,\xi'} \langle _,\sigma',(\emptyset,J',\emptyset,R')\rangle \quad \Leftrightarrow\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle _,\sigma',(J',R')\rangle\\
\mathcal{L}_0(\emptyset,\emptyset) \models\ & \langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',(\emptyset,J',\emptyset,R')\rangle \quad \Leftrightarrow\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',(J',R')\rangle\\
\mathcal{L}_0(\emptyset,\emptyset) \models\ & \langle p,\sigma,(\emptyset,J,\emptyset,R)\rangle \overset{\xi}{\rightsquigarrow} \quad \Leftrightarrow\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \overset{\xi}{\rightsquigarrow},
\end{aligned}$$

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

6.4 Relating $\mathcal{L}_2$ and timed Chi

Let $\hbar$ be a bijective mapping that maps the process terms of the $\mathcal{L}_2$ language to the process terms of the timed χ language. It is defined as

$$\hbar([\![_{\mathrm{V}}\,\sigma_\perp,\emptyset,\emptyset\ \text{`}|\text{'}\ P\ ]\!]) = [\![_{\mathrm{V}}\,\sigma_\perp\ \text{`}|\text{'}\ \hbar(P)\ ]\!],$$

and distributes over all other operators.

Lemma 5 *Let p and p' be closed process terms from the $\mathcal{L}_2$ language, σ,σ',ξ,ξ' be valuations, J and J' be sets of jumping variables, R and R' be recursion definitions, a be an action, ρ be a trajectory, and $t \in T$. Then*

$$\begin{aligned}
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle _,\sigma',(J',R')\rangle \quad \Leftrightarrow\\
\text{timed } \chi \models\ & \langle \hbar(p),\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle _,\sigma',(J',R')\rangle\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \overset{t,\rho}{\longmapsto} \langle p',\sigma',(J',R')\rangle \quad \Leftrightarrow\\
\text{timed } \chi \models\ & \langle \hbar(p),\sigma,(J,R)\rangle \overset{t,\rho}{\longmapsto} \langle \hbar(p'),\sigma',(J',R')\rangle\\
\mathcal{L}_2 \models\ & \langle p,\sigma,(J,R)\rangle \overset{\xi}{\rightsquigarrow} \quad \Leftrightarrow\\
\text{timed } \chi \models\ & \langle \hbar(p),\sigma,(J,R)\rangle \overset{\xi}{\rightsquigarrow},
\end{aligned}$$

where $\langle p,\ \sigma,\ (J,\ R)\rangle \xrightarrow{\xi,a,\xi'} \langle \_,\ \sigma',\ (J',\ R')\rangle$ *is an abbreviation for* $\langle p,\ \sigma,\ (J,\ R)\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{p'},\sigma',(J',R')\rangle$ *for some* p', *and* $\langle \hbar(p),\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle \_,\sigma',(J',R')\rangle$ *is an abbreviation for* $\langle \hbar(p),\sigma,(J,R)\rangle \xrightarrow{\xi,a,\xi'} \langle \overset{\checkmark}{\hbar(p')},\sigma',(J',R')\rangle$ *for some* p'.

Proof. Trivial ⊠

Chapter 7

Validation of the semantics

First we consider the well-definedness of the semantics in Section 7.1. Then, in Section 7.2, some properties of the Chi semantics are given. In Section 7.3, a notion of equivalence is defined, called stateless bisimilarity [12], which is similar to the well-known notion of bisimilarity [15, 10]. It is also shown that this relation is an equivalence and a congruence for all χ operators. Some useful properties of closed χ process terms are given in Section 7.4. Many of these properties express intuitions about the meaning of the χ 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 χ 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.

7.1 Well-definedness of the semantics

In the term deduction system negative hypotheses are used in Rule 34 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* [2]. 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.

7.2 Properties of the semantics

In this section, some useful properties about the semantics of χ are introduced that can be applied in the remainder of the report (especially in the proofs of the properties in Section 7.4).

With the current set of deduction rules for the semantics of χ, the left-hand and right-hand valuation are always the same as the initial and resulting valuation of an action transition, re-

spectively. 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 that the valuation on the consistency transition is the same as the initial valuation.

The following lemma captures these facts.

Lemma 6 *Let p and p' be closed process terms, $\sigma, \sigma', \xi, \xi'$ be valuations, E and E' be environments, a be an action, ρ be a trajectory, and $t \in T$. Then*

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

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

Proof. See Lemma 5, Corollary 1, and [17, Lemma 1]. ☒

If a χ process can perform action or time transitions, the process is consistent.

Lemma 7 *Let p and p' be closed process terms, $\sigma, \sigma', \xi, \xi'$ be valuations, E and E' be environments, 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 $\exists_{p',\sigma',E'} \langle p, \sigma, E\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{smallmatrix}\checkmark\\ p'\end{smallmatrix}, \sigma', E'\rangle$.

Proof. See Lemma 5, Corollary 1, and [17, Lemma 2]. ☒

Lemma 8 *Let p and p' be closed process terms, σ and σ' be valuations, E and E' be environments, $t \in T$, and ρ be a trajectory. Then,*

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

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

Proof. See Lemma 5, Corollary 1, and [17, Lemma 3]. ⊠

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

Lemma 9 *Let p be a closed process term, σ, ξ be a valuations, J, W be sets of variables such that J and $W \subseteq \mathrm{dom}(\sigma) \setminus \{\mathsf{time}\}$, and R be a recursion definition. Then*

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

Proof. See Lemma 5, Corollary 1, and [17, Lemma 4]. ⊠

7.3 Stateless bisimilarity

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

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

Two closed process terms p and q are stateless bisimilar, denoted by $p \overset{t}{\underline{\leftrightarrow}} q$, 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 that the left-hand and right-hand valuation of model variables, as displayed on an action transition, are different from the respective valuations of model variables of the state before performing the action transition and the state reached by performing the action transition. Similarly, the valuation of model variables before and after a time transition can be different from the initial and end-point of a trajectory, respectively.

As a consequence of Lemma 6, the definition of stateless bisimilarity can be simplified considerably. Yet, with in mind future extensions of the χ 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 χ 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 χ operators.*

Proof. The deduction rules of the χ language, satisfy the *process-tyft* format of [12]. Therefore, stateless bisimilarity is a congruence. ☒

The hybrid χ language without continuous variables and algebraic variables, denoted by $\mathcal{L}_0(\emptyset, \emptyset)$ (see Chapter 6), is an operational conservative extension of the timed χ language, i.e., for closed timed χ-terms, any equality and only equalities that can be derived in timed χ can also be derived in $\mathcal{L}_0(\emptyset, \emptyset)$.

Lemma 10 *Let p and q be closed process terms from timed χ. Then*

$$\mathcal{L}_0(\emptyset, \emptyset) \models \hbar^{-1}(p) \underline{\leftrightarrow} \hbar^{-1}(q) \qquad \Leftrightarrow \qquad \text{timed } \chi \models p \overset{t}{\underline{\leftrightarrow}} q.$$

Proof. This follows immediately from the definition of $\hbar$ as described in Section 6.4, Lemma 5, Corollary 1, and the definitions of $\underline{\leftrightarrow}$ (see [17]) and $\overset{t}{\underline{\leftrightarrow}}$ (see Definition 1). ☒

7.4 Properties of the Chi operators

In this section, some properties of the operators of χ 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). For the proofs of the properties, we refer to Lemma 10 and the proofs of the corresponding properties in [17].

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

If a true predicate is emitted, the process term is simply executed. If falsity holds initially, the process term is inconsistent. 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 $p = \text{false} \rightarrow \text{skip}$. 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 τ transition, while δ 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
$b \rightarrow (p \,[]\, q)$	$\underline{\leftrightarrow}$	$b \rightarrow p \,[]\, b \rightarrow q$

If a process term is guarded by a true predicate, the process term is simply executed. 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:*

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

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:*

$u \curvearrowright \bot$	$\underline{\leftrightarrow}$	$\bot$
$p \,[]\, \bot$	$\underline{\leftrightarrow}$	$\bot$
$p \parallel \bot$	$\underline{\leftrightarrow}$	$\bot$
$\partial_{\mathcal{A}}(\bot)$	$\underline{\leftrightarrow}$	$\bot$
$\bot;\ p$	$\underline{\leftrightarrow}$	$\bot$
$\mathrm{skip};\ \bot$	$\underline{\leftrightarrow}$	δ

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.

Bibliography

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

[2] 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.

[3] 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.

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

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

[6] 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.

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

[8] 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.

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

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

[11] M. Möller. *Structure and Hierarchy in Real-Time Systems*. PhD thesis, University of Aarhus, 2002.

[12] 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.

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

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

[15] 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.

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

[17] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Syntax and consistent equation semantics of hybrid Chi. Technical Report CS-Report 04-37, Eindhoven University of Technology, Department of Computer Science, The Netherlands, 2004.

[18] 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.

[19] 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.

Appendix A

Derivation of $\mathcal{D}_{\mathcal{L}_2}$ from $\mathcal{D}_{\mathcal{L}_1}$

This appendix shows the derivation of the deduction rules of $\mathcal{D}_{\mathcal{L}_2}$ from $\mathcal{D}_{\mathcal{L}_1}$, as described in Section 6.2. Sections A.1 and A.2 show the derivation for the atomic process terms and the operators, respectively. The deduction rules from $\mathcal{D}_{\mathcal{L}_2}$ which are not shown in these sections are syntactically equivalent to their corresponding deduction rules in $\mathcal{D}_{\mathcal{L}_1}$. Deduction rules with a label of the form $\mathcal{L}_2$-n, where n denotes the number of the deduction rule, are from $\mathcal{D}_{\mathcal{L}_2}$. The other deduction rules are from $\mathcal{D}_{\mathcal{L}_1}$.

A.1 Derivation for atomic process terms

Action predicate

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

$$\frac{\sigma' \in \Xi(\sigma, J \cup W),\ \sigma^- \cup \sigma' \models r}{(J, R) \Vdash \langle W : r \gg l_a, \sigma \rangle \xrightarrow{\sigma,\ l_a,\ \sigma'} \langle \checkmark, \sigma' \rangle}\ \mathcal{L}_2\text{-}1$$

- Environment (C, J, L, R) is simplified to (J, R),
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \xi = \sigma$,
- $\mathrm{dom}(\xi') = \mathrm{dom}(\sigma) \implies \xi'_\sigma = \xi'$,
- free variables ξ' and ξ^- are renamed to σ' and σ^-, respectively,
- for function Ξ see Section 6.2.

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

$$\frac{}{E \Vdash \langle W : r \gg l_{\mathrm{a}}, \sigma \rangle \overset{\sigma}{\rightsquigarrow}} \mathcal{L}_2\text{-}2$$

- Environment $(C,\, J,\, L,\, R)$ is simplified to $(J,\, R)$, and denoted by E,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \sigma \cup \xi^{\dot{C}L} = \sigma$.

Send and receive

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J, L)}{(C,\, J,\, L,\, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \xrightarrow{\xi,\ \mathrm{isa}(h, [\xi(\mathbf{e}_n)]),\ \xi'} \langle \checkmark, \xi'_\sigma \rangle} 3$$

$$\frac{\sigma' \in \Xi(\sigma, J)}{(J,\, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \xrightarrow{\sigma,\ \mathrm{isa}(h, [\sigma(\mathbf{e}_n)]),\ \sigma'} \langle \checkmark, \sigma' \rangle} \mathcal{L}_2\text{-}3$$

- Environment $(C,\, J,\, L,\, R)$ is simplified to $(J,\, R)$,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \xi = \sigma$,
- $\mathrm{dom}(\xi') = \mathrm{dom}(\sigma) \implies \xi'_\sigma = \xi'$,
- free variable ξ' is renamed to σ',
- for function Ξ see Section 6.2.

$$\frac{\xi = \sigma \cup \xi^{\dot{C}L},\ \xi' \in \Xi(\sigma, C, J \cup \{\mathbf{x}_n\}, L),\ \xi'(\mathbf{x}_n) = \mathbf{c}_n}{(C,\, J,\, L,\, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \xrightarrow{\xi,\ \mathrm{ira}(h, [\mathbf{c}_n], \{\mathbf{x}_n\}),\ \xi'} \langle \checkmark, \xi'_\sigma \rangle} 4$$

$$\frac{\sigma' \in \Xi(\sigma, J \cup \{\mathbf{x}_n\}),\ \sigma'(\mathbf{x}_n) = \mathbf{c}_n}{(J,\, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \xrightarrow{\sigma,\ \mathrm{ira}(h, [\mathbf{c}_n], \{\mathbf{x}_n\}),\ \sigma'} \langle \checkmark, \sigma' \rangle} \mathcal{L}_2\text{-}4$$

- Environment $(C,\, J,\, L,\, R)$ is simplified to $(J,\, R)$,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \xi = \sigma$,
- $\mathrm{dom}(\xi') = \mathrm{dom}(\sigma) \implies \xi'_\sigma = \xi'$,

- free variable ξ' is renamed to σ',
- for function Ξ see Section 6.2.

$$\frac{}{(C,\, J,\, L\,,\, R) \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\, 5$$

$$\frac{}{E \Vdash \langle h \,!!\, \mathbf{e}_n, \sigma \rangle \overset{\sigma}{\rightsquigarrow}}\, \mathcal{L}_2\text{-}5$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$, and denoted by E,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \sigma \cup \xi^{\dot{C}L} = \sigma$.

$$\frac{}{(C,\, J,\, L\,,\, R) \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\, 6$$

$$\frac{}{E \Vdash \langle h \,??\, \mathbf{x}_n, \sigma \rangle \overset{\sigma}{\rightsquigarrow}}\, \mathcal{L}_2\text{-}6$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$, and denoted by E,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \sigma \cup \xi^{\dot{C}L} = \sigma$.

Consistent deadlock

$$\frac{}{(C,\, J,\, L\,,\, R) \Vdash \langle \delta, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\, 7$$

$$\frac{}{E \Vdash \langle \delta, \sigma \rangle \overset{\sigma}{\rightsquigarrow}}\, \mathcal{L}_2\text{-}7$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$, and denoted by E,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \sigma \cup \xi^{\dot{C}L} = \sigma$.

A.2 Derivation for operators

Delay enabling operator

$$E \quad \frac{\rho \in \Omega_{\sigma E t}}{\langle [p], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [p], \rho_\sigma(t) \rangle} \; 9$$

$$E \quad \frac{\rho \in \Omega(\sigma, t)}{\langle [p], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [p], \rho(t) \rangle} \; \mathcal{L}_2\text{-}9$$

- $\mathrm{dom}(\mathrm{range}(\rho)) = \mathrm{dom}(\sigma) \implies \rho_\sigma = \rho$,
- for function Ω see Section 6.2.

$$\frac{}{(C, J, L, R) \Vdash \langle [p], \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}} \; 10$$

$$\frac{}{E \Vdash \langle [p], \sigma \rangle \overset{\sigma}{\rightsquigarrow}} \; \mathcal{L}_2\text{-}10$$

- Environment (C, J, L, R) is simplified to (J, R), and denoted by E,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \sigma \cup \xi^{\dot{C}L} = \sigma$.

Guard operator

$$E \quad \frac{\begin{array}{c}\rho \in \Omega_{\sigma E t},\ \forall_{s \in (0,t)}\ \rho(s) \models \neg b, \\ \rho(0) \models b \Rightarrow \langle p, \sigma \rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma' \rangle, \\ \rho(t) \models b \Rightarrow \langle p, \rho_\sigma(t) \rangle \overset{\rho(t)}{\rightsquigarrow}\end{array}}{\langle b \to p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle b \to p, \rho_\sigma(t) \rangle} \; 20$$

$$E \quad \frac{\begin{array}{c}\rho \in \Omega(\sigma, t),\ \forall_{s \in (0,t)}\ \rho(s) \models \neg b, \\ \rho(0) \models b \Rightarrow \langle p, \sigma \rangle \overset{0,\rho\restriction\{0\}}{\longmapsto} \langle p', \sigma' \rangle \\ \rho(t) \models b \Rightarrow \langle p, \rho(t) \rangle \overset{\rho(t)}{\rightsquigarrow}\end{array}}{\langle b \to p, \sigma \rangle \overset{t,\rho}{\longmapsto} \langle b \to p, \rho(t) \rangle} \; \mathcal{L}_2\text{-}20$$

- $\mathrm{dom}(\mathrm{range}(\rho)) = \mathrm{dom}(\sigma) \implies \rho_\sigma = \rho$,
- for function Ω see Section 6.2.

$$\frac{\sigma \cup \xi^{\dot{C}L} \models \neg b}{(C,\, J,\, L\,,\, R) \Vdash \langle b \rightarrow p, \sigma \rangle \overset{\sigma \cup \xi^{\dot{C}L}}{\rightsquigarrow}}\ 22$$

$$\frac{\sigma \models \neg b}{E \Vdash \langle b \rightarrow p, \sigma \rangle \overset{\sigma}{\rightsquigarrow}}\ \mathcal{L}_2\text{-}22$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$, and denoted by E,
- $\mathrm{dom}(\xi^{\dot{C}L}) = \emptyset \implies \sigma \cup \xi^{\dot{C}L} = \sigma$.

Parallel composition operator

$$\frac{\begin{array}{c}(C,\, J \cup W,\, L\,,\, R) \Vdash \langle p, \sigma \rangle \xrightarrow{\xi, \mathrm{isa}(h,cs), \xi'} \left\langle \begin{array}{c}\checkmark \\ p' \\ \checkmark \\ p'\end{array}, \sigma' \right\rangle, \\ (C,\, J,\, L\,,\, R) \Vdash \langle q, \sigma \rangle \xrightarrow{\xi, \mathrm{ira}(h,cs,W), \xi'} \left\langle \begin{array}{c}\checkmark \\ \checkmark \\ q' \\ q'\end{array}, \sigma' \right\rangle\end{array}}{\begin{array}{c}(C,\, J,\, L\,,\, R) \Vdash \langle p \parallel q, \sigma \rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \left\langle \begin{array}{c}\checkmark \\ p' \\ q' \\ p' \parallel q'\end{array}, \sigma' \right\rangle, \\ \langle q \parallel p, \sigma \rangle \xrightarrow{\xi, \mathrm{ca}(h,cs), \xi'} \left\langle \begin{array}{c}\checkmark \\ p' \\ q' \\ q' \parallel p'\end{array}, \sigma' \right\rangle\end{array}}\ 26$$

$$\frac{(J \cup W,\, R) \Vdash \langle p, \sigma\rangle \xrightarrow{\xi,\mathrm{isa}(h,cs),\xi'} \left\langle \begin{matrix}\checkmark\\ p'\\ \checkmark\\ p'\end{matrix}, \sigma' \right\rangle, \quad (J,\, R) \Vdash \langle q, \sigma\rangle \xrightarrow{\xi,\mathrm{ira}(h,cs,W),\xi'} \left\langle \begin{matrix}\checkmark\\ \checkmark\\ q'\\ q'\end{matrix}, \sigma' \right\rangle}{(J,\, R) \Vdash \langle p \parallel q, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \left\langle \begin{matrix}\checkmark\\ p'\\ q'\\ p' \parallel q'\end{matrix}, \sigma' \right\rangle,\ \langle q \parallel p, \sigma\rangle \xrightarrow{\xi,\mathrm{ca}(h,cs),\xi'} \left\langle \begin{matrix}\checkmark\\ p'\\ q'\\ q' \parallel p'\end{matrix}, \sigma' \right\rangle}\ \mathcal{L}_2\text{-}26$$

- Environment $(C,\, J \cup W,\, L\,,\, R)$ is simplified to $(J \cup W,\, R)$,
- environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$.

Recursion variable

$$(C,\, J,\, L\,,\, R)\ \frac{\langle R(X), \sigma\rangle \xrightarrow{\alpha} \langle \begin{matrix}\checkmark\\ p'\end{matrix}, \sigma'\rangle}{\langle X, \sigma\rangle \xrightarrow{\alpha} \langle \begin{matrix}\checkmark\\ p'\end{matrix}, \sigma'\rangle}\ 36$$

$$(J,\, R)\ \frac{\langle R(X), \sigma\rangle \xrightarrow{\alpha} \langle \begin{matrix}\checkmark\\ p'\end{matrix}, \sigma'\rangle}{\langle X, \sigma\rangle \xrightarrow{\alpha} \langle \begin{matrix}\checkmark\\ p'\end{matrix}, \sigma'\rangle}\ \mathcal{L}_2\text{-}36$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$.

$$(C,\, J,\, L\,,\, R)\ \frac{\langle R(X), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{\langle X, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}\ 37$$

$$(J,\, R)\ \frac{\langle R(X), \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{\langle X, \sigma\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}\ \mathcal{L}_2\text{-}37$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$.

$$(C,\, J,\, L\,,\, R)\ \frac{\langle R(X), \sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle X, \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ 38$$

$$(J,\, R)\ \frac{\langle R(X), \sigma\rangle \overset{\xi}{\rightsquigarrow}}{\langle X, \sigma\rangle \overset{\xi}{\rightsquigarrow}}\ \mathcal{L}_2\text{-}38$$

- Environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$.

Variable scope operator

$$\frac{(C \cup \{\mathbf{x}_0'\},\, J,\, L \cup \{\mathbf{g}_0'\},\, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}_0', \mathbf{g}_0'/\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}_0'}\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix}\checkmark\\ p'\end{matrix}, \sigma'\rangle}{(C,\, J,\, L\,,\, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_{0\perp}}, \{\mathbf{x}_0\}, \{\mathbf{g}_0\} \mid\ p\ ]\!], \sigma\rangle \xrightarrow{\kappa_{\sigma\dot{C}L}(\xi,a,\xi')} \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}_0'\})[\mathbf{d}, \mathbf{x}_0/\mathbf{d}', \mathbf{x}_0'], \{\mathbf{x}_0\}, \{\mathbf{g}_0\} \mid\ p'[\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0/\mathbf{d}', \mathbf{x}_0', \mathbf{g}_0']\ ]\!], \sigma_\sigma'\rangle}\ 39$$

$$E\ \frac{\langle p[\mathbf{d}'/\mathbf{d}], \sigma \cup \sigma_{\mathrm{d}'}\rangle \xrightarrow{\xi,a,\xi'} \langle \begin{matrix}\checkmark\\ p'\end{matrix}, \sigma'\rangle}{\langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{d}_\perp}, \emptyset, \emptyset \mid\ p\ ]\!], \sigma\rangle \xrightarrow{\kappa_\sigma(\xi,a,\xi')} \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}'\})[\mathbf{d}/\mathbf{d}'], \emptyset, \emptyset \mid\ p'[\mathbf{d}/\mathbf{d}']\ ]\!], \sigma_\sigma'\rangle}\ \mathcal{L}_2\text{-}39$$

- $\{\mathbf{x}_0\} = \emptyset \implies \{\mathbf{x}_0'\} = \emptyset$,
- $\{\mathbf{g}_0\} = \emptyset \implies \{\mathbf{g}_0'\} = \emptyset$,
- $\{\mathbf{x}_0'\} = \emptyset \wedge \{\mathbf{g}_0'\} = \emptyset \implies (C \cup \{\mathbf{x}_0'\},\, J,\, L \cup \{\mathbf{g}_0'\},\, R) = (C,\, J,\, L\,,\, R)$,
- environment $(C,\, J,\, L\,,\, R)$ is simplified to $(J,\, R)$, and denoted by E,
- notation $\sigma_{\mathrm{dx}_{0\perp}}$ is simplified to $\sigma_{\mathrm{d}_\perp}$, which is defined as $\sigma_{\mathrm{d}_\perp} \in \{\mathrm{d}\} \mapsto (\Lambda \cup \perp)$. Notation $\sigma_{\mathrm{d}'\mathrm{x}_0'}$ is simplified to $\sigma_{\mathrm{d}'}$, which is defined as $\sigma_{\mathrm{d}'} \in \{\mathrm{d}'\} \mapsto (\Lambda)$,
- substitution $p[\mathbf{d}',\, \mathbf{x}_0',\, \mathbf{g}_0'/\mathbf{d},\, \mathbf{x}_0,\, \mathbf{g}_0]$ is simplified to $p[\mathbf{d}'/\mathbf{d}]$, and substitution $p[\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0/\mathbf{d}', \mathbf{x}_0', \mathbf{g}_0']$ is simplified to $p[\mathbf{d}/\mathbf{d}']$,

- substitution $[\emptyset, \emptyset/C, L]$ on function definition κ 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:

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

result in the following definition for arbitrary receive actions $\mathrm{ira}(h, cs, W)$:

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

and for all other actions:

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

where valuations ξ_σ and ξ'_σ denote $\xi \restriction \mathrm{dom}(\sigma)$ and $\xi' \restriction \mathrm{dom}(\sigma)$, respectively. The signature of function κ is simplified to $\kappa \in \Sigma \times \Sigma \times A \times \Sigma \to \Sigma \times A \times \Sigma$.

$$\frac{(C \cup \{\mathbf{x}'_0\}, J, L \cup \{\mathbf{g}'_0\}, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0/\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'_0}\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_{0\perp}}, \{\mathbf{x}_0\}, \{\mathbf{g}_0\} \mid p\,]\!], \sigma\rangle \overset{t,\rho_{\sigma\dot{C}L}}{\longmapsto} \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}', \mathbf{x}'_0\})[\mathbf{d}, \mathbf{x}_0/\mathbf{d}', \mathbf{x}'_0], \{\mathbf{x}_0\}, \{\mathbf{g}_0\} \mid p'[\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0/\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0]\,]\!], \sigma'_\sigma\rangle}\ 40$$

$$E\ \frac{\langle p[\mathbf{d}'/\mathbf{d}], \sigma \cup \sigma_{\mathrm{d}'}\rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma'\rangle}{\langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{d}_\perp}, \emptyset, \emptyset \mid p\,]\!], \sigma\rangle \overset{t,\rho_\sigma}{\longmapsto} \langle [\![_{\mathrm{V}}\, (\sigma' \restriction \{\mathbf{d}'\})[\mathbf{d}/\mathbf{d}'], \emptyset, \emptyset \mid p'[\mathbf{d}/\mathbf{d}']\,]\!], \sigma'_\sigma\rangle}\ \mathcal{L}_2\text{-}40$$

- $\{\mathbf{x}_0\} = \emptyset \implies \{\mathbf{x}'_0\} = \emptyset$,
- $\{\mathbf{g}_0\} = \emptyset \implies \{\mathbf{g}'_0\} = \emptyset$,
- $\{\mathbf{x}'_0\} = \emptyset \wedge \{\mathbf{g}'_0\} = \emptyset \implies (C \cup \{\mathbf{x}'_0\}, J, L \cup \{\mathbf{g}'_0\}, R) = (C, J, L, R)$,
- environment (C, J, L, R) is simplified to (J, R), and denoted by E,
- notation $\sigma_{\mathrm{dx}_{0\perp}}$ is simplified to $\sigma_{\mathrm{d}_\perp}$, which is defined as $\sigma_{\mathrm{d}_\perp} \in \{\mathrm{d}\} \mapsto (\Lambda \cup \perp)$, and notation $\sigma_{\mathrm{d}'\mathrm{x}'_0}$ is simplified to $\sigma_{\mathrm{d}'}$, which is defined as $\sigma_{\mathrm{d}'} \in \{\mathrm{d}'\} \mapsto (\Lambda)$,
- substitution $p[\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0/\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0]$ is simplified to $p[\mathbf{d}'/\mathbf{d}]$, and substitution $p[\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0/\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0]$ is simplified to $p[\mathbf{d}/\mathbf{d}']$,
- notation $\rho_{\sigma\dot{C}L}$ is simplified to ρ_σ which denotes $\rho \downarrow \mathrm{dom}(\sigma)$.

$$\frac{(C \cup \{\mathbf{x}'_0\}, J, L \cup \{\mathbf{g}'_0\}, R) \Vdash \langle p[\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0/\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0], \sigma \cup \sigma_{\mathrm{d}'\mathrm{x}'_0}\rangle \overset{\xi}{\rightsquigarrow}}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{dx}_{0\perp}}, \{\mathbf{x}_0\}, \{\mathbf{g}_0\} \mid p\,]\!], \sigma\rangle \overset{\xi_{\sigma\dot{C}L}}{\rightsquigarrow}}\ 41$$

$$E \quad \frac{\langle p[\mathbf{d}'/\mathbf{d}], \sigma \cup \sigma_{\mathrm{d}'} \rangle \overset{\xi}{\rightsquigarrow}}{\langle [\![_{\mathrm{V}}\, \sigma_{\mathrm{d}_\perp}, \emptyset, \emptyset \mid p \,]\!], \sigma \rangle \overset{\xi_\sigma}{\rightsquigarrow}} \mathcal{L}_2\text{-}41$$

- $\{\mathbf{x}_0\} = \emptyset \implies \{\mathbf{x}'_0\} = \emptyset$,
- $\{\mathbf{g}_0\} = \emptyset \implies \{\mathbf{g}'_0\} = \emptyset$,
- $\{\mathbf{x}'_0\} = \emptyset \wedge \{\mathbf{g}'_0\} = \emptyset \implies (C \cup \{\mathbf{x}'_0\}, J, L \cup \{\mathbf{g}'_0\}, R) = (C, J, L, R)$,
- environment (C, J, L, R) is simplified to (J, R), and denoted by E,
- notation $\sigma_{\mathrm{dx}_\perp}$ is simplified to $\sigma_{\mathrm{d}_\perp}$, which is defined as $\sigma_{\mathrm{d}_\perp} \in \{\mathrm{d}\} \mapsto (\Lambda \cup \perp)$, and notation $\sigma_{\mathrm{d}'\mathrm{x}'_0}$ is simplified to $\sigma_{\mathrm{d}'}$, which is defined as $\sigma_{\mathrm{d}'} \in \{\mathrm{d}'\} \mapsto (\Lambda)$,
- substitution $p[\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0/\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0]$ is simplified to $p[\mathbf{d}'/\mathbf{d}]$, and substitution $p[\mathbf{d}, \mathbf{x}_0, \mathbf{g}_0/\mathbf{d}', \mathbf{x}'_0, \mathbf{g}'_0]$ is simplified to $p[\mathbf{d}/\mathbf{d}']$,
- abbreviation $\xi_{\sigma\dot{C}L}$ is simplified to ξ_σ which denotes $\xi \restriction \mathrm{dom}(\sigma)$.

Recursion scope operator

$$\frac{(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p \,]\!], \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{[\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}'] \,]\!]}, \sigma' \rangle} 46$$

$$\frac{(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{p'}, \sigma' \rangle}{(J, R) \Vdash \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p \,]\!], \sigma \rangle \xrightarrow{\alpha} \langle \overset{\checkmark}{[\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}'] \,]\!]}, \sigma' \rangle} \mathcal{L}_2\text{-}46$$

- Environment $(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\})$ is simplified to $(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\})$.

$$\frac{(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle,}{(C, J, L, R) \Vdash \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p \,]\!], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}'] \,]\!], \sigma' \rangle} 47$$

$$\frac{(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle p', \sigma' \rangle,}{(J, R) \Vdash \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p \,]\!], \sigma \rangle \overset{t,\rho}{\longmapsto} \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p'[\mathbf{X}/\mathbf{X}'] \,]\!], \sigma' \rangle} \mathcal{L}_2\text{-}47$$

- Environment $(C, J, L, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\})$ is simplified to $(J, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\})$.

$$\frac{(C,\, J,\, L\,,\, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \overset{\xi}{\rightsquigarrow}}{(C,\, J,\, L\,,\, R) \Vdash \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\,]\!], \sigma \rangle \overset{\xi}{\rightsquigarrow}}\ 48$$

$$\frac{(J,\, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\}) \Vdash \langle p[\mathbf{X}'/\mathbf{X}], \sigma \rangle \overset{\xi}{\rightsquigarrow}}{(J,\, R) \Vdash \langle [\![_{\mathrm{R}}\, \{\mathbf{X} \mapsto \mathbf{q}\} \mid p\,]\!], \sigma \rangle \overset{\xi}{\rightsquigarrow}}\ \mathcal{L}_2\text{-}48$$

- Environment $(C,\, J,\, L\,,\, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\})$ is simplified to $(J,\, R \cup \{\mathbf{X}' \mapsto \mathbf{q}[\mathbf{X}'/\mathbf{X}]\})$.