Translating $\chi$ models to UPPAAL timed automata

E.M. Bortnik, J.M. van de Mortel-Fronczak,
J.E. Rooda
Abstract

Due to increasing system complexity and growing competition and costs, powerful techniques are needed to design and analyze manufacturing systems. One of the most popular techniques to do performance analysis is simulation. However, simulation-based analysis cannot guarantee the correctness of a system. Our research focuses on examining other methods to make performance analysis and functional analysis, and combining the two. One of the approaches is to translate a simulation model that is used for performance analysis to a model written in an input language of an existing verification tool. The process algebraic language $\chi$ is intended for modeling, simulation, verification and real-time control and has been used extensively to simulate large manufacturing systems. UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems and has been applied successfully in case studies ranging from communication protocols to multimedia applications. In this report, we represent a translation scheme that is used to translate simulation models written in $\chi$ language to UPPAAL timed automata. As an example we apply the translation scheme to a model of a manufacturing system and show some of the properties that can be verified in UPPAAL.
1 Introduction

Nowadays, due to increasing system complexity and growing competition and costs, industry makes high demands on powerful tools and techniques used to design and analyze manufacturing systems. One of the most popular techniques to do performance analysis is simulation. However, simulation-based analysis becomes insufficient since it cannot guarantee the correctness of a system. The objective of the TIPSy project (Tools and Techniques for Integrating Performance Analysis and System Verification) is to combine performance and functional analysis, particularly in the χ environment.

The χ language is intended for modeling, simulation, verification, and real-time control of manufacturing systems [11]. It is used to model and simulate discrete-event, continuous or combined, so-called hybrid, systems. The χ language has formal semantics which makes it suitable for verification. The language and simulator have been successfully applied to a large number of industrial cases, such as an integrated circuit manufacturing plant, a brewery and process industry plants [13].

Since we do not expect that a dedicated verification tool for χ, that would be able to compete with existing optimized model checkers, could be built within reasonable time, our aim is to translate χ models to input languages of existing verification tools. As the first step, a model was manually translated to µCRL, Promela and UPPAAL timed automata, and verified in CADP, Spin and UPPAAL, respectively [4]. In this report, a general translation scheme from χ to UPPAAL is described, and its correctness is proved. Using the scheme, the χ toolset is extended with the translator to make possible to verify χ models in UPPAAL.

UPPAAL is a tool for modeling, simulation, validation and verification of real-time systems that can be modeled as a collection of non-deterministic processes with a finite control structure and real-valued clocks [8, 14]. The UPPAAL model checking engine allows to verify properties that are expressed in the UPPAAL Requirement Specification Language. This language is a subset of timed computation tree logic (TCTL), where primitive expressions are location names, variables, and clocks from the modeled system.

The remainder of the report is organized as follows. In Section 2, the translated subset of χ is described. Then, in Section 3, the formal definition of UPPAAL timed automata is given. The translation is defined in Section 4 and proved in Section 5. The informally translated process terms are described in Section 6. In Section 7, an example of the translation of a part of a manufacturing system is shown and the properties which can be verified are described. Finally, in Section 8, conclusions are drawn.

2 Translated subset of timed χ

In order to model timed discrete-event systems only, the hybrid χ language has been simplified, resulting in timed χ [12]. In the remainder of this report, we refer to timed χ as χ.

The set M of χ models that can be translated using this translation scheme consists of models $M \in M$, where $M$ is of the form $(\partial_{\lambda}(\nu_{H}(p)), \sigma, E)$, $p \in P$ is a χ process term, $\sigma$ is a valuation. The environment $E$ is a tuple $(J, H, R)$, where $f$ is a set of jumping variables, $H$ is a set of urgent channels, and $R$ denotes a set of recursive process definitions. We restrict the environment

---

1 supported by the Dutch Organization for Scientific Research (NWO), project number 612.064.205
in the following way:

- The set of the jumping variables \( J = \emptyset \). This disallows arbitrary changes of the values of the variables, which are not used in communication.
- All channels in the model are urgent \( H = H_I \), where \( H_I \) is a set of urgent channels.
- There are no recursive process definitions \( R = \emptyset \).

Moreover, for all transitions, the domain of the valuation \( \sigma \) equals the domain of valuation \( \sigma' \),
\[
\text{dom}(\sigma) = \text{dom}(\sigma'),
\]
and environment \( E \) equals environment \( E' \).

Formally, the set \( P \) consists of process terms \( p \), where \( p \) is defined by:
\[
p ::= q \mid q || q.
\]

The set \( Q \) consists of the following process terms: skip, multi-assignment \( x_n := e_n \), send \( h!!e_n \) and receive \( h??x_n \), \( x_n \) and \( e_n \) denote the vectors \((x_1, \ldots, x_n)\) and \((e_1, \ldots, e_n)\), deadlock \( \delta \) and inconsistent process term \( \bot \), delay \( \Delta \), any delay operator \([q]\), repetition \( *q \), sequential composition \( q_1; q_2 \), and alternative composition \( q_1 || q_2 \). Formally, the set \( Q \) consists of process terms \( q \), where \( q \) is defined by:
\[
q ::= \text{skip} \mid \text{multi-assignment} x_n := e_n \mid h!!e_n \\
\mid h??x_n \mid \delta \mid \bot \\
\mid \Delta \mid [q] \mid *q \\
\mid q_1; q_2 \mid q_1 || q_2.
\]

Parallel composition is not allowed inside the process terms \( q \), since a UPPAAL model is a collection of sequential processes (represented by UPPAAL timed automata) working in parallel.

### 3 UPPAAL timed automata

The model-checker UPPAAL is based on the theory of timed automata [1], where a system is modeled as a network of timed automata [2] extended with variables, guards and invariants that might involve both clocks and variables, notion of urgency (urgent and committed locations, urgent channels), etc. There are several formal definitions of UPPAAL timed automata [2, 10, 3]. For the translation we have chosen the formal description [10] as it covers most of the features of UPPAAL timed automata that are implemented in the tool. For the expressions and assignments in UPPAAL automata we will use UPPAAL notation, where \( c = d \) denotes equality and \( x = y \) denotes assignment of the value of \( y \) to variable \( x \).

#### 3.1 The formal definition of UPPAAL timed automaton

**Definition 3.1.** A UPPAAL timed automaton \( \mathcal{A} \) is a tuple \((L, L_0, E, V, C, \text{Init}, \text{Inv}, T_1)\), where:

- \( L \) is a finite set of locations
- \( L_0 \) is the initial location

\( ^{3}\)The translation of the multi-assignment process term has following restriction. Let \( Y \) be a set of variables over which the expression \( e_n \) are defined, then the translation is restricted to the cases, where \( \forall x_i \in x_n: x_i \notin Y \).

\( ^{3}\)The translation is restricted to the cases, where \( d \in \mathbb{N}^+ \). Moreover, the value of the expression \( d \) cannot be changed during delay.
• $E$ is the set of the edges defined by $E \subseteq L \times G(C, V) \times \text{Sync} \times \text{Act} \times L$, where:
  - $G(C, V)$ is the set of constraints allowed in guards,
  - $\text{Sync}$ is a set of synchronization actions which includes actions, co-actions, and the internal $\tau_i$-action. An action $\text{send}$ over a channel $h$ is denoted by $h!$ and its co-action $\text{receive}$ is denoted by $h?$. The $\tau_i$-action is an internal action which cannot synchronize and does not have a co-action.
  - Act is a set of sequences of assignment actions of the form $x_i = e_i, \ldots, x_n = e_n$, where $e_i, \ldots, e_n$ are integer expressions, clock resets and the $\tau_i$-assignment. The $\tau_i$-assignment is an empty assignment, i.e. an assignment that does not change the values of the variables.
• $V$ denotes the set of integer variables,
• $C$ denotes the set of real-valued clocks ($C \cap V = \emptyset$),
• $\text{Init} \subseteq \text{Act}$ is a set of assignments that assigns the initial values to variables.
• Inv : $L \rightarrow \mathcal{I}_{nv}(C, V)$ is a function, that assigns an invariant to each location. $\mathcal{I}_{nv}(C, V)$ is the set of invariants over clocks $C$ and variables $V$.
• $T_i : L \rightarrow \{0, u, c\}$ the function, that assigns the type (ordinary, urgent or committed) to each location. The system cannot delay if there is a process in an urgent or committed location. The transitions via the outgoing edges of a committed location have priority.

Definition 3.2. A network of timed automata $NA$ is a tuple $(\overline{A}, I_o, V', C, H, T_{1i}, \text{Init'})$, where $\overline{A} = \{A_1, \ldots, A_n\}$ is a vector of $n$ timed automata $A_i = (I_i, I_i', E_i, C_i, \text{Init}_i, \text{Inv}_i, T_{1i})$, for $1 \leq i \leq n$. $I_o = (I_{o1}, \ldots, I_{on})$ is the initial location vector, $V'$ and $C'$ are the sets of global (shared) variables and clocks, respectively, $(V' \cap C = \emptyset)$, and $H$ is a set of channels $(V' \cap H = \emptyset)$ and $C' \cap H = \emptyset)$. The function $T_{1i} : H \rightarrow \{0, u\}$ assigns the type (ordinary or urgent) to each channel. In case $H = \emptyset$, function $T_{1i}$ is undefined and is then informally denoted by $\emptyset$. $\text{Init'}$ is the set of assignments that assigns the initial values to the global variables.

3.2 UPPAAL Semantics

Below we give the description of the UPPAAL timed automata semantics. The description is based on [145]. For simplicity sake, the notation is changed to comply with the one used in $\chi$. Furthermore, in [145] two valuation functions are defined (clock valuation $\sigma_c : C \rightarrow \mathbb{R}_{\infty}$ and variable valuation $\sigma_v : V \rightarrow \mathbb{Z}$). Here we define a single valuation function $\alpha : C \cup V \rightarrow \mathbb{R}_{\infty} \cup \mathbb{Z}$, such that $\text{dom}(\alpha) = \text{dom}(\sigma_c) \cup \text{dom}(\sigma_v)$ and $\forall c \in C : \alpha(c) = \sigma_c(c)$, $\forall v \in V : \alpha(v) = \sigma_v(v)$. The set of all possible valuations is denoted as $A$.

We use $\alpha' = \alpha(a)$ to denote how the valuation $\alpha$ is changed due to assignment $a$. Note, that if there are more than one assignment on the edge, they are performed sequentially; if $a$ is of the form $[x_i = e_i, \ldots, x_n = e_n]$, then $\forall i < j \leq n : \alpha'(x_i) = \alpha(e_i[x_j \leftarrow e_j])$, where $e_i[x_j \leftarrow e_j]$ denotes that all occurrences of $x_j$ in the expression $e_i$ are replaced by the expression $e_j$, and $\forall k > n : \alpha'(x_k) = \alpha(x_k)$. For instance, if $a$ is of the form $[x = 3, y = x + 1]$, and $\alpha(x) = 0, \alpha(y) = 1$, then $\alpha'(x) = 3, \alpha'(y) = 4$.

The state of the timed automaton is defined by the location and valuation $s = (l, \alpha)$. The initial valuations $a_{\text{Init}}(c) = 0$ for all $c \in C$, $a_{\text{Init}}(x) = \text{Init}(x)$ for all $x \in V$. Let $\alpha \models \text{Inv}(l)$ denote that $\alpha$ satisfies $\text{Inv}(l)$.

A vector of locations $\overline{l}$ is a $n$-tuple. In $\overline{l}'[l_i]$ we denote a vector where the $i$th element $l_i$ of $\overline{l}$ is replaced by $l_i'$. Similarly, with $\overline{l}'[l_i, l'_i]$ we denote a vector where the $i$th element $l_i$ of $\overline{l}$ is replaced by $l'_i$, and the $j$th element $l_j$ of $\overline{l}$ is replaced by $l'_j$. 

The semantics of a network of timed automata is defined as a transition system \( \langle S, s_0, \rightarrow \rangle \), where \( S = (L_1 \times \ldots \times L_n) \times A \) is the set of states, \( s_0 = (l_0, a_0) \) is the initial state, and \( \rightarrow \subseteq S \times S \) is the transition relation defined by the following rules.

### 3.2.1 Simple Action

An simple action \( (l, \alpha) \xrightarrow{a} (l[l_i/], \alpha') \) can be performed, if there exists an edge \( l_i \xrightarrow{g_i,h_i,a} l_i' \in E_i, l_i \in I \) such that

- The guard \( g \) is satisfied in \( \alpha \)
  \[ \alpha \models g \]
- The invariants are satisfied in \( \alpha' \)
  \[ \forall l_j \in I : \alpha' \models \text{Inv}_j(l_j) \]
- The source location is committed or there is no other committed location in \( I \)
  \[ T_I(l_i) = c \lor \forall l_j \in I, i \neq j : T_I(l_j) \neq c \]
- \( \alpha' = a(\alpha) \)

### 3.2.2 Synchronization Action

A synchronized action \( (l, \alpha) \xrightarrow{h} (l[l_i/l_j], \alpha') \) can be performed, if for \( i \neq j \) there exist \( l_i \xrightarrow{g_i,h_i,a} l_i' \in E_i, l_i \in I \) and \( l_j \xrightarrow{g_j,h_j,a} l_j' \in E_j, l_j \in I \) such that

- Both guards are satisfied in \( \alpha \)
  \[ \alpha \models (g_i \land g_j) \]
- Invariants will be satisfied
  \[ \forall l_k \in I : \alpha' \models \text{Inv}_k(l_k) \]
- One or both of the processes is in the committed location or there is not any other committed location in \( I \)
  \[ T_I(l_i) = c \lor T_I(l_j) = c \lor \forall l_k \in I : T_I(l_k) = c \]
- The valuation \( \alpha' = a_j(a_i(\alpha)) \).

### 3.2.3 Delay

A delay \( (l, \alpha) \xrightarrow{t} (l, \alpha') \) can be performed, if all following holds

- There is not any process in the committed location;
  \[ \forall l_i \in I : T_I(l_i) \neq c \]
- There is not any process in the urgent location;
  \[ \forall l_i \in I : T_I(l_i) \neq u \]
4 Translation scheme

For the purpose of translation we assume existence of a set of model variables $V$, a set of communication variables $V^h$, and a set of clocks $C$, such that $V \cap V^h = \emptyset$, and $C \cap (V \cup V^h) = \emptyset$. The set of clocks $C$ is used for the translation of the delay operator.

The translation of timed $\chi$ to UPPAAL timed automata is defined by means of three translation functions. Function $T_M : M \rightarrow NA$ translates a $\chi$ model $M$ to a UPPAAL network of automata $NA$ using function $T_p$. The function $T_p : P \rightarrow P(A_p)$ translates a $\chi$ process term $p \in P$ to a set of extended timed automata using the function $T_q$. The function $T_q : Q \rightarrow A_q$ translates a $\chi$ process term $q \in Q$ to an extended timed automaton $A_q \in A_q$. The definition of an extended timed automaton $A_q$ is based on the definition of the UPPAAL timed automaton, extended with two additional elements: $A_q = \langle L, l_0, E, V, V^h, C, Init, Inv, T_1, l_i \rangle$, where $V^h \subseteq V^h$ denotes an additional set of variables, that is used for the translation of communication actions, and $l_i$ denotes a final location. The final location $l_i \in L \cup \{\top\}$ is used for the translation of sequential and alternative composition operators. The sign $\top$ denotes that there is no final location, and $Inv(l_i) = true$, $T_1(l_i) = 0$, if $l_i \in L$.

4.1 Translation function $T_M$

The translation function $T_M$ translates a $\chi$ model $M$ of the form $(\partial_{A_{ia}}(v_{H_i}(p)), \sigma, E)$, where $A_{ia}$ is a set of the internal send and receive actions and $H_i$ is a set of channels, and it is assumed that $dom(\sigma) \setminus \{\text{time}\} \subseteq V$, $H_i \cap (V \cup V^h \cup C) = \emptyset$, to a network of UPPAAL timed automata $NA = \langle \overline{A}, l_0, V', C', H_i, T_1, Init' \rangle$. The function $T_M$ is defined as follows:

$$T_M(M) = (\overline{A}, l_0, V', \{\text{time}\}, H_i, T_1, Init')$$

where $\overline{A}$ is a vector of UPPAAL timed automata obtained by means of the function $F$. The function $F : P(A_{ia}) \rightarrow \overline{A} \times P(V^h)$ transforms a set of extended automata into a vector of UPPAAL timed automata by extracting the sets of the communication variables $V^h = \cup_{h \in H_i} V^h_i$, $V^h_i \in A_{ia}, A_{ia} \in A_{ia}$, removing the final locations $l_i \in A_{ia}$, and defining the order of the automata $A_i$ in the vector $\overline{A}$.

Furthermore, $l_0 = (l_0^0, \ldots, l_0^n)$ is a vector of the initial locations $l_0^i$ of the automata $A_i \in \overline{A}$; $V' = V^h \cup (dom(\sigma) \setminus \{\text{time}\})$. Since all the channels in the model $M$ are urgent, $\forall h \in H_i : T_1(h) = u$.

Finally, $Init' = \mathcal{R}(\sigma)$, where $\mathcal{R}(\sigma)$ translates the valuation $\{x_0 \mapsto c_0, \ldots, x_n \mapsto c_n\}$ to the set of assignments $\{x_0 = c_0, \ldots, x_n = c_n\}$.
The function $T_p : P \rightarrow P(A_s)$ translates a process term $p \in P$ to a set of extended automata and is defined in the following way:

$$T_p(p) = \begin{cases} \{T_q(q)\} & \text{if } p \in Q \\ \{T_q(q_1), T_q(q_2)\} & \text{if } p \equiv q_1 \parallel q_2, q_1, q_2 \in Q. \end{cases}$$

The function $T_q : Q \rightarrow A_s$ translates a process term $q \in Q$ to an extended automaton.

### 4.2 Translation function $T_q$

In this section, the translation function $T_q(p)$ is defined inductively.

#### 4.2.1 Translation of the atomic process terms

**Skip**

The process term skip is an abbreviation for an action predicate that can only perform an internal action without changing the valuation.

$$T_q(\text{skip}) = \langle \{l_0, l_1\}, l_0, \{(l_0, \text{true}, \tau_h, \tau_a, l_1)\} \rangle,$$

where $\text{Inv}(l_0) = \text{true}$, $\text{Inv}(l_1) = \text{true}$, $T_L(l_0) = u$, $T_L(l_1) = o$ (Fig. 1).

![Figure 1: Extended automaton $T_q(\text{skip})$](image)

**Multi-assignment**

Multi-assignment $x_n := e_n, n \geq 1$ is an abbreviation for an internal action that changes the values of the variables $x_1, \ldots, x_n$ to the values of expressions $e_1, \ldots, e_n$.

$$T_q(x_n := e_n) = \langle \{l_0, l_1\}, l_0, \{(l_0, \text{true}, \tau_h, \tau_a, l_1)\} \rangle,$$

where $\text{Inv}(l_0) = \text{true}$, $\text{Inv}(l_1) = \text{true}$, $T_L(l_0) = u$, $T_L(l_1) = o$ (Fig. 2).

**Send and Receive**

Undelayable send and receive process terms $h!!e_n$ and $h??x_n$ denote undelayable sending of expressions $e_n$ via channel $h$ and undelayable receiving via channel $h$ into variables $x_n$. 

---

7 Translation scheme
In UPPAAL, the values are not transmitted via a channel. Instead, additional shared variables $y_1, \ldots, y_n$ are used. We assume existence of a bijective function $f^*: H \times \mathbb{N} \rightarrow \mathcal{V}$ that generates unique names of the communication variables: $y_i = f^*(h, i), i \in [1, n]$.

$$\mathcal{T}_q(h!!e_n) = \left\langle \{l_0, l_1\}, l_0, \{\{l_0, \text{true, } h!, \{y_1 = e_1, \ldots, y_n = e_n\}, l_1\}\} \cup \{\{l_0, \text{true, } h, \{x_1 = y_1, \ldots, x_n = y_n\}, l_1\}\}, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, \text{T}_L, l_1 \right\rangle,$$

where $\text{Inv}(l_0) = \text{true}$, $\text{Inv}(l_1) = \text{true}$, $\text{T}_L(l_0) = u$, $\text{T}_L(l_1) = o$ (Fig. 3).

$$\mathcal{T}_q(h??x_n) = \left\langle \{l_0, l_1\}, l_0, \{\{l_0, \text{true, } h??, \{x_1 = y_1, \ldots, x_n = y_n\}, l_1\}\} \cup \{\{l_0, \text{true, } h, \{x_1 = y_1, \ldots, x_n = y_n\}, l_1\}\}, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, \text{T}_L, l_1 \right\rangle,$$

where $y_i = f^*(h, i), i \in [1, n]$, $\text{Inv}(l_0) = \text{true}$, $\text{Inv}(l_1) = \text{true}$, $\text{T}_L(l_0) = u$, $\text{T}_L(l_1) = o$ (Fig. 4).

**Deadlock**

The deadlock process term cannot perform actions or delays but it is consistent. The corresponding extended timed automaton is

$$\mathcal{T}_q(\emptyset) = \left\langle \{l_0\}, l_0, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, \text{T}_L, \top \right\rangle.$$
where Inv($l_o$) = true, $T_L(l_o) = u$.

**Inconsistent process term**

The inconsistent process term $\perp$ is inconsistent for all valuations and cannot perform any action or delay. The corresponding extended timed automaton is

$$T_\perp = \langle \{l_o\}, l_o, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, T_L, \top \rangle,$$

where Inv($l_o$) = false, $T_L(l_o) = u$.

### 4.2.2 Translation of the operators

In the translation of the operators, the extended automaton that is obtained by translating the process term $q \in Q$ is denoted by $T_q = \mathcal{A}_q^\mathcal{E}$, where $\mathcal{A}_q^\mathcal{E} = \langle \mathcal{L}^\mathcal{E}, \mathcal{E}^\mathcal{E}, \mathcal{V}^\mathcal{E}, \mathcal{V}^\mathcal{E}, \mathcal{Y}^\mathcal{E}, \mathcal{Y}^\mathcal{E}, \text{Init}^\mathcal{E}, \text{Inv}^\mathcal{E}, T_L, [l_q] \rangle$.

In a similar way, $\mathcal{A}_q^r(r) \in Q$.

For the translation some additional functions are needed. The restriction of a function $f : A \to B$ to $C \subseteq A$ is denoted by $f \mid C$. If $f$ and $g$ are functions and $\text{dom}(f) \cap \text{dom}(g) = \emptyset$, then $f \cup g$ denotes function $h$ with the domain $\text{dom}(h) = \text{dom}(f) \cup \text{dom}(g)$, where $h(c) = f(c)$ if $c \in \text{dom}(f)$, and $h(c) = g(c)$ if $c \in \text{dom}(g)$.

For arbitrary sets $\mathcal{E}, \mathcal{L}, \mathcal{Act}, \mathcal{C}$, where $\mathcal{E}$ is a set of edges, $\mathcal{L}$ is a set of locations, $\mathcal{Act}$ is a set of assignments, and $\mathcal{C}$ is a set of clocks, four functions are defined. Function $\gamma : \mathcal{P}(\mathcal{E}) \times \mathcal{L} \times \mathcal{P}(\mathcal{Act}) \to \mathcal{P}(\mathcal{E})$ transforms the set of edges by adding a set of assignments to the assignment part of all incoming edges of a location. For instance, $\gamma(E, l, \{x = 1, y = 3\})$ returns a set of edges, where the set of assignments $\{x = 1, y = 3\}$ is added to the assignment parts of all incoming edges of the location $l$. Function $\sigma_1 : \mathcal{P}(\mathcal{E}) \times \mathcal{L} \times \mathcal{L} \to \mathcal{P}(\mathcal{E})$ transforms the set of edges by replacing all occurrences of the first location as a source location with the second one. For instance, the function $\sigma_1(E, l, l')$ returns the set of edges, where all occurrences of the location $l$ as a source location are replaced with $l'$. Function $\sigma_2 : \mathcal{P}(\mathcal{E}) \times \mathcal{L} \times \mathcal{L} \to \mathcal{P}(\mathcal{E})$ transforms the set of edges by replacing all occurrences of the first location as a sink location with the second one. For instance, the function $\sigma_2(E, l, l')$ returns the set of edges, where all occurrences of the location $l$ as a sink location are replaced with $l'$. Function $\varphi : \mathcal{P}(\mathcal{E}) \times \mathcal{L} \times \mathcal{P}(\mathcal{C}) \to \mathcal{P}(\mathcal{E})$ transforms the set of edges by removing all outgoing edges of a location, that have an expression over clocks as (a part of) a guard.

### Delay

The abbreviation $\triangle d$ denotes a process term that first delays for $d$ time units, and then terminates by means of an internal action $\tau$.

To translate the delay operator, additional fresh clock variables are used. We assume that a unique name of the variable $c \in \mathcal{C}$ is generated by some bijective function $f_c : L \to \mathcal{C}$.

$$T_q(\triangle d) = \langle \{l_o\}, l_o, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, T_L, l_i \rangle,$$

where $c = f_c(l_o)$, $\text{Inv}(l_o) = (c \leq d)$, $\text{Inv}(l_i) = true$, $T_L(l_o) = o$, $T_L(l_i) = o$ (Fig. 5).

### Any delay operator

The any delay operator $[\triangle]$ allows time transitions of arbitrary duration for the behavior of
In order to translate this operator, a new delayable initial location has been defined.

\[ T_q([\text{skip}]) = \langle L_q \cup \{l_0\}, l_0, E, V^q, V^{hq}, C^q, \text{Init}^q, \text{Inv}, T_L, l_q^f \rangle, \]

where \( E = \varphi(E_q \cup \sigma_1(E_q, l_q^f, l_0, C^q) \cup \text{Inv}(l_0)) = \text{true}, \text{Inv} \mid L^q = \text{Inv}^q, T_L(l_0) = 0, \text{and } T_L \mid L^q = T_L^q. \)

As an example the extended automaton \( T_q([\text{skip}]) \) is shown in Fig. 6. There the locations \( l'_1 \) and \( l_1 \) are the initial and final locations of the automaton \( T_q([\text{skip}]) \), respectively, and \( l_0 \) is a new initial location.

Due to the introduction of the location \( l_0 \), the location \( l_q^f \) can become unreachable.

**Repetition**

Process term \( *q \) represents infinite repetition of process term \( q \). If the extended automaton \( A^q_1 = T_q(q) \) has a final location, \( l_q^f \in L^q \), then the incoming edges of the final location are redirected to the initial location, and the initializations are added to the assignment parts of these edges. If the extended automaton \( A^q_1 \) does not have a final location, \( l_q^f = \top \), then \( T_q(*q) = A^q_1 \). The resulting extended automaton is defined in the following way.

\[ T_q(*q) = \langle L, l_q^f, E, V^q, V^{hq}, C^q, \text{Init}^q, \text{Inv}^q \mid L, T_L^q \mid L, \top \rangle \]

where if \( l_q^f = \top \), then \( L = L^q \), and \( E = E^q \), otherwise \( L = L^q \setminus \{l_q^f\} \), and \( E = \sigma_2(\gamma(E^q, l_q^f, \text{Init}^q), l_q^f, l_q^f) \).

In Fig. 7 the extended automaton \( T_q(*(x_n := e_n)) \) is depicted.

**Sequential composition**

The sequential composition of process terms \( q \) and \( r \) behaves as process term \( q \) until \( q \) terminates, and then continues to behave as process term \( r \). In order to combine two automata in one it is necessary that \( L^q \cap L^r = \emptyset \) and \( C^q \cap C^r = \emptyset \). For that the renaming function
\[ f : A_q \times P(L) \times P(C) \rightarrow A_q \] is used: \( A'_q = f(T_q(r), L', C') \). If the extended automaton \( A'_q \) has a final location, the sequential composition \( q; r \) is translated by replacing the final location \( l'_f \) of the extended automaton \( A'_q \) with the initial location \( l'_0 \) of the extended automaton \( A'_r \) in the following way:

\[
T_q(q; r) = \langle (L_q \setminus \{l'_f\}) \cup L', E, V_q \cup V_r, V_{has} \cup V_{hr}, C_q \cap C', \text{Init}^q, \text{Inv}, \text{T}_L, l'_0 \rangle,
\]

where \( E = \sigma_2(\gamma(E_q, l_qf, \text{Init}^q), l'_0, l'_r) \), and \( \text{Inv} = (\text{Inv}^q \mid (L_q \setminus \{l'_f\})) \cup \text{Inv}^r \), \( \text{T}_L = (T^q_L \mid (L_q \setminus \{l'_f\})) \cup T^r_L \).

In Fig. 8 the extended automaton \( T_q(x_n := e_n; \text{skip}) \) is depicted, where \( l'_0 \) and \( l_f \) are the initial and final locations of \( T_q(\text{skip}) \), respectively, and \( l_0 \) is the initial location of \( T_q(x_n := e_n) \).

\[
T_q(q; r) = \langle L, l_0, E, V_q \cup V_r, V_{has} \cup V_{hr}, C_q \cap C', \text{Init}^q \cup \text{Init}^r, \text{Inv}, \text{T}_L, l_f \rangle.
\]

If the extended automaton \( A_q \) has no final location, \( T(q; r) = A_q \).

**Alternative composition**

Alternative composition operator \( q \parallel r \) models a non-deterministic choice between \( q \) and \( r \) for action transitions. The passage of time by itself cannot result in making a choice. In order to combine two automata in one it is necessary that \( L_q \cap L_r = \emptyset \) and \( C_q \cap C_r = \emptyset \). For that the renaming function \( f \) is used: \( A'_q = f(T_q(r), L', C') \).

The alternative composition \( q \parallel r \) is translated by adding the initial location and merging the final locations of the extended automata \( A'_q \) and \( A'_r \) in the following way:

\[
T_q(q \parallel r) = \langle L, l_0, E, V_q \cup V_r, V_{has} \cup V_{hr}, C_q \cap C', \text{Init}^q \cup \text{Init}^r, \text{Inv}, \text{T}_L, l_f \rangle.
\]
where if $l_f^q /= \top$ and $l_f^r /= \top$, then $L = ((L^q \cup L^r) \cup \{l_0\}) \setminus \{l_f^l\}$, and $l_f = l_f^l$.

![Figure 9: Extended automata $T_q(x_n := e_n[\{\text{skip}\}]$)](image)

Otherwise, $L = (L^q \cup L^r) \cup \{l_0\}$, and if $l_f^q /= \top$ then $l_f = l_f^q$, otherwise $l_f = l_f^r$.

The set of edges $E = E' \cup \sigma_1(E', l_0^q, l_0, l_0^r, l_0^r)$, where $E' = \sigma_2(E', l_0^q, l_0) \cup \sigma_2(E', l_0^r, l_0)$.

The function $\text{Inv}$ is defined as follows: $\text{Inv}(l_0) = \text{Inv}q(l_0^q) \land \text{Inv}r(l_0^r) \land \text{Inv}(L \setminus \{l_0\}) = (\text{Inv}q \cup \text{Inv}r) \cup (L \setminus \{l_0\})$.

Finally, if $T_q(l_0^q) = \{u\}$, then $T_q(l_0^q) = \{u\}$, otherwise $T_q(l_0^q) = T_q(l_0^r)$. Furthermore, $T_q(l_0) = (T_q^q \cup T_q^r) \cup (L \setminus \{l_0\})$.

The case when both automata have final locations ($T_q(x_n := e_n[\{\text{skip}\}]$) is shown in Fig. 9 where $l'_0$ and $l'_1$ are the initial and the final location of $T_q(x_n := e_n)$, $l''_0$ is the initial location of the extended automaton $T_q(\{\text{skip}\})$, and $l_0$ is a new initial location.

The case when only one automaton has a final location ($T_q(x_n := e_n[\{\text{skip}\}]$) is shown in Fig. 10 where $l''_0$ and $l'_0$ are the initial and the final location of $T_q(x_n := e_n)$, $l''_0$ is the initial location of the extended automaton $T_q(\{\text{skip}\})$, and $l_0$ is a new initial location.

![Figure 10: Extended automata $T_q(x_n := e_n[\{\text{skip}\}]$)](image)
5 Correctness of the translation scheme

5.1 Preliminaries

In this section we give some notions, which will be used in the subsequent theorems. First, we provide the semantics of the repetition and delay operators. Then, we give the definition of the corresponding states in $\chi$ and UPPAAL transition systems. After that we prove one preliminary lemma.

In UPPAAL semantics there are conditions, saying that some action is possible if there are no committed locations in $l$. Since, in the translation committed locations are never used, we skip in the proofs of these conditions.

5.1.1 The semantics of the repetition and delay operators

In [12] the semantics of the repetition and delay operators is given in terms of other language elements. Since those elements have not been included into the translated set of the process terms, we provide here the deduction rules for the repetition [9] and delay operators, which do not include other language elements. It can be shown that these deduction rules coincide with the formal semantics, though we omit the proofs here.

The semantics of the repetition operator

\[ \langle p, \sigma, E \rangle \xrightarrow{\sigma, a, \sigma'} \langle \checkmark, \sigma', E \rangle \]

5.1

\[ \langle \ast p, \sigma, E \rangle \xrightarrow{\sigma, a, \sigma'} \langle \ast p, \sigma', E \rangle \]

5.2

\[ \langle p, \sigma, E \rangle \xrightarrow{\sigma, \rho, \sigma'} \langle \ast p, \sigma', E \rangle \]

5.3

The semantics of the delay operator

\[ \sigma(d) = \emptyset \]

5.4

\[ \rho \in \Omega(\sigma, t), t \leq \sigma(d) \]

\[ \langle \Delta d, \sigma, E \rangle \xrightarrow{\sigma, \rho, \sigma} \langle \Delta(d - t), \rho(t), E \rangle \]

5.5

13 Correctness of the translation scheme
During the time transition the valuation at each time point $s \in [0, t]$ is given by $\rho(s) \in \Omega(\sigma, t)$.

5.1.2 Definition of corresponding states

The transition system of the $\chi$ model $M$ is denoted by $[M]$, and the transition system of the UPPAAL model $T_M(M)$ is denoted by $[T_M(M)]$. If the transition system contains a transition, it is denoted by $\models$.

The correspondence between states of hybrid transition system of $\chi$ and states of state transition system of UPPAAL is defined as follows.

Let $M$ be a $\chi$ model $M = (\partial_{\alpha^o}, \parallel_{1}], q_0, \models, E)$ and $T_M(M) = (\overline{A}, I_0, V', C', H, T_{11}, Init')$, where $\overline{A} = \{A_1, \ldots, A_n\}$ and $\overline{f} = \{f_1, \ldots, f_n\}$.

Let $L_i$ be a set of final locations, and $Corr : (L_1 \times \ldots \times L_m) \times P(L_i) \to (L_1 \times \ldots \times L_m) \cup \{\emptyset\}$ be a function that transforms a vector of locations $(l_1, \ldots, l_m)$ into a vector of locations $(l'_1, \ldots, l'_m)$ by removing all final locations: $\forall i, 1 \leq i \leq n, l_i \in L_i : (\forall j, 1 \leq j \leq m, l'_j = l_j)$, and preserving all non-final location: $\forall j, 1 \leq j \leq m : (\exists i, 1 \leq i \leq n : l'_j = l_i)$. Moreover, the order of the locations is not changed: $\forall i, j, 1 \leq i \leq n, 1 \leq j \leq n, i < j, l_i, l_j \notin L_f : (\exists l, l_k, l_r, 1 \leq k \leq m, 1 \leq r \leq m, k < r : l_i = l'_k, l_j = l'_r)$. If all locations are final, the function returns $\emptyset$, where $\emptyset$ denotes an empty vector. A vector is a mapping with a domain $\{1, \ldots, n\}$; an empty vector shows the absence of such mapping.

Definition 5.1.

1. The initial states $(\partial_{\alpha^o}, \parallel_{1}], q_0, \models, E)$ and $(l_1, \ldots, l_n, \alpha^o)$ are corresponding if $\forall i : 1 \leq i \leq n, l_i \in T_q(q_0), l_i$ is an initial location of the automaton $T_q(q_0)$, and $\alpha^o \mid \operatorname{dom}(\alpha^o) = \sigma_o$.

2. The intermediate states $(\partial_{\alpha^o}, \parallel_{1}], q_0, \models, E')$ and $(l'_1, \ldots, l'_m, \alpha')$ are corresponding if $Corr((l'_1, \ldots, l'_m), L_i) = (l_1, \ldots, l_n)$ and $\forall i : 1 \leq i \leq m, l'_i$ is an initial location of $T_q(q_0)$, and $\alpha' \mid \operatorname{dom}(\alpha') = \sigma'$.

3. The final states $(\chi, \sigma'', E')$ and $(l'_1, \ldots, l'_n, \alpha'')$ are corresponding if $Corr((l'_1, \ldots, l'_n), L_i) = \emptyset$, and $\alpha'' \mid \operatorname{dom}(\alpha'') = \sigma''$.

4. There are no other corresponding states.

5.1.3 Preliminary lemma

To prove the correctness of the translation scheme we need two additional lemmas. In $\chi$ semantics the process term $[q]$ has the same action behaviour as $q$ and it may also perform an arbitrary time transition. To translate it we added a new location into the automaton $T_q(q)$ in which the automaton can delay. With the first lemma we show that the action behaviour stays the same nevertheless.

Lemma 1.

\[ [T_M([q], \sigma, E)] \models (l, \alpha) \leftrightarrow (l', \alpha') \iff [T_M([q], \sigma, E)] \models (l', \alpha) \leftrightarrow (l', \alpha'), \]

where $\bar{l} = (l'_1, l'_2)$, $l' = (l'_1)$, $\bar{l}' = (l'_2)$, $l'_2$ is the initial location of $T_q(q)$, $l'_1$ is a location of $T_q(q)$, $l_0$ is the initial location of $T_q([q])$.\]
Proof (Lemma 2).

First, we prove that \( T_M((q, \sigma, E)) \models (l, \alpha) \xrightarrow{a} (l', \alpha') \Rightarrow T_M((q, \sigma, E)) \models (l, \alpha) \xrightarrow{a} (l', \alpha'). \)

From UPPAAL semantics it follows that \( T_q(q) \) has the initial location \( l_0 \), a location \( l_i \), and the edge \( e_i = (l_j, g^i, \tau_i, a^i, l_r^i) \). Moreover, \( \alpha \models \text{Inv}(l_j), \alpha \models g^i, \alpha' \models \text{Inv}(l_r^i) \), and \( \alpha' = a^i(\alpha) \).

Applying \( T_q(q) \) adds a new initial location \( l_0 \) into an extended automaton \( T_q(q) \), such that \( \text{Inv}(l_0) \) is true. Moreover, new edges are added such that if the location \( l_j \) has an outgoing edge leading to a location \( l_r^i \) with no expression over the clock in the guard, then \( l_0 \) will have an outgoing edge with the same guard, synchronization label and assignment, leading to the location \( l_r^i \).

In case if \( g^i \) does not contain expression over clocks, the proof is straightforward, since the automaton \( T_M((q, \sigma, E)) \) will have the edge \( e = (l_0, g^i, \tau_i, a^i, l_r^i) \).

In case if \( g^i \) contains an expression over clocks, we prove that the automaton \( T_M((q, \sigma, E)) \) cannot make a transition over this expression, thus the lemma holds trivially. The proof is done by contradiction. From the translation scheme, \( g^i \) can contains an expression \( e = d \), where \( c \) is some clock. Moreover, \( \alpha(c) = 0, \alpha(d) \in \mathbb{N} \). \( \text{Inv}(l_j) \) contains an expression \( \leq d \). From the UPPAAL semantics, \( T_M((q, \sigma, E)) \models (l, \alpha) \xrightarrow{a} (l', \alpha') \), only if \( \alpha(c) = \alpha(d) \). This leads to the contradiction: \( \alpha(c) = 0, \alpha(d) \in \mathbb{N} \), thus \( \alpha(c) 
leq \alpha(d) \), which means that the action transition is not possible via an edge with an expression over clocks in its guard.

To prove that \( T_M((q, \sigma, E)) \models (l, \alpha) \xrightarrow{a} (l', \alpha') \Rightarrow T_M((q, \sigma, E)) \models (l, \alpha) \xrightarrow{a} (l', \alpha') \).

From UPPAAL semantics it follows that \( T_q(q) \) has the initial location \( l_0 \), a location \( l_i \), and the edge \( e_i = (l_j, g^i, \tau_i, a^i, l_r^i) \). Moreover, \( \alpha \models \text{Inv}(l_j), \alpha \models g^i, \alpha' \models \text{Inv}(l_r^i) \), and \( \alpha' = a^i(\alpha) \).

In \( \chi \) language the process term \( q \equiv \Delta d \) can perform a delay for \( t \) time units \( (t \leq d) \) and then continue as \( q' \equiv \Delta(d - t) \). According to the definition of the corresponding states we need to prove that the initial location of the extended automaton \( T_q(d - t) \) is in the location vector. The problem is that the process term \( T_q(d - t) \) is not translated. To resolve this case we will need the following lemma.

Lemma 2. Let \( NA = \langle (A), (l), V, \{\text{time}\}, H, I \rangle \) be a network of UPPAAL automata, and let a location \( l \) be the initial location of \( T_q(\Delta d) \), \( \text{Inv}(l) = c \leq d \).

Let \( NA' = \langle (A'), (l'), V, \{\text{time}\}, H', I \rangle \) be a network of UPPAAL automata, and let a location \( l' \) be the initial location of \( T_q(\Delta d') \), \( \text{Inv}(l') = c' \leq d' \), where \( d' = d - t_0 \).

Then, if \( \alpha(c) = t_0 \land \alpha(c') = c \)

\[ \llbracket NA \rrbracket \models (l, \alpha) \xrightarrow{t_0} (l', \alpha') \Leftrightarrow \llbracket NA' \rrbracket \models (l', \alpha') \xrightarrow{t_0} (l', \alpha'). \]

Proof (Lemma 2).

The proof is straightforward. According to the UPPAAL semantics, \( NA \) can make a time transition for \( t \) time units if the invariant stays satisfied, i.e. \( t \in [0, (d - t_0)] \). Similarly, \( NA' \) can make a time transition for \( t \) time units if the invariant stays satisfied, i.e. \( t \in [0, d'] \). Knowing that \( d' = d - t_0 \), we can say that the time behaviour of these two automata is the same.
5.2 Theorem 1

Theorem 1. Let $p$ and $p'$ be closed process terms, $\sigma$, $\sigma'$, $a$, $a'$ be valuations, such that $\text{dom}(\sigma) = \text{dom}(\sigma')$ and $\text{dom}(a) = \text{dom}(a')$. Let $\mathcal{M}$ be an environment, $I$ be a location vector, and the states $\langle p, \sigma, E \rangle$ and $\langle \tilde{I}, \alpha \rangle$ be corresponding states. Then:

Lemma 1.1. for any non-communication action $a$
$$\mathcal{M} \models (p, \sigma, E) \frac{\frac{\text{Val}(\mathcal{M})}{\langle \tilde{I}, \alpha \rangle}}{\langle \tilde{I}, \alpha' \rangle} = (p', \sigma', E')$$
and $\langle \tilde{I}, \alpha' \rangle$ is a corresponding state.

Lemma 1.2. for any non-communication action $a$
$$\mathcal{M} \models (p, \sigma, E) \frac{\frac{\text{Val}(\mathcal{M})}{\langle \tilde{I}, \alpha \rangle}}{\langle \tilde{I}, \alpha' \rangle} = (p', \sigma', E')$$
and $\langle \tilde{I}, \alpha' \rangle$ is a corresponding state.

Lemma 1.3. for any communication action $ca$
$$\mathcal{M} \models (p, \sigma, E) \frac{\frac{\text{Val}(\mathcal{M})}{\langle \tilde{I}, \alpha \rangle}}{\langle \tilde{I}, \alpha' \rangle} = (p', \sigma', E')$$
and $\langle \tilde{I}, \alpha' \rangle$ is a corresponding state.

Lemma 1.4. for any communication action $ca$
$$\mathcal{M} \models (p, \sigma, E) \frac{\frac{\text{Val}(\mathcal{M})}{\langle \tilde{I}, \alpha \rangle}}{\langle \tilde{I}, \alpha' \rangle} = (p', \sigma', E')$$
and $\langle \tilde{I}, \alpha' \rangle$ is a corresponding state.

Lemma 1.5. for any time transition
$$\mathcal{M} \models (p, \sigma, E) \frac{\frac{\text{Val}(\mathcal{M})}{\langle \tilde{I}, \alpha \rangle}}{\langle \tilde{I}, \alpha' \rangle} = (p', \sigma', E')$$
and $\langle \tilde{I}, \alpha' \rangle$ is a corresponding state.

The proofs are by induction on the structure of the closed process term $p$.

Proof (Lemma 1.1).
According to the rules [1.22] [1.25] the process term $(\partial_{a_i}(v_{i+1}(p)), \sigma, E)$ can perform an action $a$ and then terminate if a process term $p$ can perform an action $a$ and terminate:

1. Let us consider the case when $p \in Q$. In the $\chi$ semantics there are no termination transition rules (with $a$ as specified) defined for deadlock, inconsistent process term, send, receive, sequential composition and repetition, therefore the lemma holds trivially for these cases. The proofs for the other cases are given below.

(a) The process term skip is an abbreviation for an action predicate $\emptyset : true \triangleright \tau$ (13) that performs an internal action $\tau$ without changing valuation $\sigma = \sigma'$ and then terminates (1.3). According to the translation scheme, the corresponding extended automaton $\mathcal{T}_\mathcal{M}(\text{skip})$ has the initial location $l_0$, final location $l$, the only edge $\langle l_0, \text{true}, \tau, l_1 \rangle$, and $\text{Inv}(l_0) = \text{true}, \text{Inv}(l) = \text{true}$.

Since the translation functions $\mathcal{T}_\mathcal{P}$ and $\mathcal{T}_\mathcal{M}$ do not change locations and edges,
$$\mathcal{M} \models (\partial_{a_i}(v_{i+1}(\text{skip})), \sigma, E) \quad \mathcal{M} \models (l, \alpha) \frac{\frac{\text{Val}(\mathcal{M})}{\langle \tilde{I}, \alpha' \rangle}}{\langle \tilde{I}, \alpha' \rangle} = (l, \alpha)$$
According to the UPPAAL semantics, the empty assignment $\tau_a$ does not change the valuation, $\alpha = \alpha'$. Hence, $\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')$. Since the location $l$ is a final
location \((\text{Corr}(l_0), \{l_0\}) = \emptyset\) and \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\), the states \(\langle \check{\nu}, \sigma', E' \rangle\) and \(\langle \check{\nu}, \alpha' \rangle\) are corresponding.

(b) Multi-assignment \(\mathbf{x}_n := e_n, n \geq 1\) is an abbreviation for an action predicate \(\{\mathbf{x}_n : x_i = e_i, \ldots, x_n = e_n \Rightarrow \tau\}\) that performs an internal action \(\tau\) and assigns the values of the expressions \(e_i, \ldots, e_n\) to the variables \(x_i, \ldots, x_n\) respectively. According to the translation scheme, the corresponding extended automaton \(T_\nu(x_n := e_n)\) has the initial location \(l_0\), final location \(l_n\), and the only edge \(\langle l_0, \nu, \tau, \{x_i := e_i, \ldots, x_n := e_n\}, l_n\rangle\), and \(\text{Inv}(l_0) = \text{true}, \text{Inv}(l_n) = \text{false}\).

Since the translation functions \(T_\nu\) and \(T_M\) do not change locations and edges, \(\| T_M((\partial_{\mathbf{a}_n}(v_{1:k}(x_n := e_n)), \sigma, E)) \| \models \langle \check{\nu}, \alpha \xrightarrow{a} \langle \check{\nu}/l_0, \alpha' \rangle\).\)

According to the \textit{Uppaal} semantics, the assignment action \(\{x_i = e_i, \ldots, x_n = e_n\}\) changes the values of the variables \(x_i\), such that \(\forall i, j < i \leq n : \alpha'(x_i) = \alpha(e_i[x_j := e_j])\), and \(\forall k > n : \alpha'(x_k) = \alpha(x_k)\). Since the translation of the multi-assignment is restricted so that no variable on the left side can be used in any expression on the right side of the assignment, \(\alpha'(x_i) = \alpha(e_i)\). Hence, \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\). Since the location \(l_i\) is a final location \((\text{Corr}((l_0), \{l_i\}) = \emptyset)\) and \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\), the states \(\langle \check{\nu}, \sigma', E' \rangle\) and \(\langle \check{\nu}, \alpha' \rangle\) are corresponding.

(c) The process term \(\Delta d\) can perform an internal action \(\tau\) without changing valuation \(\sigma' = \sigma \upharpoonright \text{dom}(\tau)\) and terminate if \(\sigma(d) = \nu\) \(\sigma(d) = \nu\) \(\sigma(d) = \nu\) \(\sigma(d) = \nu\). According to the translation scheme, the corresponding extended automaton \(T_\nu(\Delta d)\) has the initial location \(l_0\), final location \(l_n\), the only edge \(\langle l_0, \nu, \tau, \{l_i := l_i\}, l_n\rangle\), and \(\text{Inv}(l_0) = \text{true}, \text{Inv}(l_n) = \text{false}\).

Since the translation functions \(T_\nu\) and \(T_M\) do not change locations and edges, \(\| T_M((\partial_{\mathbf{a}_n}(v_{1:k}(\Delta d)), \sigma, E)) \| \models \langle \check{\nu}, \alpha \xrightarrow{a} \langle \check{\nu}/l_0, \alpha' \rangle\).\)

According to the \textit{Uppaal} semantics, the empty assignment \(\tau\) does not change the valuation, \(\alpha = \alpha'\). Hence, \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\). Since the location \(l_i\) is a final location \((\text{Corr}((l_0), \{l_i\}) = \emptyset)\) and \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\), the states \(\langle \check{\nu}, \sigma', E' \rangle\) and \(\langle \check{\nu}, \alpha' \rangle\) are corresponding.

(d) According to the \(\chi\) semantics, the process term \([q]\) can perform an action \(a\) and then terminate if \(q\) can perform an action \(a\) and then terminate if \(q\) can perform an action \(a\) and then terminate if \(q\) can perform an action \(a\) and then terminate if \(q\). As the induction step, it is assumed that \(\models M \models \langle q, \sigma, E \rangle \xrightarrow{a, a, \alpha} \langle \check{\nu}, \sigma', E', \nu \rangle\), and the corresponding extended automaton \(T_\nu([q])\) has the initial location \(l_0\), final location \(l_0\), \(l_n\), and the edge \(e = \langle l_0, g^q, \tau, l_1, l_n \rangle\). Furthermore, \(\alpha \models \text{Inv}(l_0), \alpha \models g^q,\) and \(\alpha' \models \text{Inv}(l_n)\). Moreover, \(\alpha' \models \text{dom}(\sigma')\).

According to the lemma \(1\) the extended automaton \(T_\nu([q])\) will have the initial location \(l_0\), final location \(l_0\), \(l_n\), and the edge \(e = \langle l_0, g^q, \tau, l_1, l_n \rangle\). Furthermore, \(\text{Inv}(l_0) = \text{true}, \alpha \models g^q,\) and \(\alpha' \models \text{Inv}(l_n)\). Moreover, \(\alpha' \models \text{dom}(\sigma')\).

Since the functions \(T_\nu\) and \(T_M\) do not change the locations, edges and invariants, \(\| T_M((\partial_{\mathbf{a}_n}(v_{1:k}([q])), \sigma, E)) \| \models \langle \check{\nu}, \alpha \xrightarrow{a} \langle \check{\nu}/l_0, \alpha' \rangle\).\)

Since \(l_0\) is a final location \((\text{Corr}((l_0), \{l_0\}) = \emptyset)\) and \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\), the states \(\langle \check{\nu}, \sigma', E' \rangle\) and \(\langle \check{\nu}, \alpha' \rangle\) are corresponding.

(e) The process term \(q_1 \| q_2\) can perform an action \(a\) and then terminate, only if either \(q_1\) or \(q_2\) can perform this action and terminate \(\text{[T-1]}\). Since the proofs for both cases are similar, only the proof for the case, when \(q_1\) can make an action \(a\) and terminate, has been given.

As the induction step, it is assumed that \(\models M \models \langle q_1, \sigma, E \rangle \xrightarrow{a, a, \alpha} \langle \check{\nu}, \sigma', E' \rangle\), and the corresponding extended automaton \(T_\nu([q_1])\) has the initial location \(l_0\), final location \(l_0\), \(l_n\), and the edge \(e = \langle l_0, g^q, \tau, l_1, l_n \rangle\). Furthermore, \(\alpha \models \text{Inv}(l_0), \alpha \models g^q,\) and \(\alpha' \models \text{Inv}(l_n)\). Moreover, \(\sigma' = \alpha' \upharpoonright \text{dom}(\sigma')\).
The function $T_0(q_1 \parallel q_2)$ adds a new initial location $l_o$, such that $\text{Inv}(l_o) = \text{Inv}(l^0_1) \land \text{Inv}(l^0_2)$, and redirects the edges so that $\forall e \in T_0(q_1) \land e = \langle l^0_1, g^0, \tau_h, a^0, l^0_1 \rangle ; \exists e' \in T_0(q_1 \parallel q_2), e' = \langle l_o, g^0, \tau_h, a^0, l_o \rangle$, where $l_o, l_f$ are the initial and final locations of $T_0(q_1 \parallel q_2)$.

Since the functions $T_p$ and $T_M$ do not change the locations, edges and invariants, $\not\models T_M(M) \implies (l, a) \xrightarrow{a} (l_0, \alpha')$.

Since $l_f$ is the final location ($\text{Corr}(l_f), \{l_f\}) = \varnothing$) and $\sigma' = \alpha' \mid \text{dom}(\sigma)$, the states $(\sqrt{\cdot}, \sigma', E')$ and $(\tilde{l}, \alpha')$ are corresponding.

2. If $p$ is of the form $q_1 \parallel q_2$, the lemma holds trivially, since in $\chi$ semantics there are no termination transition rules (with $a$ as specified) defined for parallel composition.

\[\square\]

Proof (Lemma 1.2).

According to the rules [T-22] [T-25] the process term $\langle \partial_{\alpha_*}(v_{\text{hi}}(p)), \sigma, E \rangle$ can perform an action $a$ and then continue as $p'$ if a process term $p$ performs an action $a$ and continue as $p'$.

1. Let’s first consider the case, when $p \in Q$. In the $\chi$ semantics there is no transition rules (with $a$ as specified) defined for skip, assignment, deadlock, inconsistent process term, send, receive, and delay, therefore the lemma holds trivially for these cases.

(a) According to the $\chi$ semantics, the process term $[q]$ can perform an action $a$ and then continue as $p'$ if $q$ can perform an action $a$ and then continue as $p'$ (T-8). As the induction step, it is assumed that if $\not\models (q, \sigma, E) \implies (q, \sigma, E) \xrightarrow{a \alpha'} (p', \sigma', E')$, then the corresponding extended automaton $T_0(q)$ has the initial location $l^0_0$, location $l^0_i /\sim l^0_f$, and the edge $e = \langle l^0_0, g^0, \tau_h, a^0, l^0_i \rangle$. Furthermore, $\alpha \models \text{Inv}(l^0_0), \alpha \models g^0$, and $\alpha' \models \text{Inv}(l^0_f)$. Moreover, the location $l^0_i$ is an initial location of the extended automaton $T_0(p')$, and $\sigma' = \alpha' \mid \text{dom}(\sigma)$.

Applying $T_0([q])$ adds a new initial location $l_o$ into an extended automaton $T_0(q)$, such that $\text{Inv}(l_o) = \text{true}$. The outgoing edges as the location $l^0_i$ are duplicated and reconnected to the location $l_o$, except for the edges with clock expressions in guards. From the proof of the Lemma 11 follows that the extended automaton $T_0([q])$ has the edge $e = \langle l_o, g^0, \tau_h, a^0, l^0_i \rangle$.

Since the functions $T_p$ and $T_M$ do not change the locations, edges and invariants, $\not\models T_M(\langle \partial_{\alpha_*}(v_{\text{hi}}([q])), \sigma, E \rangle) \implies (l, a) \xrightarrow{a} (l_0, \alpha')$.

Since $l^0_i$ is an initial location of $T_0(p')$ ($\text{Corr}(l^0_i), \{l^0_i\}) = \varnothing$) and $\sigma' = \alpha' \mid \text{dom}(\sigma)$, the states $(p', \sigma', E')$ and $(\tilde{l}, \alpha')$ are corresponding.

(b) The process term $q_1; q_2$ can perform an action $a$ and then continue as $p'$ in two cases: if $q_1$ performs an action $a$ and terminates, $p' \equiv q_2$, (T-13) or if $q_1$ performs an action $a$ and then continues as $q_1', p' \equiv q_1' ; q_2$. (T-12).

- Let’s first consider the case when $q_1$ can perform an action $a$ and terminate. As the induction step, it is assumed that the corresponding extended automaton $T_0(q_1)$ has the initial location $l^0_0$, the final location $l^0_f$ and the edge $\langle l^0_0, g, \tau_h, a^0, l^0_f \rangle$, such that $\text{Inv}(l^0_0)$ and $g$ are evaluated as true in $a$, and $\text{Inv}(l^0_f) = \text{true}$. Furthermore, the extended automaton $T_0(q_1)$ has the initial location $l^0_0$. Moreover, $\forall x, \sigma(x) /\equiv \sigma'(x) : x := e \in a^0, e = \sigma'(x)$. 

18
The translation function $T_\delta(q; q_i)$ reconnects all input edges of the location $l_i^0$ to the location $l_0^0$ and removes $l_i^0$. Since the functions $T_\delta$ and $T_\delta$ do not change the locations, edges and invariants, $\models T_M(\langle \partial_{\mathcal{A}_0}(v_{\mathcal{A}_0}(q_i; q_i)), \sigma, E \rangle) \models^{\langle \overline{l}, \alpha \rangle} \langle \overline{l}_l^0, \overline{l}_f^0 \rangle, \alpha')$.

Since $l_i^0$ is an initial location of $T_q(p') (\text{Corr}(l_i^0), \{l_i^0\}) = \langle l_i^0 \rangle)$ and $\sigma' = \alpha' \mid \text{dom}(\sigma')$, the states $(p', \sigma', E')$ and $(\overline{l}, \alpha')$ are corresponding.

- Let’s consider the case when $q_i$ performs an action $a$ and then continues as $q_i'$. As the induction step, it is assumed that the corresponding extended automaton $T_q(q_i)$ has the initial location $l_i^0$, a non-final location $l_i^0$, and the edge $\langle l_i^0, g^b, \tau_0, a_i^0, l_i^0 \rangle$, such that $\text{Inv}(l_i^0)$ and $g^b$ are evaluated as true in $\alpha'$, and $\text{Inv}(l_i^0) = \text{true}$. Furthermore, the extended automaton $T_q(q_i)$ has the initial location $l_i^0$. Moreover, $\alpha' \mid \text{dom}(\sigma') = \sigma'$.

Since the translation function $T_q(q_i; q_i)$ does not change the locations $l_i^0$, $l_i^0$ and the edge $\langle l_i^0, g, \tau_0, a_i^0, l_i^0 \rangle$, and the functions $T_\delta$ and $T_M$ do not change the locations, edges and invariants, $\models T_M(\langle \partial_{\mathcal{A}_0}(v_{\mathcal{A}_0}(q_i; q_i)), \sigma, E \rangle) \models^{\langle \overline{l}, \alpha \rangle} \langle \overline{l}_l^0, \overline{l}_f^0 \rangle, \alpha')$.

Since $l_i^0$ is an initial location of $T_q(p') (\text{Corr}(l_i^0), \{l_i^0\}) = \langle l_i^0 \rangle)$ and $\sigma' = \alpha' \mid \text{dom}(\sigma')$, the states $(p', \sigma', E')$ and $(\overline{l}, \alpha')$ are corresponding.

(c) The process term $q_i \parallel q_i$ can perform an action $a$ and then continue as $p'$, only if either $q_i$ can perform an action $a$ and continue as $q_i'$, $p' \equiv q_i'$, or $q_i$ can perform an action $a$ and continue as $q_i'$, $p' \equiv q_i'$. Since the proofs for both cases are similar, only the proof for the case, when $q_i$ can make an action $a$ and then continue as $q_i'$, has been given.

As the induction step, it is assumed that $\models \langle q_i, \sigma, E \rangle \models^{\langle \overline{l}, \alpha \rangle} \langle q_i, \sigma, E \rangle \overset{\epsilon_a}{\rightarrow} \langle q_i', \sigma', E' \rangle$, and the corresponding extended automaton $T_q(q_i)$ has the initial location $l_i^0$, a non-final location $l_i^0$, and $\exists \epsilon_i \in \langle l_i^0, g, \tau_0, a_i^0, l_i^0 \rangle$. Furthermore, $\alpha \models \text{Inv}(l_i^0)$, $\alpha \models \text{Inv}(l_i^0)$, $\alpha \models \text{Inv}(l_i^0)$, and $\alpha' \models \text{Inv}(l_i^0)$. The location $l_i^0$ is the initial location of the extended automaton $T_q(q_i)$. Moreover, $\alpha' \mid \text{dom}(\sigma') = \sigma'$.

The function $T_q(q_i \parallel q_i)$ adds a new initial location $l_0$, such that $\text{Inv}(l_0) = \text{Inv}(l_0) \land \text{Inv}(l_0)$, and redirects the edges so that $\forall \epsilon_i \in T_q(q_i), \epsilon_i = \langle l_i^0, g, \tau_0, a_i^0, l_i^0 \rangle : \exists \epsilon_i \in T_q(q_i \parallel q_i), \epsilon_i' = \langle l_0, g, \tau_0, a_i^0, l_i^0 \rangle$, where $l_0$ is the initial location of $T_q(q_i \parallel q_i)$.

Since the functions $T_\delta$ and $T_M$ do not change the locations, edges and invariants, $\models T_M(\langle \partial_{\mathcal{A}_0}(v_{\mathcal{A}_0}(q_i \parallel q_i)), \sigma, E \rangle) \models^{\langle \overline{l}, \alpha \rangle} \langle \overline{l}_l^0, \overline{l}_f^0 \rangle, \alpha')$.

Since $l_i^0$ is the initial location of the extended automaton $T_q(q_i) (\text{Corr}(l_i^0), \{l_i^0\}) = \langle l_i^0 \rangle)$ and $\sigma' = \alpha' \mid \text{dom}(\sigma')$, the states $(q_i', \sigma', E')$ and $(\overline{l}, \alpha')$ are corresponding.

(d) According to the $\chi$ semantics, $\ast q$ can make an action $a$ and continue as $p'$ if $q$ can make an action $a$ and terminate, $p' \equiv \ast q$, or if $q$ can make an action $a$ and continue as $q_i'$. 

- In case when $p' \equiv \ast q$, as an induction step it is assumed that the corresponding extended automaton $T_q(q_i)$ has the initial location $l_i^0$, final location $l_i^0$, and the edge $\langle l_i^0, g, \tau_0, a_i^0, l_i^0 \rangle$, and $\text{Inv}(l_i^0)$, and $g^b$ are evaluated as true in the valuation $\alpha$, and $\text{Inv}(l_i^0) = \text{false}$. Moreover, $\alpha' \mid \text{dom}(\sigma') = \sigma'$.

Since the functions $T_\delta$ and $T_M$ do not change the locations, edges and invariants, $\models T_M(\langle \partial_{\mathcal{A}_0}(v_{\mathcal{A}_0}(\ast q)), \sigma, E \rangle) \models^{\langle \overline{l}, \alpha \rangle} \langle \overline{l}_l^0, \overline{l}_f^0 \rangle, \alpha')$.

Since $l_i^0$ is the initial location of the extended automaton $T_q(q_i) (\text{Corr}(l_i^0), \{l_i^0\}) = \langle l_i^0 \rangle)$ and $\sigma' = \alpha' \mid \text{dom}(\sigma')$, the states $(\ast q, \sigma', E')$ and $(\overline{l}, \alpha')$ are corresponding.
In case when \( p' \equiv q' ; * q \), as an induction step it is assumed that the corresponding extended automaton \( T_q(q) \) has the initial location \( l^0 \), non-final location \( l^1 \), and the edge \( (l^0, g^q, \tau_h, n^q, l^1) \). The location \( l^1 \) is the initial location of the extended automaton \( T_q(q'; * q) \). Moreover, \( \text{Inv}(l^1) \) and \( g^q \) are evaluated as true in the valuation \( a \), and \( \text{Inv}(l^0) \) is evaluated as true in \( a' \). Furthermore, \( a' \upharpoonright \text{dom}(\sigma) = \sigma' \).

The translation function \( T_q(q) \) does not affect non-final locations and the edges between them. Hence, the corresponding extended automaton \( T_q(q) \) has the edge \( (l^0, g^q, \tau_h, n^q, l^1) \). Moreover, both \( \text{Inv}(l^0) \) and \( g^q \) are evaluated as true in \( a \), and \( \text{Inv}(l^0) \) is evaluated as true in \( a' \).

Since the functions \( T_p \) and \( T_M \) do not change the locations, edges and invariants, \( \llbracket T_M((\partial_{\lambda}(v_H(q_l)), \sigma, E)) \rrbracket \models^* \llbracket \langle l^0(l^0) / l^0(l^0) \rangle, \alpha' \rangle \).

Since \( l^0 \) is a non-final location \( \text{Corr}(l^0), (l^0) = (l^0) \) and \( \sigma' = \alpha' \upharpoonright \text{dom}(\sigma') \), the states \( (q, \sigma', E') \) and \( (l', \alpha') \) are corresponding.

2. In the case, when \( q_l \parallel q_2, p \) can make an action \( a \) and continue as \( p' \) in following cases (1-7):

(a) \( q_l \) can perform the action \( a \) and terminate \( (p' \equiv q_4) \)
(b) \( q_l \) can perform the action \( a \) and continue as \( q_4 \) \( (p' \equiv q_4 \parallel q_3) \)
(c) \( q_2 \) can perform the action \( a \) and terminate \( (p' \equiv q_3) \)
(d) \( q_2 \) can perform the action \( a \) and continue as \( q_3 \) \( (p' \equiv q_3 \parallel q_4) \)

Since the proofs for the cases (a), (c) and (b), (d) are similar, only the proofs for the first two cases have been given.

- Let's consider the case, when \( q_l \) can perform the action \( a \) and terminate \( (p' \equiv q_3) \). Then, according to the translation scheme, the corresponding extended automaton \( T_q(q_l) \) has the initial location \( l^2 \), final location \( l^0 \), and the edge \( (l^2, g^q, \tau_h, n^q, l^0) \), and \( \text{Inv}(l^0) \) and \( g^q \) are evaluated as true in the valuation \( a \), and \( \text{Inv}(l^0) = \text{true} \). Moreover, the extended automaton \( T_q(q_l) \) has the initial location \( l^2 \), \( \text{Inv}(l^0) \) is evaluated as true in the valuations \( a \) and \( a' \). Furthermore, \( a' \upharpoonright \text{dom}(\sigma) = \sigma' \).

Since the functions \( T_p \) and \( T_M \) do not change the locations, edges and invariants, \( \llbracket T_M((\partial_{\lambda}(v_H(q_l \parallel q_3)), \sigma, E)) \rrbracket \models^* \llbracket \langle l^0(l^0) / l^0(l^0) \rangle, \alpha' \rangle \).

Since the location \( l^0 \) is final \( \text{Corr}(l^0), (l^0) = (l^0) \) and \( \sigma' = \alpha' \upharpoonright \text{dom}(\sigma') \), the states \( (q_3, \sigma', E') \) and \( (l', \alpha') \) are corresponding.

- In case when \( q_l \) can perform the action \( a \) and continue as \( q_3 \) \( (p' \equiv q_3 \parallel q_3) \), the corresponding extended automaton \( T_q(q_l) \) has the initial location \( l^0 \), some non-final location \( l^1 \), and the edge \( (l^0, g^q, \tau_h, n^q, l^1) \), and \( \text{Inv}(l^1) \) and \( g^q \) are evaluated as true in the valuation \( a \), and \( \text{Inv}(l^0) = \text{true} \). Moreover, the extended automaton \( T_q(q_l) \) has the initial location \( l^0 \), \( \text{Inv}(l^0) \) is evaluated as true in the valuations \( a \) and \( a' \). Furthermore, \( a' \upharpoonright \text{dom}(\sigma) = \sigma' \).

Since the functions \( T_p \) and \( T_M \) do not change the locations, edges and invariants, \( \llbracket T_M((\partial_{\lambda}(v_H(q_l \parallel q_3)), \sigma, E)) \rrbracket \models^* \llbracket \langle l^0(l^0) / l^0(l^0) \rangle, \alpha' \rangle \).

Since the location \( l^0 \) is an initial location of the extended automaton \( T_q(q_l) \), i.e. \( \text{Corr}(l^0), (l^0) = (l^0) \), and \( \sigma' = \alpha' \upharpoonright \text{dom}(\sigma') \), the states \( (q_3, \sigma', E') \) and \( (l', \alpha') \) are corresponding.
Proof (Lemma 1.3). According to the rules [T-22] [T-25], the process term \( (\partial_A, (v_{le}(p)), \sigma, E) \) can perform a communication action \( \sigma a \) and then terminate if a process term \( p \) perform an action \( \sigma a \) and then terminate.

1. In case \( p \in Q \), the lemma holds trivially, since in \( \chi \) semantics there are no transition rules with \( \sigma a \) as specified defined for non-parallel processes.

2. In case if \( p \) is of the form \( q_1 \parallel q_2 \), the process term \( p \) can perform an action \( \sigma a \) and then terminate in the following cases (T-i8):

   (a) if \( q_1 \) can perform a send action and then terminate, and \( q_2 \) can perform a receive action and then terminate

   (b) if \( q_1 \) can perform a receive action and then terminate, and \( q_2 \) can perform a send action and terminate

Since the proofs for both cases are similar, only the proof of the first case has been given.

The lemma holds trivially for skip, assignment, deadlock, inconsistent process term, delay, repetition and sequential composition, since in \( \chi \) semantics there is no transition rules with \( \sigma a \) as specified.

The send action can be performed by the process terms send \( h ! ! e_n \), any delay operator \([q]\), and alternative composition \( q_1 \parallel q_2 \). Similarly, the receive action can be performed by the process terms receive \( h ? ? x_n \), any delay operator \([q]\), and alternative composition \( q_1 \parallel q_2 \).

Let’s consider the case, when \( q_1 \equiv h ! ! e_n \). According to the translation scheme an extended automaton \( T_q(h ! ! e_n) \) has the initial location \( l_0^h \), final location \( l_1^h \), and the edge \( l_0^h, g^h, h \), \( \{y_1 = e_1, \ldots, y_n = e_n\}, l_1^h \). The invariant Inv \((l_0^h)\) and a guard g^h, are evaluated as true in the valuation \( a \), and Inv \((l_1^h)\) = true. Since the functions \( T_p \) and \( T_M \) preserve all edges, the corresponding UPPAAL automaton \( A \), contains the edge \( l_0^h, g^h, h \), \( \{y_1 = e_1, \ldots, y_n = e_n\}, l_1^h \).

In case of any delay operator and alternative composition, it can be proved that the corresponding UPPAAL automaton \( A \), contains the initial location \( l_0^h \), final location \( l_1^h \), and the edge \( l_0^h, g^h, h \), \( \{y_1 = e_1, \ldots, y_n = e_n\}, l_1^h \). Furthermore, the invariant Inv \((l_0^h)\) and a guard g^h, are evaluated as true in the valuation \( a \), and Inv \((l_1^h)\) = true. The proofs for these cases are similar to the ones in the proof of the Lemma 1.3 cases \([a] [a] \), thus we do not adudge them here.

In the similar way it can be shown that if a process term \( q_2 \) can perform a receive action and terminate, that the corresponding UPPAAL automaton contains the initial location \( l_0^h \), final location \( l_1^h \), and the edge \( l_0^h, g^h, h \), \( \{x_1 = y_1, \ldots, x_n = y_n\}, l_1^h \).

Then, according to the UPPAAL semantics \( \mathcal{T}_M(\partial_A, (v_{le}(q_1 \parallel q_2)), \sigma, E) \) \( \parallel\vdash \langle I, \alpha \rangle \xrightarrow{h} \langle I', \alpha' \rangle \), where \( I = \{l_0^h, l_1^h, l_0^h, l_1^h\} \) and \( \alpha' = a \cdot (a^h(\alpha)) \).

Since the location \( l_0^h \) and \( l_1^h \) are final \( \langle Corr(l_0^h, l_1^h), \{l_0^h, l_1^h\}\rangle = \mathcal{Q} \), and \( \sigma' = \alpha' \cdot \text{dom}(\sigma') \), the states \( \langle q_2, \sigma', E' \rangle \) and \( \langle I', \alpha' \rangle \) are corresponding.

\[ \square \]
Proof (Lemma [1,3]).
According to the rules [1-22, 1-25], the process term \( \langle \partial_n, (v_{1k}(p)), \sigma, E \rangle \) can perform a communication action \( ca \) and then continue as \( p' \) if a process term \( p \) perform an action \( ca \) and continue as \( p' \):

1. In case if \( p \in Q \), the lemma holds trivially, since in \( \chi \) semantics there is no transition rules with \( ca \) as specified defined for non-parallel processes.

2. In case if \( p \) is of the form \( q_1 \parallel q_2 \), the process term \( p \) can perform an action \( ca \) and then continue as \( p' \) if one of the process terms can perform a send action and the other one can perform a receive action over the same channel, and at least one of the process terms does not terminate [1-28]. Since the proofs for most cases are similar, only the proofs for the following cases have been given:

(a) if \( q_1 \) can perform a send action and then continue as \( q'_1 \), and \( q_2 \) can perform a receive action and then terminate

(b) if \( q_1 \) can perform a send action and then continue as \( q'_1 \), and \( q_2 \) can perform a receive action and continue as \( q'_2 \)

According to the \( \chi \) semantics only the process terms any delay, sequential composition, repetition and alternative composition can perform a send (receive) action and then continue as \( q'_1 (q'_2) \). The process term receive, any delay and alternative composition can perform a send or receive action an then terminate.

In the same way as it was done in the proof of the Lemma [1,2] it can be shown that an extended automaton \( T_\alpha(q_1) \) has the initial location \( l_\alpha^0 \), some non-final location \( l_\alpha \), and the edge \( \langle l_\alpha^0, g_\alpha^0, h_\alpha, \{y_1 = e_1, \ldots, y_n = e_n\}, l_\alpha \rangle \). The invariant \( Inv(l_\alpha^0) \) and a guard \( g_\alpha^0 \), are evaluated as true in the valuation \( \alpha \), and \( Inv(l_\alpha^0) = \text{true} \).

Let us consider the first case. From the proof of the Lemma [1,3] it follows that \( T_\alpha(q_2) \) contains an edge \( \langle l_\alpha^0, g_\alpha^0, h_\alpha, \{x_1 = y_1, \ldots, x_n = y_n\}, l_\alpha^0 \rangle \), where \( l_\alpha^0 \) is a final location.

Then, according to the UPFAAL semantics \( [\mathcal{T}_M((\partial_n, (v_{1k}(q_1 \parallel q_2)), \sigma, E)) \models \langle \bar{l}, \bar{\alpha} \rangle \xrightarrow{h} \langle \bar{l}, \bar{\alpha}' \rangle \), \) where \( \bar{l} = \bar{l}^0_{\alpha_1} \parallel \bar{l}^0_{\alpha_2}, \bar{l}^0_{\alpha_1} \parallel l_\alpha^0 \rangle \), and \( \bar{\alpha}' = \alpha' \upharpoonright \sigma' \).

Since the location \( l_\alpha^0 \) is the initial location of the extended automaton \( T_\alpha(q_1) \), \( l_\alpha^0 \) is a final location \( \langle Corr((l_\alpha^0, l_\alpha^0), (l_\alpha^0, l_\alpha^0)), \{\bar{\sigma}, \bar{l}^0_{\bar{\alpha}}\} \rangle \) and \( \bar{\sigma}' = \sigma' \upharpoonright \text{dom}(\sigma') \), the states \( \langle q'_1, \sigma', E' \rangle \) and \( \langle \bar{l}, \bar{\alpha}' \rangle \) are corresponding.

In the second case the proof is similar, it can be shown that the automaton \( T_\alpha(q_2) \) contains the edge \( \langle l_\alpha^0, g_\alpha^0, h_\alpha, \{x_1 = y_1, \ldots, x_n = y_n\}, l_\alpha^0 \rangle \), \( l_\alpha^0 \) is a non-final location.

Then, according to the UPFAAL semantics \( [\mathcal{T}_M((\partial_n, (v_{1k}(q_1 \parallel q_2)), \sigma, E)) \models \langle \bar{l}, \bar{\alpha} \rangle \xrightarrow{h} \langle \bar{l}, \bar{\alpha}' \rangle \), \) where \( \bar{l} = \bar{l}^0_{\alpha_1} \parallel \bar{l}^0_{\alpha_2}, \bar{l}^0_{\alpha_1} \parallel l_\alpha^0 \rangle \), and \( \bar{\alpha}' = \alpha' \upharpoonright \sigma' \).

Since the location \( l_\alpha^0 \) and \( l_\alpha^0 \) are the initial locations of the extended automata \( T_\alpha(q_1) \) and \( T_\alpha(q_1) \) respectively \( \langle Corr((l_\alpha^0, l_\alpha^0), (l_\alpha^0, l_\alpha^0)), \{\bar{\sigma}, \bar{l}^0_{\bar{\alpha}}\} \rangle \) and \( \bar{\sigma}' = \sigma' \upharpoonright \text{dom}(\sigma') \), the states \( \langle q'_1, \sigma', E' \rangle \) and \( \langle \bar{l}, \bar{\alpha}' \rangle \) are corresponding.

\[\blacksquare\]

Proof (Lemma [1,7]).
According to the rules [T-23, T-26] the process term \( \langle \partial_{\alpha_0}(v_{1d}(p)), \sigma, E \rangle \) can make a time transition \( t \) and then continue as \( \langle \partial_{\alpha_0}(v_{1d}(p')), \sigma', E' \rangle \) if a process term \( p \) can make a delay transition \( t \) and continue as \( p' \) and no communication is possible during the delay.

According to the UPPAAL semantics, delays are performed in the locations, thus \( l = \bar{l} \). For simplicity below we write \( l \) instead of \( \bar{l} \).

1. Let us first consider the case if \( p \in Q \). The lemma holds trivially for skip, multi-assignment, send, receive, deadlock, and inconsistent process term, since there are no time transition rules defined for these process terms.

(a) The process term \( \Delta d \) can make a delay transition for \( t \) time units and continue as \( p' \equiv \Delta(d - t) \), if \( \rho \in \Omega(\sigma, t) \) and \( t \leq d \) [T-5]. Furthermore, \( \sigma' = \rho(t) \) and \( \forall x, \tau \in (\text{dom}(\sigma) \setminus \{ \text{time} \}) : \sigma'(x) = \sigma(x) \), and \( \sigma'(\text{time}) = \sigma(\text{time}) + t \).

According to the translation scheme, the extended automaton \( T_\sigma(\Delta d) \) has the initial location \( l_0 \), such that \( \text{Inv}(l_0) = \{ c \leq d \} \), where \( c \in C \), \( \alpha(c) = 0 \), \( T_1(l_0) = 0 \). Since \( l_0 \) is the only location in \( l \), \( t \leq d \), and the functions \( T_p \) and \( T_M \) do not change the locations, edges and invariants, \( [T_M(\langle \partial_{\alpha_0}(v_{1d}(\Delta d)), \sigma, E \rangle)] \models (l, \alpha) \rightarrow (l, \alpha') \).

Moreover, according to the UPPAAL semantics, during a delay value of all clock variables are changed synchronously and values of all non-clock variables remain the same: \( \forall c \in C : \alpha'(c) = \alpha(c) + t \). According to the translation scheme time is the only clock variable in \( \text{dom}(\sigma) \), thus \( \alpha' \upharpoonright (\text{dom}(\sigma) \setminus \{ \text{time} \}) = \alpha \).

Since we do not translate the process term \( \Delta(d - t) \), we cannot refer to its initial location. However from the Lemma [2] follows that after \( t \) time units, \( \alpha(c) = t \), \( T_M(\langle \partial_{\alpha_0}(v_{1d}(\Delta d)), \sigma, E \rangle) \) has the same time behaviour as \( T_M(\langle \partial_{\alpha_0}(v_{1d}(\Delta(d - t))), \sigma, E \rangle) \).

Moreover, since \( \alpha'(\text{time}) = \alpha(\text{time}) + t \) and \( \forall x : \alpha'(x) = \alpha(x) \), the states \( (p', \sigma', E') \) and \( (l, \alpha') \) are corresponding.

(b) The process term \( \lceil q \rceil \) can make an arbitrary time transition \( t \) and continue as \( p' \equiv \lceil q \rceil \), if \( \rho \in \Omega(\sigma, t) \). Moreover, \( \forall x, \tau \in (\text{dom}(\sigma) \setminus \{ \text{time} \}) : \sigma'(x) = \sigma(x) \), and \( \sigma'(\text{time}) = \sigma(\text{time}) + t \).

Applying \( T_\sigma(\lceil q \rceil) \) adds a new initial location \( l_0 \) into an extended automaton \( T_\sigma(q) \), such that \( \text{Inv}(l_0) = \text{true} \), and \( T_1(l_0) = o \).

Since the functions \( T_p \) and \( T_M \) do not change the locations, edges and invariants, \( [T_M(\langle \partial_{\alpha_0}(v_{1d}(\lceil q \rceil)), \sigma, E \rangle)] \models (l, \alpha) \rightarrow (l, \alpha') \).

Moreover, since \( \alpha'(\text{time}) = \alpha(\text{time}) + t \) and \( \forall x : \alpha'(x) = \alpha(x) \), the states \( (p', \sigma', E') \) and \( (l, \alpha') \) are corresponding.

(c) The process terms \( q_1; q_2 \) can make a delay transition and continue as \( p' \) if \( q_2 \) can make a delay transition and continue as \( q'_2 \) \( (p' \equiv q'_2; q_2) \) [T-13]. By induction, the corresponding extended automaton \( T_\sigma(q_1) \) has the initial location \( l_0 \), such that \( T_1(l_0) = o \), \( \alpha \upharpoonright \text{Inv}(l_0) \), \( \alpha' \upharpoonright \text{Inv}(l_0) \), and \( \alpha' \upharpoonright \text{dom}(\sigma') = \sigma' \). Furthermore, the extended automaton \( T_\sigma(q'_1) \) has the same initial location \( l_0 \).

Since the translation function \( T_\sigma(q_1; q_2) \) does not change the location \( l_0 \), and the functions \( T_p \) and \( T_M \) do not change the locations, edges and invariants, \( [T_M(\langle \partial_{\alpha_0}(v_{1d}(q_1; q_2)), \sigma, E \rangle)] \models (l, \alpha) \rightarrow (l, \alpha') \).

Moreover, since \( \alpha'(\text{time}) = \alpha(\text{time}) + t \) and \( \forall x : \alpha'(x) = \alpha(x) \), the states \( (p', \sigma', E') \) and \( (l, \alpha') \) are corresponding.

(d) The process terms \( q_1; q_2 \) can make a delay transition for \( t \) time units and continue as \( p' \) if both \( q_1 \) and \( q_2 \) can make a delay transition for \( t \) time units, \( p' \equiv q'_1; q'_2 \) [T-16]. By induction, the extended automata \( T_\sigma(q_1) \) and \( T_\sigma(q_2) \) have the initial locations \( l_0 \) and \( l_0 \), respectively. Moreover, \( \alpha \upharpoonright \text{Inv}(l_0), \alpha' \upharpoonright \text{Inv}(l_0), \alpha' \upharpoonright \text{Inv}(l_0), \alpha' \upharpoonright \text{Inv}(l_0) \).
Theorem 2. Let $p$ and $p'$ be closed process terms, $\sigma$, $\sigma'$, $\alpha$, $\alpha'$ be valuations, $E = \{J, H, \emptyset\}$ be an environment, and $\bar{l}$ be a location vector, such that the states $\langle p, \sigma, E \rangle$ and $\langle \bar{l}, \alpha \rangle$ are corresponding. Then:

**Lemma 2.1.** For any non-communication action $a$

$$ \llbracket T_M(M) \rrbracket \models \langle \bar{l}, \alpha \rangle \triangleq \langle \bar{l}', \alpha' \rangle \land \forall l \in \bar{l} : 1 \in L_d \Rightarrow $$

$$ \exists \sigma' : \{M, \mu \} \models \langle (\partial_{\alpha_{\mu}(v_H(p))) , \sigma, E \rangle \xrightarrow{\sigma, \alpha} \langle \sqrt{\sigma'}, E' \rangle, $$

and $\langle \bar{l}', \alpha' \rangle$ and $\langle \sqrt{\sigma'}, E' \rangle$ are corresponding states.

**Lemma 2.2.** For any non-communication action $a$

$$ \llbracket T_M(M) \rrbracket \models \langle \bar{l}, \alpha \rangle \triangleq \langle \bar{l}', \alpha' \rangle \land \exists l \in \bar{l} : l \notin L_d \Rightarrow $$

$$ \exists \sigma', p' : \{M, \mu \} \models \langle (\partial_{\alpha_{\mu}(v_H(p))) , \sigma, E \rangle \xrightarrow{\sigma, \alpha} \langle \partial_{\alpha_{\mu}(v_H(p'))} , \sigma', E' \rangle, $$

and $\langle \bar{l}', \alpha' \rangle$ and $\langle \partial_{\alpha_{\mu}(v_H(p'))} , \sigma', E' \rangle$ are corresponding states.

**Lemma 2.3.** For any communication action $ca$

$$ \llbracket T_M(M) \rrbracket \models \langle \bar{l}, \alpha \rangle \triangleq \langle \bar{l}', \alpha' \rangle \land \forall l \in \bar{l} : 1 \in L_d \land c_{S_1} = \alpha'(y_1) \land \ldots \land c_{S_m} = \alpha'(y_m) \Rightarrow $$

$$ \exists \sigma' : \{M, \mu \} \models \langle (\partial_{\alpha_{\mu}(v_H(p))) , \sigma, E \rangle \xrightarrow{\sigma, \alpha} \langle \partial_{\alpha_{\mu}(v_H(p'))} , \sigma', E' \rangle, $$

and $\langle \bar{l}', \alpha' \rangle$ and $\langle \partial_{\alpha_{\mu}(v_H(p'))} , \sigma', E' \rangle$ are corresponding states.

**Lemma 2.4.** For any communication action $ca$

$$ \llbracket T_M(M) \rrbracket \models \langle \bar{l}, \alpha \rangle \triangleq \langle \bar{l}', \alpha' \rangle \land \exists l \in \bar{l} : l \notin L_d \Rightarrow $$

$$ \exists \sigma', p' : \{M, \mu \} \models \langle (\partial_{\alpha_{\mu}(v_H(p))) , \sigma, E \rangle \xrightarrow{\sigma, \alpha} \langle \partial_{\alpha_{\mu}(v_H(p'))} , \sigma', E' \rangle. $$

2. In case if $p$ is of the form $q_i \parallel q_d$, the proof is similar to the proof for the alternative composition.

\[\square\]
∀ \mathcal{C}_2 = \alpha'(y_1) \land \ldots \land \mathcal{C}_n = \alpha'(y_n),

and \((\mathcal{I}', \alpha')\) and \((\partial_{\mathcal{A}_n}(v_{h}(p')), \sigma', E')\) are corresponding states.

**Lemma 2.5.** For any time transition
\[ [\mathcal{T}_M(M)] \vdash \langle \mathcal{T}, \alpha', E \rangle \land \exists I : I \notin L_t \Rightarrow \exists \sigma', p' : [\mathcal{M}] \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(p)), \sigma, E \rangle \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(p')), \sigma', E' \rangle,

and \((\mathcal{I}', \alpha')\) and \((\partial_{\mathcal{A}_n}(v_{h}(p')), \sigma', E')\) are corresponding states.

**Proof (Lemma 2.5).**

1. Let us first consider the case if \( p \in Q \). According to the translation scheme, the extended automata \( \mathcal{T}_q(\delta) \), \( \mathcal{T}_q(\lfloor \rfloor) \), \( \mathcal{T}_q(h! e_n) \), \( \mathcal{T}_q(h? x_n) \), \( \mathcal{T}_q(q) \) do not have an edge \( e \uparrow \mathcal{A}_n(x), \mathcal{A}_n(l) \), \( l, l \in L_t \), thus, the lemma holds trivially for these process terms.

   (a) According to the translation scheme the extended automaton \( \mathcal{T}_q(\text{skip}) \) has the initial location \( l_0 \), final location \( l_1 \), and the edge \( e = \langle l_0, \text{true}, \tau, l_1 \rangle \). Moreover, \( \text{Inv}(l_0) = \text{true}, \text{Inv}(l_1) = \text{true} \).

   According to the \( \chi \) semantics, the process term \( \text{skip} \) can make an internal action and terminate \([I-1]\). Then, from \([T-22][T-25]\),
\[ [\mathcal{M}] \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(\text{skip})), \sigma, E \rangle \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(\text{skip})), \sigma', E' \rangle. \]

Since \( \forall l \in I : l \in L_t \) and valuation \( \alpha' \uparrow \text{dom}(\sigma) = \sigma' \) (as \( \alpha = \alpha', \sigma = \sigma' \)), the states \((\mathcal{I}', \alpha')\) and \((p', \sigma', E')\) are corresponding.

(b) According to the translation scheme the extended automaton \( \mathcal{T}_q(x_n := e_n) \) has the initial location \( l_0 \), final location \( l_1 \), and the edge \( e = \langle l_0, \text{true}, \tau, l_1, \{x = e_1, \ldots, x = e_n\} \rangle \). Moreover, \( \text{Inv}(l_0) = \text{true}, \text{Inv}(l_1) = \text{true} \).

According to the \( \chi \) semantics, the process term \( x_n := e_n \) can make an internal action and terminate \([I-1]\). Then, from \([T-22][T-25]\),
\[ [\mathcal{M}] \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(x_n := e_n)), \sigma, E \rangle \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(x_n := e_n)), \sigma', E' \rangle. \]

From \texttt{UPPAAL} semantics, \( \alpha' = \alpha(a) \). According to \( \chi \) semantics, the multi-assignment process term changes the value of the variables \( x_1, \ldots, x_n \) to the values of the expressions \( e_1, \ldots, e_n \), respectively, leaving the rest of the variables unchanged. Hence, \( \alpha' \uparrow \text{dom}(\sigma) = \sigma' \).

Since \( \forall l \in I : l \in L_t \) and valuation \( \alpha' \uparrow \text{dom}(\sigma) = \sigma' \), the states \((\mathcal{I}', \alpha')\) and \((p', \sigma', E')\) are corresponding.

(c) According to the translation scheme the extended automaton \( \mathcal{T}_q(\Delta d) \) has the initial location \( l_0 \), final location \( l_1 \), and the edge \( e = \langle l_0, c \leftarrow d, \tau, l_1 \rangle \). Moreover, \( \text{Inv}(l_0) = c \leftarrow d, \text{Inv}(l_1) = \text{true} \).

According to the \( \chi \) semantics, the process term \( \Delta d \) can make an internal action and terminate, if \( \sigma(d) = \Delta(5,4) \). Since the states \((l_c, \alpha)\) and \((p, \sigma, E)\) are corresponding, \( \alpha(\text{time}) = \sigma(\text{time}) \). From the proof of the Lemma \([T-23]\), \( \Delta d = \alpha \). Then, from \([T-22][T-25]\),
\[ [\mathcal{M}] \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(\Delta d)), \sigma, E \rangle \vdash \langle \partial_{\mathcal{A}_n}(v_{h}(\Delta d)), \sigma', E' \rangle. \]

Since \( \forall l \in I : l \in L_t \) and valuation \( \alpha' \uparrow \text{dom}(\sigma) = \sigma' \) (as \( \alpha = \alpha', \sigma = \sigma' \)), the states \((\mathcal{I}', \alpha')\) and \((p', \sigma', E')\) are corresponding.

(d) According to the translation scheme the extended automaton \( \mathcal{T}_q([q]) \) has the initial location \( l_0 \), final location \( l_1 \), and the edge \( e = \langle l_0, g, \tau, l_1 \rangle \), such that \( \alpha \vdash \text{Inv}(l_0), \alpha' \vdash \text{Inv}(l_1), \alpha \vdash g \). Moreover, from the translation scheme and from Lemma \([P-1]\)
follows that the extended automaton $T_q(q)$ has the initial location $l_0^q$, final location $l_0^q$ (in $l_0$) and the edge $e^q = (l_0^q, g^q, \tau_h, a^q, l_0^q)$, such that $g^q = g, a^q = a$.

By induction we assume that the process term $q$ can make an action and terminate and the extended automaton $T_q(q)$ has the initial location $l_0^q$, final location $l_0^q$, and the edge $e = (l_0^q, g, \tau_h, a, l_0^q)$. Moreover, $a \models \text{Inv}(l_0), \alpha' \models \text{Inv}(l_0), a \models \text{dom}(\sigma) = \sigma$, $\alpha' \models \text{dom}(\sigma') = \sigma'$.

According to the $\chi$ semantics, if $q$ can make an $a$ action and terminate, than the process term $[q]$ can make an action $a$ and terminate (1.8). Then, from $[1.2]$, $[1.5]$

\[ \langle \partial_{\alpha}(v_{\chi}([q])), \sigma, E \rangle \xrightarrow{e^q, a^q} \langle \chi, \sigma', E' \rangle. \]

Since $\forall l \in I, l \in L_q$ and valuation $\alpha' \models \text{dom}(\sigma) = \sigma'$, the states $\langle l, \alpha' \rangle$ and $\langle l', \sigma', E' \rangle$ are corresponding.

(e) According to the translation scheme the extended automaton $T_q(q)$ has the initial location $l_0$, final location $l_0$ and the edge $e = (l_0, g, \tau_h, a, l_0)$, such that $a \models \text{Inv}(l_0), \alpha' \models \text{Inv}(l_0), a \models \text{dom}(\sigma) = \sigma$, $\alpha' \models \text{dom}(\sigma') = \sigma'$.

From the translation scheme $T_q(q_1)$ and $T_q(q_2)$ have the initial locations $l_0^q, l_0^q$, respectively, such that $a \models \text{Inv}(l_0^q)$ and $a \models \text{Inv}(l_0^q)$. Furthermore, $\exists l' = (l_0^q, l_0^q, \alpha') \in L_q$ or $l' = (l_0^q, l_0^q, \alpha') \in L_q$ and $a \models g$. Since the proofs for both cases are similar, we show the proof for the first case.

By induction we assume, that the process term $q_1$ can make an action $a$ and terminate, and $\alpha' \models \text{dom}(\sigma) = \sigma'$. Then, according to the $\chi$ semantics, the process term $q_1, [q_2]$ can make an action $a$ and terminate $\langle 1.1 \rangle$ and $\alpha' \models \text{dom}(\sigma) = \sigma'$. From $[1.2], [1.5]$

\[ \langle \partial_{\alpha}(v_{\chi}([q_1, q_2])), \sigma, E \rangle \xrightarrow{e^q, a^q} \langle \chi, \sigma', E' \rangle. \]

Since $\forall l \in I, l \in L_q$ and valuation $\alpha' \models \text{dom}(\sigma) = \sigma'$, the states $\langle l, \alpha' \rangle$ and $\langle l', \sigma', E' \rangle$ are corresponding.

2. Let us now consider the case if $p$ is of the form $q_1, q_2$. Then, according to the translation scheme $[1.2]$, $[1.5]$

\[ \langle \partial_{\alpha}(v_{\chi}([q_1, q_2])), \sigma, E \rangle \xrightarrow{e^q, a^q} \langle \chi, \sigma', E' \rangle. \]

Since $\forall l \in I, l \in L_q$ and valuation $\alpha' \models \text{dom}(\sigma) = \sigma'$, the states $\langle l, \alpha' \rangle$ and $\langle l', \sigma', E' \rangle$ are corresponding.

Proof (Lemma $[2.2]$).

1. Let us first consider the case, when $p \in Q$. According to the translation scheme, the extended automata $T_q(\text{skip}), T_q(\text{e}_n), T_q(h!!e_n), T_q(h??x_n), T_q(\partial), T_q(\perp), T_q(\Delta d)$, do not have an edge $e^q \xrightarrow{e^q, a^q} l_1, l_1 \notin L_q$, thus, the lemma hold trivially in these cases.

(a) According to the translation scheme the extended automaton $T_q([q])$ has the initial location $l_0$, a non-final location $l_0$, and an edge $e = (l_0, g, \tau_h, a, l_0)$, such that $\text{Inv}(l_0) \models a$, $\text{Inv}(l_0) \models \alpha', g \models a$.

Furthermore, from the translation scheme follows that the extended automaton $T_q(q)$ has the initial location $l_0$, a non-final location $l_0$ and the edge $e^q = (l_0, g, \tau_h, a, l_0)$. As it proved in $[1.5]$ applying $T_q([q])$ does not change action behavior. That means that the edge $e = (l_0, g, \tau_h, a, l_0)$ will be in $T_q([q])$.

By induction we assume that the process term $q$ can make an action $a$ and continue as $q'$. Moreover, $l_0$ is the initial location of $T_q(q')$, and $\alpha' \models \text{dom}(\sigma') = \sigma$. 

26
Let us consider the case, when \( q \) can make an action \( a \) and continue as \( q' \). From \( \text{T-25} \) we have \( \langle \partial_{\lambda_0}(v_{11}(q)), \sigma, E \rangle \models_{\epsilon} \langle \partial_{\lambda_0}(v_{11}(q')), \sigma', E' \rangle \).

Since \( l_0 \) is the initial location of \( T_q(q) \) and valuation \( \alpha' \upharpoonright \text{dom}(\sigma) = \sigma' \), the states \( \langle T, \alpha' \rangle \) and \( \langle p', \sigma', E' \rangle \) are corresponding.

(b) According to the translation scheme the extended automaton \( T_q(q') \) has the initial location \( l_0 \), a non-final location \( l_1 \), and an edge \( e = (l_0, g, r_1, a, l_1) \), such that \( a \models \text{Inv}(l_0), \alpha' \models \text{Inv}(l_1), a \models g \).

Furthermore, from the translation scheme follows that the extended automaton \( T_q(q) \) has the initial location \( l_0 \), a location \( l_m \) and the edge \( e' = (l_0, g, r_1, a, l_m) \).

i. Let us consider the case, when \( l_m \in L_e \). Applying \( T_q(q) \) redirects the edge \( e' \) to the initial location \( l_0 \) and adds initialization part \( \text{Init}^\delta \) to the assignment part of the edge. Since, according to the translation scheme, it is only clock resets that can be in the \( \text{Init}^\delta \), it will not make influence on \( \alpha' \upharpoonright \text{dom}(\sigma') \).

By induction we assume that the process term \( q \) can make an action and terminate. Moreover, \( \alpha' \upharpoonright \text{dom}(\sigma') = \sigma' \). Then, according to the \( \chi \) semantics, the process term \( \ast q \) can make an action \( a \) and continue as \( q \). From \( \text{T-22} \) we have \( \langle \partial_{\lambda_0}(v_{11}(q)), \sigma, E \rangle \models_{\epsilon} \langle \partial_{\lambda_0}(v_{11}(q)), \sigma', E' \rangle \).

Since \( l_0 \) is the initial location of \( T_q(q) \) and valuation \( \alpha' \upharpoonright \text{dom}(\sigma) = \sigma' \), the states \( \langle T, \alpha' \rangle \) and \( \langle p', \sigma', E' \rangle \) are corresponding.

ii. Let us consider the case, when \( l_m \notin L_e \). Applying \( T_q(q) \) will not change the edge \( e' \) (\( e' = e, l_m = l_0 \)).

By induction we assume that the process term \( q \) can make an action and continue as \( q' \). Moreover, \( \alpha' \upharpoonright \text{dom}(\sigma') = \sigma' \) and \( l_0 \) is the initial location of \( T_q(q) \). Then, according to the \( \chi \) semantics, the process term \( \ast q \) can make an action \( a \) and continue as \( q' \). From \( \text{T-22} \) we have \( \langle \partial_{\lambda_0}(v_{11}(q)), \sigma, E \rangle \models_{\epsilon} \langle \partial_{\lambda_0}(v_{11}(q')), \sigma, E' \rangle \).

Since \( l_0 \) is the initial location of \( T_q(q) \) and valuation \( \alpha' \upharpoonright \text{dom}(\sigma) = \sigma' \), the states \( \langle T, \alpha' \rangle \) and \( \langle p', \sigma', E' \rangle \) are corresponding.

(c) According to the translation scheme the extended automaton \( T_q(q_2) \) has the initial location \( l_0 \), a non-final location \( l_1 \), and an edge \( e = (l_0, g, r_1, a, l_1) \), where \( l_1 \in T_q(q_2), l_1 \notin L_e \) or \( l_0 \) is the initial location of \( T_q(q_2) \).

i. Let us consider the case, when \( l_1 \in T_q(q_2), l_1 \notin L_e \). By induction we assume that the process term \( q_2 \) can make an action \( a \) and continue as \( q'_2 \). Moreover, \( l_0 \) is the initial location of \( T_q(q'_2) \), and \( \alpha' \upharpoonright \text{dom}(\sigma') = \sigma' \).

Then, according to the \( \chi \) semantics, the process term \( q_2 \) can make an action \( a \) and continue as \( q'_2 \). From \( \text{T-22} \) we have \( \langle \partial_{\lambda_0}(v_{11}(q_2)), \sigma, E \rangle \models_{\epsilon} \langle \partial_{\lambda_0}(v_{11}(q'_2)), \sigma', E' \rangle \).

Since \( l_0 \) is the initial location of \( T_q(q'_2) \) and valuation \( \alpha' \upharpoonright \text{dom}(\sigma) = \sigma' \), the states \( \langle T, \alpha' \rangle \) and \( \langle p', \sigma', E' \rangle \) are corresponding.

ii. Let us consider the case, when \( l_1 \) is the initial location of \( T_q(q_2) \). By induction we assume that the process term \( q_2 \) can make an action \( a \) and terminate, and \( \alpha' \upharpoonright \text{dom}(\sigma') = \sigma' \).

Then, according to the \( \chi \) semantics, the process term \( q_2 \) can make an action \( a \) and continue as \( q'_2 \). From \( \text{T-22} \) we have \( \langle \partial_{\lambda_0}(v_{11}(q_2)), \sigma, E \rangle \models_{\epsilon} \langle \partial_{\lambda_0}(v_{11}(q'_2)), \sigma', E' \rangle \).

Since \( l_1 \) is the initial location of \( T_q(q_2) \) and valuation \( \alpha' \upharpoonright \text{dom}(\sigma) = \sigma' \), the states \( \langle T, \alpha' \rangle \) and \( \langle p', \sigma', E' \rangle \) are corresponding.
According to the translation scheme the extended automaton $T_1(q_1 \parallel q_2)$ has the initial location $l_0$, a non-final location $l_n$ and the edge $e = \langle l_0, g, l_n, l_n \rangle$, such that $a \models \text{Inv}(l_0)$, $a' \models \text{Inv}(l_n)$, $a \models g$.

From the translation scheme $T_1(q_1)$ and $T_1(q_2)$ have the initial locations $l_0^1, l_0^2$, respectively, such that $a \models \text{Inv}(l_0^1)$ and $a \models \text{Inv}(l_0^2)$. Furthermore, $\exists' = \langle l', g, r, a, l'' \rangle$, such that (1) $l = l_0^1, l'' = l_n^1, l'' \not\in L_2$ or (2) $l = l_0^2, l'' = l_n^2, l'' \not\in L_2$, and $a \models g$. Since the proofs for both cases are similar, we show the proof for the first case.

By induction we assume, that the process term $q_1$ can make an action $a$ and continue as $q_1'$. Moreover, $l_0^1$ is the initial location of $T_1(q_1')$ and $a' \models \text{dom}(\sigma) = \sigma'$. Then, according to the $\chi$ semantics, the process term $q_1 \parallel q_2$ can make an action $a$ and continue as $q_1 \parallel q_2 \models q_1'$. From \[T-22\] and $a' \models \text{dom}(\sigma) = \sigma'$, the states $\langle T, a' \rangle$ and $\langle p', \sigma', E \rangle$ are corresponding.

2. Let us consider the case, if $p$ is of the form $q_1 \parallel q_2$. Then, we can distinguish the following cases:

(a) $l_0^1, l_0^2 \in \bar{I}$: $l_0^1, l_0^2 \in l', e = \langle l_0^1, g, r, a, l_0^2 \rangle, l_0^1 \not\in L_2, l_0^2 \not\in L_2.$

(b) $l_0^1, l_0^2 \in \bar{I}$: $l_0^1, l_0^2 \in l', e = \langle l_0^2, g, r, a, l_0^1 \rangle, l_0^1 \not\in L_2, l_0^2 \not\in L_2.$

(c) $l_0^1, l_0^2 \in \bar{I}$: $l_0^1, l_0^2 \in l', e = \langle l_0^1, g, r, a, l_0^2 \rangle, l_0^1 \not\in L_2, l_0^2 \not\in L_2.$

(d) $l_0^1, l_0^2 \in \bar{I}$: $l_0^1, l_0^2 \in l', e = \langle l_0^2, g, r, a, l_0^1 \rangle, l_0^1 \not\in L_2, l_0^2 \not\in L_2.$

Since the proofs for first two cases and second two cases are similar, we only give proofs for the cases\[2a\] and \[2b\].

(a) By induction we assume that the process term $q_1$ can make an action $a$ and terminate. Moreover, $l_0^1$ is the initial location of $T_1(q_1)$, $l_0^2$ is the final location of $T_1(q_1)$, and $l_0^2$ is the initial location of $T_1(q_2)$. Furthermore, $a' \models \text{dom}(\sigma) = \sigma'$.

Then, according to the $\chi$ semantics, the process term $q_1 \parallel q_2$ can make an action $a$ and continue as $q_1 \parallel q_2 \models q_1'$. From \[T-22\] and $a' \models \text{dom}(\sigma) = \sigma'$, the states $\langle T, a' \rangle$ and $\langle p', \sigma', E \rangle$ are corresponding.

(b) By induction we assume that the process term $q_1$ can make an action $a$ and continue as $q_1'$. Moreover, $l_0^2$ is the initial location of $T_1(q_1)$, $l_0^1$ is the initial location of $T_1(q_2)$, and $l_0^2$ is the initial location of $T_1(q_1)$. Furthermore, $a' \models \text{dom}(\sigma) = \sigma'$.

Then, according to the $\chi$ semantics, the process term $q_1 \parallel q_2$ can make an action $a$ and continue as $q_1 \parallel q_2 \models q_1'$. From \[T-22\] and $a' \models \text{dom}(\sigma) = \sigma'$, the states $\langle T, a' \rangle$ and $\langle p', \sigma', E \rangle$ are corresponding.

\[\blacksquare\]

**Proof (Lemma 2.3).**
1. Let us first consider the case, when $p \in Q$. Then, according to the translation scheme, there is only one automaton in the UPPAAL model. Thus, the lemma hold trivially.

2. If $p$ is of the form $q_i \parallel q_3$, than either of the following holds.

   (a) The initial location of $T_q(q_i) \not\in \tilde{T}$ has an outgoing edge $\frac{g, h, a}{\tilde{L}_1}$, $\tilde{L}_1 \in L_1$. And the initial location of $T_q(q_3) \not\in \tilde{T}$ has an outgoing edge $\frac{g, h, a}{\tilde{L}_1}$, $\tilde{L}_1 \in L_1$.

   (b) The initial location of $T_q(q_i) \not\in \tilde{T}$ has an outgoing edge $\frac{g, h, a}{\tilde{L}_1}$, $\tilde{L}_1 \in L_1$. And the initial location of $T_q(q_3) \not\in \tilde{T}$ has an outgoing edge $\frac{g, h, a}{\tilde{L}_1}$, $\tilde{L}_1 \in L_1$.

Since the proofs for both cases are similar, only the proof for the first case is given.

According to the translation scheme the extended automata $T_q(\text{skip})$, $T_q(x_a := e_u)$, $T_q(\delta)$, $T_q(\bot)$, $T_q(\Delta d)$, $T_q(\sigma q)$, $T_q(q)$ do not have an edge $\frac{g, h, a}{l}$, or $\frac{g, h, a}{l}$, or $l \in L_1$, thus, the lemma holds trivially for these process terms.

According to the translation scheme, the extended automaton $T_q(q_i)$ can have an edge $\frac{g, h, a}{l}$ if $q_i$ is either $h || e_u$, or $[q]$, or $q || r$. And the extended automaton $T_q(q_3)$ can have an edge $\frac{g, h, a}{l}$ if $q_3$ is either $h || e_u$, or $[q]$, or $q || r$.

Let us first consider the case if $q_i \equiv h || e_u$. Then, according to the $\chi$ semantics (1,3),

$$\not{\parallel} \langle q_i, \sigma, E \rangle \not{\parallel} \langle q_i \parallel q_3, \sigma, E \rangle \frac{\text{c,ira}([h || e_u], \sigma', \partial)}{\langle \forall, \sigma', E' \rangle}$$

In the cases when $q_i \equiv [q]$ or $q_i \equiv [q] || r$, it can be proved that

$$\not{\parallel} \langle q_i, \sigma, E \rangle \not{\parallel} \langle q_i, \sigma, E \rangle \frac{\text{c,ira}([h || e_u], \sigma', \partial)}{\langle \forall, \sigma', E' \rangle}$$

and $\sigma = \sigma'$, in the way similar to the proof of the Lemma 2.2 (cases 1a and 1d respectively). Thus we do not adue these proofs here.

Similar, if $q_3 \equiv h ? \equiv e_u | [q] | q || r$, $\not{\parallel} \langle q_2, \sigma, E \rangle \not{\parallel} \langle q_2, \sigma, E \rangle \frac{\text{c,ira}([h || e_u], \sigma', \partial)}{\langle \forall, \sigma', E' \rangle}$ and $\sigma'(x) = e_u$.

Then, according to the $\chi$ semantics (1,3), the process term $q_1 \parallel q_3$ can make a communication action and terminate: $\not{\parallel} \langle q_1 \parallel q_2, \sigma, E \rangle \not{\parallel} \langle q_1 \parallel q_3, \sigma, E \rangle \frac{\text{c,ira}([h || e_u], \sigma', \partial)}{\langle \forall, \sigma', E' \rangle}$ and $\sigma'(x_a) = e_u$. $\forall x \notin x_a : \sigma'(x) = \sigma(x)$.

From (1,2,3) if $\not{\parallel} \langle q_1 \parallel q_2, \sigma, E \rangle \not{\parallel} \langle q_1 \parallel q_3, \sigma, E \rangle \frac{\text{c,ira}([h || e_u], \sigma', \partial)}{\langle \forall, \sigma', E' \rangle}$, then

$$\not{\parallel} \langle q_1 \parallel q_2, \sigma, E \rangle \not{\parallel} \langle q_1 \parallel q_3, \sigma, E \rangle \frac{\text{c,ira}([h || e_u], \sigma', \partial)}{\langle \forall, \sigma', E' \rangle}.$$ Note that, since the actions isa, ira in $A_{isa}$, they cannot be executed.

According to the UPPAAL semantics, $\forall x_i \in x_a : \alpha'(x_i) = x_i$ and from the translation scheme $\forall x \notin x_a, x \in \text{dom}(\alpha) : \alpha'(x) = \alpha(x)$. Since $\forall l \in \tilde{T} : l \in L_1$ and valuation $\alpha' \mid \text{dom}(\sigma) = \sigma'$, the states $\tilde{T}, \alpha'$ and $\langle \forall, \sigma', E' \rangle$ are corresponding.

Proof (Lemma 2.2).
2. If \( p \) is of the form \( \llbracket q_1, q_2 \rrbracket \), than a synchronization action is possible if:

   \( \text{a)} \) The initial location of \( T_{\theta}(q_1) \) \( \in I \) has an outgoing edge \( \overset{g,h,a}{\longrightarrow} p_0 \). And the initial location of \( T_{\theta}(q_2) \) \( \in I \) has an outgoing edge \( \overset{g,h,a}{\longrightarrow} p_0 \).

   \( \text{b)} \) The initial location of \( T_{\theta}(q_1) \) \( \in I \) has an outgoing edge \( \overset{g,h,a}{\longrightarrow} p_0 \). And the initial location of \( T_{\theta}(q_2) \) \( \in I \) has an outgoing edge \( \overset{g,h,a}{\longrightarrow} p_0 \).

Since the proofs for both cases are similar, only the proof for the first case is given. Here we consider three subcases:

   \( \text{a)} \) \( p_0 \in L_f, p_0 \notin L_f \)

   \( \text{b)} \) \( p_0 \notin L_f, p_0 \in L_f \)

   \( \text{c)} \) \( p_0 \notin L_f, p_0 \notin L_f \)

The proofs for the subcases are similar, so we only prove the first one.

From the Part 3 [REF] of this proof follows that if the extended automaton \( T_{\theta}(q_1) \) has an edge \( e^{h} = (p_0, g, h, a, p_0) \), then \( \llbracket (q_1, \sigma, E) \rrbracket \models \llbracket (q_1, \sigma, E) \rrbracket \overset{\sigma, e^{h} \llbracket \{x_1 \} \rrbracket, \sigma'} \longrightarrow \llbracket (q_1', \sigma', E) \rrbracket \) and \( \sigma = \sigma' \).

According to the translation scheme, the extended automaton \( T_{\theta}(q_2) \) can have an edge \( e^{h} = (p_0, g, h, a, p_0) \) if \( q_2 \) is either \( q_1 \) or \( q_1 \). In the similar way as it was done in the proof [10], it can be shown that \( \llbracket (q_2, \sigma, E) \rrbracket \models \llbracket (q_2, \sigma, E) \rrbracket \overset{\sigma, e^{h} \llbracket \{x_1 \} \rrbracket, \sigma'} \longrightarrow \llbracket (q_2', \sigma', E) \rrbracket \) and \( \sigma' \). Note, that \( \ell_f \) is the initial location of \( T_{\theta}(q_2) \).

Then, according to the \( \chi \) semantics [11,8], the process term \( \llbracket q_1, q_2 \rrbracket \) can make a communication action and continue as \( q_2' \): \( \llbracket (q_1, \llbracket q_2, \sigma, E \rrbracket) \rrbracket \models \llbracket (q_1, \llbracket q_2, \sigma, E \rrbracket) \rrbracket \overset{\sigma, e^{h} \llbracket \{x_1 \} \rrbracket, \sigma'} \longrightarrow \llbracket (q_2', \sigma', E) \rrbracket \) and \( \sigma' \).

From the rule [11,22,11,25] if \( \llbracket (q_1, \llbracket q_2, \sigma, E \rrbracket) \rrbracket \models \llbracket (q_1, \llbracket q_2, \sigma, E \rrbracket) \rrbracket \overset{\sigma, e^{h} \llbracket \{x_1 \} \rrbracket, \sigma'} \longrightarrow \llbracket (q_2', \sigma', E) \rrbracket \), then \( \llbracket (q_1, \llbracket q_2, \sigma, E \rrbracket) \rrbracket \models \llbracket (q_1, \llbracket q_2, \sigma, E \rrbracket) \rrbracket \overset{\sigma, e^{h} \llbracket \{x_1 \} \rrbracket, \sigma'} \longrightarrow \llbracket (q_2', \sigma', E) \rrbracket \). Note, that since the actions is, \( a, q_i \in \mathcal{A}_{sa} \), they cannot be executed.

According to the UPPAAL semantics, \( \forall x \in x_0, \alpha'(x) = e_i \), and from the translation scheme \( \forall x \in x_0, x \in \text{dom}(\sigma) : \alpha'(x) = \alpha(x) \). Since \( \forall l \in L, l \neq l_f \) and valuation \( \alpha \llbracket \text{dom}(\sigma) = \alpha \), the states \( \{l', \alpha'\} \) and \( \llbracket q_2', \sigma', E \rrbracket \) are corresponding.

\( \Box \)

Proof (Lemma 2.3).

According to the UPPAAL semantics, delays are performed in the locations, thus \( I = \hat{I} \). For simplicity below we write \( I \) instead of \( \hat{I} \).

1. Let us first consider the case, when \( p \in Q \). According to the translation scheme, in the extended automata \( T_{\theta}(\text{skip}) \), \( T_{\theta}(x_i := e_i) \), \( T_{\theta}(h!! e_i) \), \( T_{\theta}(h?? x_i) \), \( T_{\theta}(\delta) \), \( T_{\theta}(\bot) \), the initial locations are urgent, thus, the lemma hold trivially in these cases.
(a) According to the translation scheme the extended automaton $T_\Delta(\Delta \ell)$ has the initial location $l_0$, such that $Inv(l_0) = \epsilon \leq \Delta \ell$, $T_1(l_0) = o$. According to the UpPAAL semantics, $T_M((\partial_{\Delta}(v_{t\mathcal{H}}(\Delta \ell)), \sigma, E))$ can make a time transition for $t$ time units, $t \leq \Delta \ell$. Note, that the communication is not possible. Moreover, $\alpha' \upharpoonright (dom(\sigma) \setminus \{\text{time}\}) = \alpha$, and $\alpha' (\text{time}) = \alpha (\text{time}) + t$.

According to the $\chi$ semantics, the process term $\Delta \ell$ can make a time transition for $t$ time units, $t \leq \Delta \ell$ and continue as $\Delta(\Delta - t)$ [5.3]. Moreover, $\sigma' \upharpoonright (dom(\sigma') \setminus \{\text{time}\}) = \sigma$, and $\sigma'(\text{time}) = \sigma(\text{time}) + t$.

From $T-23$ $T-26$ $[\langle \partial_{\Delta}(v_{t\mathcal{H}}(\Delta \ell)), \sigma', E \rangle \models (\partial_{\Delta}(v_{t\mathcal{H}}(\Delta \ell)), \sigma, E) \xrightarrow{t, E} (\partial_{\Delta}(v_{t\mathcal{H}}(\Delta(\Delta - t))), \sigma', E)]$.

Since we do not translate the process term $\Delta(\Delta - t)$, we cannot refer to its initial location. However, from the Lemma [2] follows that after $t$ time units, $\alpha(c) = t$, $T_M((\partial_{\Delta}(v_{t\mathcal{H}}(\Delta \ell)), \sigma, E))$ has the same time behavior as $T_M((\partial_{\Delta}(v_{t\mathcal{H}}(\Delta(\Delta - t))), \sigma, E))$.

Since valuation $\alpha' \upharpoonright \text{dom}(\sigma) = \sigma'$, the states $\langle l, \alpha' \rangle$ and $\langle \ell', \sigma', E \rangle$ are corresponding.

(b) According to the translation scheme the extended automaton $T_\ell([q])$ has the initial location $l_0$, such that $Inv(l_0) = \text{true}$, $T_1(l_0) = o$. According to the UpPAAL semantics, $T_M((\partial_{\Delta}(v_{t\mathcal{H}}([q])), \sigma, E))$ can make a time transition for $t$ time units. Note, that the communication is not possible. Moreover, $\alpha' \upharpoonright (\text{dom}(\sigma) \setminus \{\text{time}\}) = \alpha$, and $\alpha' (\text{time}) = \alpha (\text{time}) + t$.

According to the $\chi$ semantics, the process term $[q]$ can make a time transition for $t$ time units and continue as $[q]$ [5.3]. Note, that $[q]$ disregards time behavior of $q$.

Moreover, $\sigma' \upharpoonright (\text{dom}(\sigma') \setminus \{\text{time}\}) = \sigma$, and $\sigma'(\text{time}) = \sigma(\text{time}) + t$.

From $T-23$ $T-26$ $[\langle \partial_{\Delta}(v_{t\mathcal{H}}([q])), \sigma', E \rangle \models (\partial_{\Delta}(v_{t\mathcal{H}}([q])), \sigma, E) \xrightarrow{t, E} (\partial_{\Delta}(v_{t\mathcal{H}}([q])), \sigma', E)]$.

Since $l_0$ is the initial location of $T_\ell([q])$ and valuation $\alpha' \upharpoonright \text{dom}(\sigma) = \sigma'$, the states $\langle l, \alpha' \rangle$ and $\langle \ell', \sigma', E \rangle$ are corresponding.

(c) According to the translation scheme the extended automaton $T_\ell([q] + q)$ has the initial location $l_0$, $\alpha \models \text{Inv}(l_0)$. $T_1(l_0) = o$. From the translation scheme follows that the extended automaton $T_\ell([q])$ has the initial location $l^1_0 = \text{Inv}(l^1_0) = \text{Inv}(l_0)$. $T_1(l^1_0) = o$.

That means that $\models T_M([q]) \models (l, \alpha) \xrightarrow{t} (l, \alpha')$, and $\alpha' \upharpoonright (\text{dom}(\sigma) \cup \{\text{time}\}) = \alpha, \alpha'(\text{time}) = \alpha(\text{time}) + t$.

By induction we assume that the process term $q$ can make a time transition for $t$ time units and continue as $q$. Moreover, $\alpha' \upharpoonright \text{dom}(\sigma') = \sigma'$ and $l_0$ is the initial location of $T_\ell([q])$. Then, according to the $\chi$ semantics, the process term $+q$ can make a time transition for $t$ time units and continue as $[q]$ [REF], and $\alpha' \upharpoonright \text{dom}(\sigma') = \sigma'$.

From $T-23$ $T-26$ $[\models T_M((\partial_{\Delta}(v_{t\mathcal{H}}([q] + q)), \sigma, E) \xrightarrow{t, E} (\partial_{\Delta}(v_{t\mathcal{H}}([q] + q)), \sigma', E)]$.

Note, that the communication is not possible.

Since $l_0$ is the initial location of $T_\ell([q] + q)$ and valuation $\alpha' \upharpoonright \text{dom}(\sigma) = \sigma'$, the states $\langle l, \alpha' \rangle$ and $\langle \ell', \sigma', E \rangle$ are corresponding.

(d) According to the translation scheme the extended automaton $T_\ell(q:\ q)$ has the initial location $l_0$, $\alpha \models \text{Inv}(l_0)$.

From the translation scheme follows that the extended automaton $T_\ell(q:\ q)$ has the initial location $l^1_0 = l_0$, $\text{Inv}(l^1_0) = \text{Inv}(l_0)$. $T_1(l^1_0) = o$. That means that $\models T_M(q:\ q) \models (l, \alpha) \xrightarrow{t} (l, \alpha')$, and $\alpha' \upharpoonright (\text{dom}(\sigma) \cup \{\text{time}\}) = \alpha, \alpha'(\text{time}) = \alpha(\text{time}) + t$.

Note, that the communication is not possible.

By induction we assume that the process term $q$ can make a time transition for $t$ time units and continue as $q$. Moreover, $l_0$ is the initial location of $T_\ell(q:\ q)$, and $\alpha' \upharpoonright \text{dom}(\sigma') = \sigma$. 

31 Correctness of the translation scheme
Then, according to the \( \chi \) semantics, the process term \( q_1; q_2 \) can make a time transition for \( t \) time units and continue as \( q' \) [REF] and \( \alpha' \upharpoonright \sigma' = \sigma' \). From \[T-23\]

\[
\begin{align*}
& T-23 \quad \exists \quad (\partial_{A_0}(v_{1h}(q_1; q_2)), \sigma, E) \implies (\partial_{A_0}(v_{1h}(q_1)), \sigma, E) \xrightarrow{\sigma_{A_0}} (\partial_{A_0}(v_{1h}(q_2)), \sigma', E).
\end{align*}
\]

Since \( l_0 \) is the initial location of \( T_q(q') \) and valuation \( \alpha' \upharpoonright \dom(\sigma) = \sigma' \), the states \( \langle \overline{l}, \alpha' \rangle \) and \( \langle \overline{p}', \sigma', E' \rangle \) are corresponding.

(c) According to the translation scheme the extended automaton \( T_q(q_1; q_2) \) has the initial location \( l_0 \), and \( \alpha \models \Inv(l_0) \).

From the translation scheme \( T_q(q_1) \) and \( T_q(q_2) \) have the initial locations \( l_0^{(1)} \), \( l_0^{(2)} \), respectively, such that \( \alpha \models \Inv(l_0^{(1)}) \) and \( \alpha \models \Inv(l_0^{(2)}) \).

That means that \[T-27\]

\[
\begin{align*}
& T-27 \quad \exists \quad (\partial_{A_0}(v_{1h}(q_1)), \sigma, E) \implies (\partial_{A_0}(v_{1h}(q_1)), \sigma, E) \xrightarrow{\sigma_{A_0}} (\partial_{A_0}(v_{1h}(q_2)), \sigma', E).
\end{align*}
\]

Since \( l_0 \) is the initial location of \( T_q(q') \) and valuation \( \alpha' \upharpoonright \dom(\sigma) = \sigma' \), the states \( \langle \overline{l}, \alpha' \rangle \) and \( \langle \overline{p}', \sigma', E' \rangle \) are corresponding.

2. The proof for the case, when \( p :: = \overline{q_1} \parallel \overline{q_2} \) is similar to the proof for the alternative composition \[Te\].

\[\Box\]

5.4 Theorem 3

Theorem 3. Let \( p \) be closed process term, \( \sigma \) and \( \alpha \) be valuations, \( E = (J, H, \emptyset) \) be an environment, and \( \overline{l} \) be a location vector, such that the states \( \langle \overline{l}, \sigma, E \rangle \) and \( \langle \overline{l}, \alpha \rangle \) are corresponding. Then:

\[ \forall l \in \overline{l} : \alpha \models \Inv(l) \Leftrightarrow (\partial_{A_0}(v_{1h}(p)), \sigma, E) \not\sim. \]

Let us first prove that \( \forall l \in \overline{l} : \alpha \models \Inv(l) \Rightarrow (\partial_{A_0}(v_{1h}(p)), \sigma, E) \not\sim. \)

Proof (Lemma 1).

1. Let us consider the case when \( p \in Q \). Then, according to the translation \( \overline{l} = \langle l_0 \rangle \), where \( l_0 \) is the initial location of the automaton \( T_q(p) \). By induction we assume that \( \langle p, \sigma, E \rangle \not\sim. \)

Then, according to the rules \[T-24\] and \[T-27\] the process term \( \langle \partial_{A_0}(v_{1h}(p)), \sigma, E \rangle \) is consistent.

Furthermore, according to the translation scheme, the invariants of the initial locations of the automata \( T_q(\text{skip}) \), \( T_q(\text{x} := e) \), \( T_q(h \parallel e) \), \( T_q(h ? \text{x}) \), \( T_q(\theta) \), and \( T_q(\text{g}) \) are always true. The proofs for these cases are trivial, since according to the rules \[T-2\], \[T-5\], \[T-6\], \[T-22\] and \[T-26\] these process terms are always consistent.

(a) According to the translation scheme, the invariant of the initial location \( l_0 \) of the automaton \( T_q(\bot) \) is always false. The proof is trivial, since the inconsistent process term \( \bot \) is not consistent in any valuation.
(b) The invariant of the initial location of the automaton $T_q(\Delta d)$ is of the form $c \leq d$, where $c$ is a clock variable and $d$ is an expression. According to the UPPAAL semantics, the invariant must stay satisfied during delay, thus $t \in [0, d]$.

According to the $\chi$ semantics the process term $\langle \Delta d, \sigma, E \rangle$ can delay for $t$ time units, $0 \leq t \leq \sigma(d)$, and is always consistent [5.6].

(c) According to the translation scheme, the initial location $I_0$ of the automaton $T_q(\ast q)$ is also the initial location of the automaton $T_q(q)$. By induction we assume that the process term $\langle q, \sigma, E \rangle$ is consistent. Then, according to the rule [5.3] $\langle \ast q, \sigma, E \rangle \not\leq$. [5.6]

(d) According to the translation scheme, the initial location $I_0$ of the automaton $T_q(q_2)$ is also the initial location of the automaton $T_q(q)$. By induction we assume that the process term $\langle q_1, \sigma, E \rangle$ is consistent. Then, according to the rule $T-24$ $\langle q_1, q_2, \sigma, E \rangle \not\leq$. [5.6]

(e) According to the translation scheme, the invariant of the initial location $I_0$ of the automaton $T_q(q_1 \parallel q_2)$ is $\text{Inv}(I_0) = \text{Inv}^b(I_0^c) \land \text{Inv}^b(I_0^d)$. By induction we assume that the process terms $\langle q_1, \sigma, E \rangle$ and $\langle q_2, \sigma, E \rangle$ are consistent. Then, according to the rule $T-27$ $\langle q_1, q_2, \sigma, E \rangle \not\leq$. [5.6]

2. Let us now consider the case when $p$ is of the form $q_1 \parallel q_2$. Then, according to the translation $I_0^c, I_0^d \in I$. By induction we assume that the process terms $\langle q_1, \sigma, E \rangle$ and $\langle q_2, \sigma, E \rangle$ are consistent. Then, according to the rule $T-24$ and $T-27$ the process term $\langle \partial_{\lambda_0}(v_{ih}(p)), \sigma, E \rangle$ is consistent.

Now we prove that $\langle \partial_{\lambda_0}(v_{ih}(p)), \sigma, E \rangle \not\leq \Rightarrow \forall I : a \models \text{Inv}(I)$.

**Proof (Lemma ).**

1. According to the rules $T-24$ and $T-27$, the process term $\langle \partial_{\lambda_0}(v_{ih}(p)), \sigma, E \rangle$ is consistent only if $\langle p, \sigma, E \rangle \not\leq$. By induction we assume that $\forall I' : a \models \text{Inv}(I')$. Since from the translation scheme $I = I'$, the proof is trivial.

2. Let us consider the case when $p \in Q$. According to the rules $T-2$, $T-5$, $T-6$, $T-7$, and $T-10$, the process terms $\text{skip}$, $x_a := e_a$, $h \ll e_a$, $h \ll ? x_a$, $\delta$, and $[\ast q]$ are always consistent. The proofs for these cases are trivial, since according to the translation scheme, the invariants of the initial locations of the automata $T_q(\text{skip})$, $T_q(x_a := e_a)$, $T_q(h \ll e_a)$, $T_q(h \ll ? x_a)$, $T_q(\delta)$, and $T_q(\ast q)$ are always true.

   (a) According to $\chi$ semantics, the inconsistent process term $\bot$ is not consistent in any valuation. The proof is trivial, since from the translation scheme the invariant of the initial location $I_0$ of the automaton $T_q(\bot)$ is false.

   (b) According to the $\chi$ semantics the process term $\langle \Delta d, \sigma, E \rangle$ is always consistent [5.6], and $0 \leq t \leq \sigma(d)$.

The invariant of the initial location of the automaton $T_q(\Delta d)$ is of the form $c \leq d$, where $c$ is a clock variable and $d$ is an expression. Since $t \in [0, d]$, the invariant is always satisfied.

(c) From the rule [5.3] the process term $\langle \ast q, \sigma, E \rangle$ is consistent only if $\langle q, \sigma, E \rangle \not\leq$. By induction we assume that $a \models \text{Inv}(I_0)$. The proof is trivial, since from the translation scheme $I_0 = I_0^c$. [5.6]
(d) From the rule [14], the process term \(q_1; q_2, \sigma, E\) is consistent only if \(q_1, \sigma, E \models \varphi\). By induction we assume that \(\alpha \models \text{Inv}^\varphi (l_0^0)\). The proof is trivial, since from the translation scheme \(I_0 = l_0^0\).

(e) From the rule [17] the process term \(q_1 \parallel q_2, \sigma, E\) is consistent only if \(q_1, \sigma, E \not\models \varphi\) and \(q_2, \sigma, E \not\models \varphi\). By induction we assume that \(\alpha \models \text{Inv}^\varphi (l_0^0)\). The proof is trivial, since from the translation scheme \(\text{Inv}(l_0) = \text{Inv}^\varphi (l_0^0) \land \text{Inv}^\varphi (l_1^0)\).

3. Let us consider the case when the process term \(p\) is of the form \(q_1 \parallel q_2\). From the rule [21] the process term \(q_1 \parallel q_2, \sigma, E\) is consistent only if \(q_1, \sigma, E \not\models \varphi\) and \(q_2, \sigma, E \not\models \varphi\). By induction we assume that \(\alpha \models \text{Inv}^\varphi (l_0^0) \land \text{Inv}^\varphi (l_1^0)\). The proof is trivial, since from the translation scheme \(l_0^0, l_1^0 \in I\).

6 Informal translation

For several \(\chi\) language constructs, like guard operator or guarded repetition, we have not found a general translation. However, in some special cases the translation is possible and is implemented. In this section we present these cases.

6.1 Guarded skip and multi-assignment

The guarded skip and multi-assignment in \(\chi\) behave as follows. If guard is true, the action transition is performed without a delay. If guard is false, the process term delays till it becomes true.

Since there is no notion of urgent actions in timed automata, the translation of the guarded skip and multi-assignment is not possible. However, if the guard expression does not involve the variable time, time \(\not\in b\), the possible solution is to introduce a new automaton, which has only one location \(l_0\), \(\text{Inv}(l_0) = \text{true}\), \(T_1 = 0\) and the only edge \(e = (l_0, \text{true}, \text{dummy}\top, a, l_0)\). Then, the extended automaton \(T_q(b \rightarrow \text{skip})\) can be defined as follows:

\[
T_q(b \rightarrow \text{skip}) = \langle \{l_0, l_1\}, l_0, \{l_0, b, \text{dummy}\top, \tau_1, l_1\} \rangle,
\]

where \(\text{Inv}(l_0) = \text{true}\), \(\text{Inv}(l_1) = \text{true}\), \(T_1(l_0) = 0\), \(T_1(l_1) = 0\).

Similarly, the extended automaton \(T_q(b \rightarrow x_n := e_n)\) is defined as follows:

\[
T_q(x_n := e_n) = \langle \{l_0, l_1\}, l_0 \rangle,
\]

where \(\text{Inv}(l_0) = \text{true}\), \(\text{Inv}(l_1) = \text{true}\), \(T_1(l_0) = 0\), \(T_1(l_1) = 0\).
6.2 Guarded Send and Receive

According to the $\chi$ semantics the guarded send and receive process terms behave as follows. If the guards in false, the process term delays till the guard becomes true. Then, the send or receive action is performed without delay. If the guard is true and the communication action is not possible, the process is deadlocked. Remember, that the action encapsulation operator $\partial a$, disallows internal send and receive transitions. Although we did not find the proper translation for these process terms, we were able to define the translation of the guarded delayable send and receive, i.e. $b \rightarrow [h !! e_1]$ and $b \rightarrow [h ?? x_1]$ under the condition $\text{time} \nleq b$.

Let $\mathcal{T}_0(p) = \langle L^p, p^0, E^p, V^{hp}, C^p, \text{Init}^p, \text{Inv}^p, T^p_L, l^0_p \rangle$. Then,

$\mathcal{T}_0(b \rightarrow p) = \langle (L^p \cup \{l_0, l_1\}) \setminus \{l^0_p\}, l_0, E, V^p, V^{hp}, C^p, \text{Init}^p, \text{Inv}^p, T^p_L, l_1 \rangle$, where

- $l_0$ and $l_1$ are new initial and final locations.
- $E = \sigma(\langle L^p, \{l_0, l_1\}, l_0 \rangle) \cup \{\langle l_0, b, \tau_h, \tau_s, l_1 \rangle, \langle l_0, \neg b, \tau_h, \tau_s, l_1 \rangle\}$.
- Inv $\mathcal{T}_0(L^p \setminus \{l^0_p\}) = \text{Inv}^p$ $\mathcal{T}_0(L^p \setminus \{l^0_p\})$ and Inv($l_0$) = true, Inv($l_1$) = true.
- $T^p_L \mathcal{T}_0(L^p \setminus \{l^0_p\}) = T^p_L \mathcal{T}_0(L^p \setminus \{l^0_p\})$ and $T^p_L(l_0) = u$, $T^p_L(l_1) = o$.

Informal translation
6.4 Some notes

We were not able to translate a guarded delay, $b \rightarrow \Delta d$. The main problem here is the time behaviour. For instance, if $b$ is false, the process term $b \rightarrow \Delta d$ delays till $b$ becomes true, disregarding the time behaviour of $\Delta d$. That means, that if first the guard $b$ is true, the process term $b \rightarrow \Delta d$ starts delaying. If after $t$ time units, $t < d$, the guard becomes false, the process term $b \rightarrow \Delta(d - t)$ delays till $b$ becomes true disregarding the time behaviour of $\Delta(d - t)$. Then, if $b$ is true, it continues to delay as $\Delta(d - t)$.

Nevertheless, many of the $\chi$ guarded process terms can be translated using the informal translation given above, as well as using the properties of $\chi$ operators given in [9]. For instance, the non-translated guarded sequential composition $b \rightarrow (p; q)$ may become translatable after re-writing it as $(b \rightarrow p); q$.

Furthermore, although we omit the proofs here, for the given subset of timed $\chi$ under the condition that there is no clocks in guard expressions, it can be shown that $[b \rightarrow p]$ is bisimilar to $b \rightarrow [p]$ and $b_1 \rightarrow b_2 \rightarrow p$ is bisimilar to $b_1 \land b_2 \rightarrow p$.

The other thing to be considered is that there is no notion of successful termination in timed automata. Thus, to check if there is no deadlocks in the model we have to use the property $A[\_a1.end \land a2.end \land \ldots \land an.end \implies \text{no deadlock, where } a1.end, a2.end, \ldots, an.end \text{ are the final locations.}$

7 Case Study: Verification of the Turntable System

7.1 System description

As an example we consider the translation of a part of a turntable system. The turntable system illustrates a part of real-life manufacturing system belonging to the application domain of (real-time) control research [5;6;7]. In [4] the $\chi_c$ model of the system was manually translated to Promela, $\mu$CRL and UPPAAL timed automata, and then verified using the Spin, CADP and UPPAAL tools, respectively. In this case study we created a new turntable model using only translatable subset of timed $\chi$. After that we used the translator to obtain the corresponding UPPAAL model. Finally, it was verified using UPPAAL model checker.

The turntable system consists of a round turntable, a clamp, a drill and a testing device (Figure 11). The turntable transports products to the drill and the testing device. The drill drills holes in the products. After drilling a hole, the products are delivered to the tester, where the depth of the hole is measured, since it is possible that drilling went wrong. To control the turntable system, sensors and actuators are used. A sensor detects a physical phenomenon, and changes its state. The controller reads the state of the sensor, and sends output to actuators. The actuators translate output from the controller to a physical change in the machine.

The turntable has four slots that can hold a product. Each slot can hold at most one product and can be in input, drill, test or output position. There are three sensors attached to the turntable: the sensor $s_1$ at the input position (to detect if a product has been added by the environment), the sensor $s_3$ in the output position (to detect if a product has been removed by the environment) and the sensor $s_2$ that detects whether the turntable has completed the turn.
The drilling module consists of the drill and the clamp. Every product should be locked before drilling and unlocked afterwards. To detect whether the clamp is locked or not two sensors are used (c1 and c2 respectively). The drill also has two sensors to detect whether the drill is in its up (d1) or down (d2) position. These sensors are located above the surface of the turntable, so it is not possible to say whether the product has been drilled successfully or not.

In the testing position there are two sensors to detect whether the tester has reached its up (t1) or down (t2) position. If the tester has reached its down position the test result of the product is good and if the sensor at the down position did not send a signal during a certain amount of time the test result of the product is bad.

The turntable control system consists of the main controller, turntable controller, drill controller, and tester controller. The main controller supervises the other controllers and the environment. It stores current information about products and operations being performed and based on this information it issues commands to the other controllers and the environment to start operations. When operations are completed the main controller updates the information about the products.

The turntable controller gets signals from the turntable sensors and passes them to the main controller. It also starts rotation of the turntable at the command of the main controller.

The drill controller supervises the drill and the clamp. It switches the drill on/off and commands to lock/unlock the clamp or to start or stop drilling. The drill controller also gets signals from the drill and clamp sensors.

The test controller sends a signal to the tester to start the operation. Then it waits for a signal from the sensor at the down position. If the hole is not deep enough, the sensor is not activated and the current product should be rejected.

The operation-routing sequence of each product is following: add a product to the input
position, make a turn (now product is in the drilling position), lock the clamp, switch on the
 drills, drill, switch off the drill, unlock the clamp, make another turn (now product is in the
test position), test, and make a turn again (product is in the removing position).

No product can be added if the adding slot is not empty. No drilling, testing or removing can
be performed if the corresponding slot is empty. The turntable can treat up to four products
at the same time, that means that the operations can be done in parallel.

7.2 Design rules and assumptions

Creating the model we consider only “good weather” behavior, i.e. the assumption is that
the system works without faults and there is no product loss. The initial state is defined as
follows: all slots are empty and no operation is started.

For reasons of simplicity, we decided to concentrate on the control system. That means that
we do not model material flow as this information can be obtained from the information
stored by the main controller.

We assume that the main controller sends messages to the environment to allow adding and
removing of products and the environment informs the main controller when the operations
are completed. The environment can skip the adding or removing operations. A product
can be removed from the removing position only if it has been drilled properly. If a product
has a good test result and it has not been removed, it should not be drilled and tested again.
If a product has a bad test result it must be drilled and tested again. That means that the
information whether product has been added or removed is necessary only after the rotation
of the turntable.

We also assume that the order of starting and ending of the adding, drilling, testing and
removing operations is not known in advance.

The execution of each turntable operation requires a certain amount of time. Because the
duration of the turntable operations has not been defined anywhere, we have decided to use
the delays, that have been defined in other turntable models, like [6]. We assume that the
environment needs 2 time units to perform adding or removing of a product. The clamp
needs 3 time units to lock or unlock a product. The drilling operation takes 3 time units and
returning the drill to its up position takes 2 time units. Testing and returning the tester to its
initial (up) position require 2 time units each.

Since the turntable model is going to be verified in UPPAAL, we restrict ourselves to use only
the translated subset of timed $\chi$.

7.3 Verified properties

The following properties were verified in this case study:

1. The system does not contain a deadlock, i.e. it cannot come to a state from which it
cannot continue operating.

2. The turntable doesn’t rotate while drilling.

3. If the product has a bad test result then the product remains on the table and is drilled
again.
4. No drilling takes place if there is no product in the slot and no adding is performed if there is a product in the slot.

7.4 \(\chi\) model of the turntable

The turntable system architecture is depicted in Figure 12. The mechanical components are Tester, Drill, Clamp and Table. These components are controlled by switching commands: \(c_{\text{DrillSwitch}}\) switches the drill on/off, \(c_{\text{DrillMove}}\) instructs the drill to start or stop drilling, \(c_{\text{ClampSwitch}}\) instructs the clamp to lock or unlock the product, and \(c_{\text{TesterMove}}\) instructs the tester to start or stop testing. The other signals that are used are \(c_{\text{Turn}}\) (commands the turntable to start rotating), \(c_{\text{EnvAdd}}, c_{\text{EnvRemove}}\) (inform the environment that it can perform adding or removing operations respectively).

![Turntable architecture diagram](image-url)

Figure 12: Turntable architecture.

The control system model consists of the main controller, drill and clamp controller, tester controller and turntable controller: Main controller, Drill controller, and Tester controller, respectively. The component Environment represent the environment.

Although in Figure 12 eight components are depicted, the \(\chi\) model of the turntable consists of ten parallel processes. The reason for that is that some components consist of two independent parts. Two of the processes model the environment (adding and removing a product). The other two model the drill, since the power switch and the drill part are handling commands independently. One process is used to specify the table itself, and another one is used to specify the tester. There are also three controllers: a main controller, drill controller and a tester controller. The environment is represented by two processes; one is for removing products and the other one is for adding products on the table.
Below we show a turntable process as an example. The complete χ model is given in the Appendix 2.1.

The turntable process models the table itself (Fig. 13). It has 4 positions for adding, drilling, testing and removing, modeled as variables $t_1$, $t_2$, $t_3$ and $t_4$, respectively. These variables can have a value 0, if there is no product in the slot, or 1, if there is a product in the slot. The signals from the environment notify the turntable process if a product added or removed via the channels $c_{EnvAdded}$, $c_{EnvRemoved}$ respectively. If there is a product in the adding or removing position, the main controller is informed via the channels $c_{Added}$ and $c_{Removed}$ respectively. When the turntable process gets a command to turn ($c_{Turn}$) from the main controller, it rotates. The rotation takes 4 time units; the slot states are changed accordingly. After that the turntable process informs the main controller that the turn is finished ($c_{Turned}$).

7.5 **UPPAAL model of the turntable**

The χ model of the turntable was translated to UPPAAL model automatically. The UPPAAL model consists of ten automata, nine of them correspond to the χ processes. The additional dummy process has been created to translate guarded assignments and guarded skip.

As an example we show the automaton, which corresponds to the turntable process (Figure 14). The rest of the automata are given in Appendix 2.2.

7.6 **Verification**

After translating the χ model of the complete turntable system to UPPAAL it becomes possible to verify properties such as:

- The absence of deadlock.

- The turntable is not rotating if any of operations (drilling, testing, adding or removing) is being performed.

- The test result of a product will be known not later than 31 seconds after the product has been added.
8 Conclusions and future work

Nowadays, system specification and modeling become more and more important for handling increasing system complexity. Satisfying industry demands on reducing the development time (time-to-market), costs, and increasing reliability of systems requires early detection of the design errors, which reduces the amount of re-work. One of the most popular techniques to make performance analysis is simulation. The process algebraic language $\chi$ has been used extensively to model and simulate the manufacturing systems. However, simulation-based performance analysis becomes insufficient since it cannot guarantee the correctness of the system. In order to check correctness of the systems designed in $\chi$ we suggest to translate $\chi$ models to UPPAAL timed automata and verify their properties using UPPAAL model-checking tool.

In this report, the translation scheme of the subset of $\chi$ to UPPAAL has been presented. The subset includes following process terms: skip, multiple assignment, communication actions send and receive, deadlock, inconsistent process term, delay and delay enabling operator, repetition, sequential and alternative composition. Moreover, the informal (not proved) translation is defined for such process terms as guarded skip and multi-assignment, guarded delayable send and receive, and guarded repetition.

The translation scheme has been implemented as a part of the $\chi$ toolset and the translator from $\chi$ to UPPAAL has been used to verify a part of an industrial system.

The future work includes the proofs for the informal part of the translation, as well as formal translation of such $\chi$ constructs as variable scope operator and process instantiation. There is also a necessity of improving the graphical layout of the translated automata.
Bibliography


1 Timed Chi semantics used in the report

1.1 Skip and Multi-assignment

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

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

Multi-assignment $x_n := e_n$ 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 $x := e$.

$$x_n := e_n \triangleq \{x_n\} : x_i = e_i' \wedge \cdots \wedge x_n = e_n' \gg \tau$$

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 $\{x, y\} : (x = x' + y') \wedge (y = x' - y')$.

1.2 Variable trajectories

During the time transition the valuation at each time point $s \in [0, t]$ is given by $\rho(s) \in \Omega(\sigma, t)$, which is defined in the following way.

$$\Omega(\sigma, t) = \{ \rho \in [0, t] \mapsto (\text{dom}(\rho) \to \Lambda) \} \quad \text{for} \quad t \geq 0$$

$$\forall x \in \text{dom}(\sigma) \setminus \{\text{time}\} : \quad \rho(x) = \text{constant function}$$

$$\forall x \in \text{dom}(\sigma) : \quad (\rho(x))(\sigma(x))$$

$$\forall s \in [0, t] : \quad \rho(s)(\text{time}) = \rho(s)(\text{time}) + s$$

where $\rho \downarrow x$ denotes the function, such that $\text{dom}(\rho \downarrow x) = \text{dom}(\rho)$ and $(\rho \downarrow x)(y) = \rho(y)(x)$ for each $y \in \text{dom}(\rho \downarrow x)$.

1.3 SOS Rules

The semantics of timed $\chi$ is defined through so-called deduction rules [2]. Below we show the SOS rules, which were refereed in this report. 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.

Action predicate

$$\frac{\sigma' \in \Xi(\sigma, J \cup W), \sigma'' \cup \sigma' \vdash r}{(J, R) \vdash \langle W : r \gg l, \sigma' \rangle \xrightarrow{\sigma''} \langle \sigma', \sigma'' \rangle} \text{T-1}$$

43 Timed Chi semantics used in the report
Send and receive

\[
E \vdash (W : r \gg I_s, \sigma) \xrightarrow{\sigma} T-2
\]

\[
\sigma' \in \Xi(\sigma, J) \\
(j, R) \vdash (h!!e_n, \sigma) \xrightarrow{\sigma', \text{ma}(h,e_n[\sigma])), \sigma'} \langle \checkmark, \sigma' \rangle T-3
\]

\[
\sigma' \in \Xi(\sigma, J \cup \{x_n\}), \sigma'(x_n) = e_n \\
(j, R) \vdash (h??x_n, \sigma) \xrightarrow{\sigma', \text{ma}(h,[x_n], \sigma)), \sigma'} \langle \checkmark, \sigma' \rangle T-4
\]

\[
E \vdash (h!!e_n, \sigma) \xrightarrow{\sigma} T-5 \\
E \vdash (h??x_n, \sigma) \xrightarrow{\sigma} T-6
\]

Consistent deadlock

\[
E \vdash (\delta, \sigma) \xrightarrow{\sigma} T-7
\]

Inconsistent process term  
Inconsistent process term \( \perp \) is considered to be in an inconsistent state from its start. It cannot perform neither action transitions, nor time transitions.

Any delay operator

\[
E \xrightarrow{\langle p, \sigma \rangle \xrightarrow{a} \langle p', \sigma' \rangle} T-8 \\
\langle [p], \sigma \rangle \xrightarrow{\sigma'} \langle p', \sigma' \rangle T-9
\]

\[
E \vdash (\sigma) \xrightarrow{\sigma} T-10
\]

Sequential composition operator

\[
E \xrightarrow{\langle p, q, \sigma \rangle \xrightarrow{\sigma, a, \sigma'} \langle \checkmark, \sigma' \rangle, \langle q, \sigma' \rangle \xrightarrow{\sigma'} \langle q, \sigma' \rangle} T-11 \\
E \xrightarrow{\langle p, q, \sigma \rangle \xrightarrow{a} \langle p', \sigma' \rangle} T-12
\]

\[
E \xrightarrow{\langle \rho, [p], \sigma \rangle \xrightarrow{\sigma} \langle [p], \rho(\sigma) \rangle} T-13 \\
E \xrightarrow{\langle p, q, \sigma \rangle \xrightarrow{\rho} \langle p', q, \sigma' \rangle} T-14
\]
Alternative composition operator

\[
E \quad \frac{\langle p, \sigma \rangle \xrightarrow{\tau, \rho} \langle p', \sigma' \rangle, \langle q, \sigma \rangle \xrightarrow{\tau, \rho} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \xrightarrow{\tau, \rho} \langle p' \parallel q', \sigma' \rangle}
\]

\[
\frac{\langle p, \sigma \rangle \xrightarrow{\tau, \rho} \langle p', \sigma' \rangle, \langle q, \sigma \rangle \xrightarrow{\tau, \rho} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \xrightarrow{\tau, \rho} \langle p' \parallel q', \sigma' \rangle}
\]

Parallel composition operator

\[
\frac{\langle p, \sigma \rangle \xrightarrow{\tau, \rho} \langle p', \sigma' \rangle, \langle q, \sigma \rangle \xrightarrow{\tau, \rho} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \xrightarrow{\tau, \rho} \langle p' \parallel q', \sigma' \rangle}
\]

\[
\frac{\langle p, \sigma \rangle \xrightarrow{\tau, \rho} \langle p', \sigma' \rangle, \langle q, \sigma \rangle \xrightarrow{\tau, \rho} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \xrightarrow{\tau, \rho} \langle p' \parallel q', \sigma' \rangle}
\]

\[
\frac{\langle p, \sigma \rangle \xrightarrow{\tau, \rho} \langle p', \sigma' \rangle, \langle q, \sigma \rangle \xrightarrow{\tau, \rho} \langle q', \sigma' \rangle}{\langle p \parallel q, \sigma \rangle \xrightarrow{\tau, \rho} \langle p' \parallel q', \sigma' \rangle}
\]

45 Timed Chi semantics used in the report
2 The $\chi$ and UPPAAL models of the turntable

2.1 The $\chi$ model of the turntable

```plaintext
model Turntable() =
| chan cEnvAdded?, cEnvRemoved?, cTurn?, cTurned?: void
 , cAdded?, cRemoved?: nat
 , cAdd?, cRemove?: nat
 , cClampSwitch!: void, cLocked?: nat
 , cDrillSwitch!, cDrillMove!: void
 , cDrillSwitched?, cDrillMoved!: nat
 , cTesterMove!: void, cTesterMoved!: nat
 , cTest!: void, cTested!: nat
 , cDrill!, cDrilled!: void
 , var t1: nat = 0, t2: nat = 0, t3: nat = 0, t4: nat = 0, ts: nat
 , add_allowed: nat = 0, remove_allowed: nat = 0
 , clamp_on: nat = 0
 , drill_on: nat = 0, drill_up: nat = 0
```
the χ and UPPAAL models of the turntable
// Drill Controller
|| *( [cDrill??]; [cClampSwitch!!]; [cLocked??d]
| d = 0 -> drill_error := 1
| d /= 0 -> [cDrillSwitch!!]; [cDrillSwitched??d]
| d = 0 -> drill_error := 2
| d /= 0 -> [cDrillMove!!]; [cDrillMoved??d]
| d = 0 -> drill_error := 3
| d /= 0 -> [cDrillMove!!]; [cDrillMoved??d]
| d = 1 -> drill_error := 4
| d /= 1 -> [cDrillSwitch!!]
| [cDrillSwitched??d]
| ( d = 1 -> drill_error := 5
| d /= 1 -> [cClampSwitch!!]
| [cLocked??d]
| ( d = 1 -> drill_error := 6
| d /= 1 -> [cDrilled!!]
)

// Tester Controller
|| *( [cTest??]
| [cTesterMove!!]
| [cTesterMoved??t]
| ( t = 0 -> tester_error := 1
| t /= 0 -> test_result := 1
| delay 3; test_result := 0
| [cTesterMove!!]
| [cTesterMoved??t]
| ( t = 1 -> tester_error := 2
| t /= 1 -> [cTested!!test_result]

// Main controller
|| *( ( p1 = 0 -> [cAdd!!1] | p1 /= 0 -> skip )
| ( p4 = 3 -> [cRemove!!1] | p4 /= 3 -> skip )
| ( p2 = 1 -> [cDrill!!] | p2 /= 1 -> skip )
| ( p3 = 2 -> [cTest!!] | p3 /= 2 -> skip )
| ( p3 /= 2 -> skip
| p3 = 2 -> [cTested??m]; ( m = 0 -> p3 := 1 | m /= 0 -> p3 := 3 )
)
| ( p2 = 1 -> [cDrilled??]; p2 := 2 | p2 /= 1 -> skip )
| [cAdd!!0]; [cRemove!!0]
| [cAdded??p1]; [cRemoved??p4]
| [cTurn!!]; ps:=p4; p4:=p3; p3:=p2; p2:=p1; p1:=ps; [cTurned??] )
2.2 The UPPAAL model of the turntable

![Turntable process diagram]

Figure 15: Turntable process.

![Adding process of the environment diagram]

Figure 16: Adding process of the environment.
Figure 17: Removing process of the environment.

Figure 18: Clamp process.

Figure 19: Switching process of the drill.

Figure 20: Moving process of the drill.
Figure 21: Tester process.

Figure 22: Drill controller process.

51 The $\chi$ and UPPAAL models of the turntable
Figure 23: Tester controller process.

Figure 24: Main controller process.

Figure 25: Dummy process.