VERIFICATION OF TIMED CHI MODELS USING UPPAAL

E.M. Bortnik, D.A. van Beek, J.M. van de Mortel-Fronczak, J.E. Rooda
Department of Mechanical Engineering, Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

Keywords: Discrete-event systems, process algebra, timed automata, performance analysis, functional analysis, verification.

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 χ 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 paper, we represent a translation scheme that is used to translate simulation models written in χ language to UPPAAL timed automata and show a small example of the translation. Future work includes defining an equivalence relation between χ and UPPAAL transition systems, implementing the translator as a part of the χ toolset, and applying it for verification of models of manufacturing systems.

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 make 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[1] (Tools and Techniques for Integrating Performance Analysis and System Verification) is to combine performance and functional analysis, particulary in the χ environment.

The χ language is intended for modeling, simulation, verification, and real-time control of manufacturing systems (van Beek et al., 2004). It is used to model and simulate discrete-event, continuous or combined, so-called hybrid, systems. The χ language has a 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 (van Beek et al., 2002).

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 simple but representative model was manually translated to μCRL, Promela and UPPAAL timed automata, and verified in CADP, Spin and UPPAAL, respectively (Bortnik et al., 2005). In this paper, a general translation scheme from a subset of χ to UPPAAL is described. Using the scheme, the χ toolset will be extended with the translator to make possible to verify χ models in UPPAAL.

The related work includes (Nicollin et al., 1992), where a process algebraic language is defined and then translated into timed automata, and (Daws et al., 1995), where a subset of ET-LOTOS is translated into the KRONOS timed automata. The simi-

[1]supported by the Dutch Organization for Scientific Research (NWO), project number 612.064.205

M. Bortnik E., A. van Beek D., M. van de Mortel-Fronczak J. and E. Rooda J. (2005).
VERIFICATION OF TIMED CHI MODELS USING UPPAAL.
In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics - Robotics and Automation*, pages 486-492
DOI: 10.5220/0001191204860492

lar, *singleformalizm-multisolution*, approach has been used in (D'Argenio et al., 2001; Bohnenkamp et al., 2003), where systems are modelled in stochastic process algebraic language Modest.

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 finite control structure and real-valued clocks (Larsen et al., 1997; Yi et al., 1994). 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 paper is organized as follows. In Section 2, the subset of χ to be translated is described. Then, in Section 3, the formal definition of UPPAAL timed automata is given. The translation is defined in Section 4. In Section 5, 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 6, conclusions are drawn.

2 THE χ LANGUAGE

In order to model timed discrete-event systems only, the hybrid χ language has been simplified, resulting in timed χ (van Beek et al., 2005). In the remainder of this paper, we refer to timed χ as χ. The set M of χ models, that can be translated using this translation scheme, consists of models $M \in \mathrm{M}$, where M is of the following form:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ \ s_1, \ldots, s_k \\
,\ \mathsf{chan}\ h_1, \ldots, h_l \\
,\ s_1 = c_1 \wedge \cdots \wedge s_k = c_k \\
|\ [\![_{\mathrm{V}}\ \mathsf{disc}\ a_1, \ldots, a_n \\
\quad ,\ a_1 = b_1 \wedge \cdots \wedge a_n = b_n \\
\quad |\ p_1 \\
\quad]\!] \\
\|\ \ldots \\
\|\ [\![_{\mathrm{V}}\ \mathsf{disc}\ a_1, \ldots, a_m \\
\quad ,\ a_1 = b_1 \wedge \cdots \wedge a_m = b_m \\
\quad |\ p_r \\
\quad]\!] \\
\rangle\ ,
\end{array}
$$

where $s_1, \ldots, s_k$, $a_1, \ldots, a_n$, and $a_1, \ldots, a_m$ denote the global and local discrete variables, $h_1, \ldots, h_l$ denote the urgent channels, $s_1 = c_1 \wedge \cdots \wedge s_k = c_k$, $a_1 = b_1 \wedge \cdots \wedge a_n = b_n$, and $a_1 = b_1 \wedge \cdots \wedge a_m = b_m$ are initialization predicates that restrict the allowed values of the variables initially, and $\|$ denotes the parallel composition operator. Parallel composition and the variable scope operators are not allowed inside the process terms p_i, since a UPPAAL model is a collection of sequential processes (represented by UPPAAL timed automata) working in parallel.

The set of inductively defined process terms P consists of the following process terms: skip, multi-assignment $\mathbf{x}_n := \mathbf{e}_n$, send $h!!\mathbf{e}_n$ and receive $h??\mathbf{x}_n$, where $\mathbf{x}_n$ and $\mathbf{e}_n$ denote the vectors $(x_1, \ldots, x_n)$ and $(e_1, \ldots, e_n)$, deadlock δ and inconsistent process term $\bot$, delay Δd, where d denotes a constant integer valued expression, delay enabling process term $\lceil p \rceil$, repetition $*p$, sequential composition $p;\ q$, and alternative composition $p \,[\!]\, q$. The detailed explanations can be found in (van Beek et al., 2005).

Formally, the set P of process terms $p \in P$ is defined by:

$$
\begin{array}{rcccccc}
p & ::= & \mathrm{skip} & | & \mathbf{x}_n := \mathbf{e}_n & | & h!!\mathbf{e}_n \\
 & | & h??\mathbf{x}_n & | & \delta & | & \bot \\
 & | & \Delta d & | & \lceil p \rceil & | & *p \\
 & | & p;\ p & | & p' \,[\!]\, p' & &
\end{array}
$$

where p' can be any process term except $*p$ and $\lceil *p \rceil$. Note, that the process term $q \,[\!]\, *p$ still can be translated, since $*p$ can be rewritten as $p; *p$. Similarly, the process term $\lceil *p \rceil$ can be rewritten as $\lceil p \rceil; *p$.

3 UPPAAL TIMED AUTOMATA

In literature, several formal definitions of UPPAAL timed automata can be found (Behrmann et al., 2004; Bengtsson and Yi, 2004; Larsen et al., 1997; Möller, 2002; Yi et al., 1994). For the translation, the formal description of M.O. Möller (Möller, 2002) has been chosen, as it covers most of the features of UPPAAL timed automata that have been implemented in the tool.

A UPPAAL timed automaton $\mathcal{A}$ is a tuple $\langle L, l_0, E, V, C, \mathit{Init}, \mathrm{Inv}, \mathrm{T_L} \rangle$, where L is a finite set of locations, and l_0 is the initial location. The set of the edges E is defined by $E \subseteq L \times \mathcal{G}(C, V) \times \mathit{Sync} \times \mathit{Act} \times L$, where $\mathcal{G}(C, V)$ is the set of constraints allowed in guards, V denotes the set of integer variables, C denotes the set of real-valued clocks ($C \cap V = \emptyset$), and Sync is a set of synchronization actions which includes actions, co-actions, and the internal τ_{h}-action. An action *send* over a channel h is denoted by $h!$ and its co-action *receive* is denoted by $h?$. The τ_{h}-action is an internal action which cannot synchronize and does not have a co-action. Act is a set of assignment actions, which includes assignments, clock resets and the τ_{a}-assignment. The τ_{a}-assignment is an empty assignment, i.e. an assignment that does not change the values of the variables. $\mathit{Init} \subseteq \mathit{Act}$ is a set of assignments that assigns the initial values to variables. The function $\mathrm{Inv} : L \to \mathcal{I}nv(C, V)$ assigns an invariant to each location. $\mathcal{I}nv(C, V)$ is the set of invariants over clocks

C and variables V. The function $\mathrm{T_L} : L \to \{\mathrm{o}, \mathrm{u}, \mathrm{c}\}$ 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.

A network of timed automata NA is a tuple $\langle \overline{A}, \bar{l}_0, V', C', H, \mathrm{T_H}, Init' \rangle$, where $\overline{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is a vector of n timed automata $\mathcal{A}_i = \langle L_i, l_i^0, E_i, V_i, C_i, Init_i, \mathrm{Inv}_i, \mathrm{T^L}_i \rangle$, for $1 \leq i \leq n$. $\bar{l}_0 = (l_1^0, \ldots, l_n^0)$ 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 $\mathrm{T_H} : H \to \{\mathrm{o}, \mathrm{u}\}$ assigns the type (ordinary or urgent) to each channel. In case $H = \emptyset$, function $\mathrm{T_H}$ is undefined and is then informally denoted by $\emptyset$. $Init'$ is the set of assignments that assigns the initial values to the global variables. The formal semantics of UPPAAL timed automata can be found in (Möller, 2002).

4 TRANSLATION SCHEME

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

The translation of timed χ to UPPAAL timed automata is defined by the means of two translation functions. Function $\mathcal{T}_{\mathrm{M}} : \mathrm{M} \to \mathrm{NA}$ translates a χ model M to a UPPAAL network of automata NA using function $\mathcal{T} : P \to \mathrm{A_s}$ that translates a χ process term $p \in P$ to an extended timed automaton. The definition of an extended timed automaton $\mathcal{A}_{\mathrm{s}}$ is based on the definition of the UPPAAL timed automaton, extended with two additional elements: $\mathcal{A}_{\mathrm{s}} = \langle L, l_0, E, V, V^{\mathrm{h}}, C, Init, \mathrm{Inv}, \mathrm{T_L}, l_{\mathrm{f}} \rangle$, where $V^{\mathrm{h}} \subseteq \mathcal{V}^{\mathrm{h}}$ denotes an additional set of variables, that is used for the translation of communication actions, and l_{f} denotes a final location. The final location $l_{\mathrm{f}} \in L \cup \{\top\}$, where $\top$ denotes that there is no final location, and $\mathrm{Inv}(l_{\mathrm{f}}) = \mathrm{true}, \mathrm{T_L}(l_{\mathrm{f}}) = \mathrm{o}$ for all $l_{\mathrm{f}} \in L$, is used for the translation of sequential and alternative composition operators.

4.1 Translation function $\mathcal{T}_{\mathrm{M}}$

The translation function $\mathcal{T}_{\mathrm{M}}$ translates a χ model M of the form:

$$
\begin{array}{l}
\langle\ \mathsf{disc}\ \ s_1, \ldots, s_k \\
,\ \mathsf{chan}\ h_1, \ldots, h_l \\
,\ s_1 = c_1 \wedge \cdots \wedge s_k = c_k \\
\mid\ [\![_{\mathrm{V}}\ \mathsf{disc}\ a_1, \ldots, a_n \\
\quad ,\ a_1 = b_1 \wedge \cdots \wedge a_n = b_n \\
\quad \mid\ p_1 \\
\quad]\!] \\
\|\ \ldots \\
\|\ [\![_{\mathrm{V}}\ \mathsf{disc}\ a_1, \ldots, a_m \\
\quad ,\ a_1 = b_1 \wedge \cdots \wedge a_m = b_m \\
\quad \mid\ p_r \\
\quad]\!] \\
\rangle\ ,
\end{array}
$$

where we assume $\{s_1, \ldots, s_k\} \subseteq \mathcal{V}$, $\{a_1, \ldots, a_n\} \cup \ldots \cup \{a_1, \ldots, a_m\} \subseteq \mathcal{V}$, $(\{a_1, \ldots, a_n\} \cup \ldots \cup \{a_1, \ldots, a_m\}) \cap \{s_1, \ldots, s_k\} = \emptyset$ and $\{h_1, \ldots, h_l\} \cap (\mathcal{V} \cup \mathcal{V}^{\mathrm{h}} \cup \mathcal{C}) = \emptyset$, to a network of UPPAAL timed automata $NA = \langle \overline{A}, \bar{l}_0, V', C', H, \mathrm{T_H}, Init' \rangle$. The function $\mathcal{T}_{\mathrm{M}}$ is defined as follows:

$$\mathcal{T}_{\mathrm{M}}(M) = \langle \overline{A}, \bar{l}_0, V', \{\mathsf{time}\}, \{h_1, \ldots, h_l\}, \mathrm{T_H}, Init' \rangle,$$

where $\overline{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_r)$ is a vector of r timed automata $\mathcal{A}_i = \mathcal{F}(\mathcal{T}(p_i))$, for $1 \leq i \leq r$, and the function $\mathcal{F} : \mathrm{A_s} \to \mathrm{A}$ transforms an extended automaton into a UPPAAL timed automaton $\mathcal{A}_i$ by removing the set of the global variables V^{h} and the final location l_{f}; $\bar{l}_0 = (l_1^0, \ldots, l_r^0)$ is a vector of the initial locations l_i^0 of the automata $\mathcal{A}_i, 1 \leq i \leq r$; $V' = \cup_{1 \leq i \leq r} V_i^{\mathrm{h}} \cup \{s_1, \ldots, s_k\}$, where V_i^{h} is the set of communication variables of the automaton $\mathcal{T}(p_i)$. Since the channels $h_1, \ldots, h_l$ in the model M are urgent, $\mathrm{T_H}(h_j) = \mathrm{u}, 1 \leq j \leq l$. Finally, $Init' = \{\mathsf{time} := 0, s_1 := c_1, \ldots, s_k := c_k\}$.

4.2 Translation function $\mathcal{T}$

In this section, the translation function $\mathcal{T}(p)$ 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.

$$
\begin{array}{l}
\mathcal{T}(\mathrm{skip}) = \\
\langle\ \{l_0, l_1\}, l_0, \{\langle l_0, \mathrm{true}, \tau_{\mathrm{h}}, \tau_{\mathrm{a}}, l_1 \rangle\} \\
,\ \emptyset, \emptyset, \emptyset, \emptyset, \mathrm{Inv}, \mathrm{T_L}, l_1 \\
\rangle,
\end{array}
$$

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

Multi-assignment

Multi-assignment $\mathbf{x}_n := \mathbf{e}_n, n \geq 1$ is an abbreviation for an internal action that changes the values of the variables $x_1, \ldots, x_n$ to the values of

expressions $e_1, \ldots, e_n$.

$$\begin{array}{l}\mathcal{T}(\mathbf{x}_n := \mathbf{e}_n) = \\ \langle\ \{l_0, l_1\}, l_0 \\ ,\ \{\langle l_0, \text{true}, \tau_\text{h}, \{x_1 := e_1, \ldots, x_n := e_n\}, l_1\rangle\} \\ ,\ \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, \text{T}_\text{L}, l_1 \\ \rangle,\end{array}$$

where $\text{Inv}(l_0) = \text{true}, \text{Inv}(l_1) = \text{true}, \text{T}_\text{L}(l_0) = \text{u}, \text{T}_\text{L}(l_1) = \text{o}$.

Send and Receive

Undelayable send and receive process terms $h!!\mathbf{e}_n$ and $h??\mathbf{x}_n$ denote undelayable sending of expressions $\mathbf{e}_n$ via channel h and undelayable receiving via channel h into variables $\mathbf{x}_n$.

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 $f^\text{v} : H \times \mathbb{N} \to \mathcal{V}^\text{h}$ that generates unique names of the communication variables: $y_i = f^\text{v}(h, i), i \in [1, n]$.

$$\begin{array}{l}\mathcal{T}(h!!\mathbf{e}_n) = \\ \langle\ \{l_0, l_1\}, l_0 \\ ,\ \{\langle l_0, \text{true}, h!, \{y_1 := e_1, \ldots, y_n := e_n\}, l_1\rangle\} \\ ,\ \emptyset, \{y_1, \ldots, y_n\}, \emptyset, \emptyset, \text{Inv}, \text{T}_\text{L}, l_1 \\ \rangle,\end{array}$$

where $\text{Inv}(l_0) = \text{true}, \text{Inv}(l_1) = \text{true}, \text{T}_\text{L}(l_0) = \text{u}, \text{T}_\text{L}(l_1) = \text{o}$.

$$\begin{array}{l}\mathcal{T}(h??\mathbf{x}_n) = \\ \langle\ \{l_0, l_1\}, l_0 \\ ,\ \{\langle l_0, \text{true}, h?, \{x_1 := y_1, \ldots, x_n := y_n\}, l_1\rangle\} \\ ,\ \emptyset, \{y_1, \ldots, y_n\}, \emptyset, \emptyset, \text{Inv}, \text{T}_\text{L}, l_1 \\ \rangle,\end{array}$$

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

Deadlock

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

$$\mathcal{T}(\delta) = \langle \{l_0\}, l_0, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, \text{T}_\text{L}, \top \rangle,$$

where $\text{Inv}(l_0) = \text{true}, \text{T}_\text{L}(l_0) = \text{u}$.

Inconsistent process term

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

$$\mathcal{T}(\perp) = \langle \{l_0\}, l_0, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \text{Inv}, \text{T}_\text{L}, \top \rangle,$$

where $\text{Inv}(l_0) = \text{false}, \text{T}_\text{L}(l_0) = \text{u}$.

4.2.2 Translation of the operators

In the translation of the operators, the extended automaton that is obtained by translating the process term $p \in P$ is denoted by $\mathcal{T}(p) = \mathcal{A}_\text{s}^p$, where $\mathcal{A}_\text{s}^p = \langle L^p, l_0^p, E^p, V^p, V^{\text{h}p}, C^p, Init^p, \text{Inv}^p, \text{T}_\text{L}^p, l_\text{f}^p \rangle$. In a similar way, $\mathcal{A}_\text{s}^q$ denotes $\mathcal{T}(q)$.

For the translation some additional functions are needed. The restriction of a function $f : A \to B$ to $C \subseteq A$ is denoted by $f \restriction C$. 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 $\mathcal{E}, \mathcal{L}, \mathcal{A}ct$, where $\mathcal{E}$ is a set of edges, $\mathcal{L}$ is a set of locations, and $\mathcal{A}ct$ is a set of assignments, two more functions are defined. Function $\gamma : \mathcal{P}(\mathcal{E}) \times \mathcal{L} \times \mathcal{P}(\mathcal{A}ct) \to \mathcal{P}(\mathcal{E})$ transforms the set of edges by adding a set of assignments to the assignment part of all incoming edges of a location. For instance, the function $\gamma(E, l, \{x := 1, y := 3\})$ 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 $\sigma : \mathcal{P}(\mathcal{E}) \times \mathcal{L} \times \mathcal{L} \to \mathcal{P}(\mathcal{E})$ transforms the set of edges by replacing all occurrences of the first location with the second one. For instance, the function $\sigma(E, l, l')$ returns the set of edges, where all occurrences of the location l are replaced with l'.

Variable scope operator

Local variables are introduced in a χ process by means of the variable scope operator.

$$\begin{array}{l}\mathcal{T}([\![_\text{V}\ \textsf{disc}\ a_1, \ldots, a_n \\ \quad ,\ a_1 = b_1 \wedge \cdots \wedge a_n = b_n \\ \quad \mid\ p \\ \quad]\!]) = \\ \langle L^p, l_0^p, E^p, V^p \cup \{a_1, \ldots, a_n\}, V^{\text{h}p}, C^p \\ , Init^p \cup \{a_1 := b_1, \ldots, a_n := b_n\}, \text{Inv}, \text{T}_\text{L}, l_\text{f}^p \\ \rangle.\end{array}$$

Delay operator

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

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 $f^\text{c} : L \to \mathcal{C}$.

$$\begin{array}{l}\mathcal{T}(\triangle d) = \\ \langle\ \{l_0, l_1\}, l_0, \{\langle l_0, c == d, \tau_\text{h}, \tau_\text{a}, l_1\rangle\} \\ ,\ \emptyset, \emptyset, \{c\}, \{c := 0\}, \text{Inv}, \text{T}_\text{L}, l_1 \\ \rangle,\end{array}$$

where $c = f^{\mathrm{c}}(l_0)$, $\mathrm{Inv}(l_0) = (c \leq d)$, $\mathrm{Inv}(l_1) = \mathrm{true}$, $\mathrm{T_L}(l_0) = \mathrm{o}$, $\mathrm{T_L}(l_1) = \mathrm{o}$.

Delay enabling operator

The delay enabling operator $[p]$ allows time transitions of arbitrary duration for the behavior of p. In order to translate this operator, the initial position of the extended automaton has to become delayable (ordinary).

$$\mathcal{T}([p]) = \langle L^p, l_0^p, E^p, V^p, V^{\mathrm{hp}}, C^p, \mathit{Init}^p, \mathrm{Inv}, \mathrm{T_L}, l_{\mathrm{f}}^p \rangle,$$

where $\mathrm{Inv}(l_0^p) = \mathrm{true}$, $\mathrm{T_L}(l_0^p) = \mathrm{o}$, and $\forall l^p \in L^p \setminus \{l_0^p\} : \mathrm{Inv}(l^p) = \mathrm{Inv}^p(l^p), \mathrm{T_L}(l^p) = \mathrm{T_L^p}(l^p)$.

Repetition

Process term $*p$ represents infinite repetition of process term p. If the extended automaton $\mathcal{A}_{\mathrm{s}}^p = \mathcal{T}(p)$ has a final location ($l_{\mathrm{f}}^p \in L^p$), 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 $\mathcal{A}_{\mathrm{s}}^p$ does not have a final location ($l_{\mathrm{f}}^p = \top$), then $\mathcal{T}(*p) = \mathcal{A}_{\mathrm{s}}^p$. The resulting extended automaton is defined in the following way.

$$\mathcal{T}(*p) = \langle L, l_0^p, E, V^p, V^{\mathrm{hp}}, C^p, \mathit{Init}^p, \mathrm{Inv}^p \restriction L, \mathrm{T_L^p} \restriction L, \top \rangle$$

where if $l_{\mathrm{f}}^p = \top$, then $L = L^p$, and $E = E^p$, otherwise $L = L^p \setminus \{l_{\mathrm{f}}^p\}$, and $E = \sigma(\gamma(E^p, l_{\mathrm{f}}^p, \mathit{Init}^p), l_{\mathrm{f}}^p, l_0^p)$.

Sequential composition

The sequential composition of process terms p and q behaves as process term p until p terminates, and then continues to behave as process term q. If the extended automaton $\mathcal{A}_{\mathrm{s}}^p$ has a final location, the sequential composition $p; q$ is translated by replacing the final location l_{f}^p of the extended automaton $\mathcal{A}_{\mathrm{s}}^p$ with the initial location l_0^q of the extended automaton $\mathcal{A}_{\mathrm{s}}^q$ in the following way.

$$\mathcal{T}(p;\, q) = \langle\, (L^p \setminus \{l_{\mathrm{f}}^p\}) \cup L^q, l_0^p, E, V^p \cup V^q, V^{\mathrm{hp}} \cup V^{\mathrm{hq}}, C^p \cup C^q, \mathit{Init}^p, \mathrm{Inv}, \mathrm{T_L}, l_{\mathrm{f}}^q \,\rangle,$$

where $E = \sigma(\gamma(E^p, l_{\mathrm{f}}^p, \mathit{Init}^q), l_{\mathrm{f}}^p, l_0^q)$, and $\mathrm{Inv} = (\mathrm{Inv}^p \restriction (L^p \setminus \{l_{\mathrm{f}}^p\})) \cup \mathrm{Inv}^q$, $\mathrm{T_L} = (\mathrm{T_L^p} \restriction (L^p \setminus \{l_{\mathrm{f}}^p\})) \cup \mathrm{T_L^q}$.

If the extended automaton $\mathcal{A}_{\mathrm{s}}^p$ has no final location, $\mathcal{T}(p;\, q) = \mathcal{A}_{\mathrm{s}}^p$.

Alternative composition

Alternative composition operator $p \,[\!]\, q$ models a non-deterministic choice between p and q for action transitions. The passage of time by itself cannot result in making a choice. The alternative composition $p \,[\!]\, q$ is translated by merging the initial and final locations of the extended automata $\mathcal{A}_{\mathrm{s}}^p$ and $\mathcal{A}_{\mathrm{s}}^q$ in the following way.

$$\mathcal{T}(p \,[\!]\, q) = \langle\, L^p \cup L', l_0^p, E, V^p \cup V^q, V^{\mathrm{hp}} \cup V^{\mathrm{hq}}, C^p \cup C^q, \mathit{Init}^p \cup \mathit{Init}^q, \mathrm{Inv}, \mathrm{T_L}, l_{\mathrm{f}} \,\rangle,$$

where if $l_{\mathrm{f}}^p \neq \top$ and $l_{\mathrm{f}}^q \neq \top$, then $L' = L^q \setminus \{l_0^q, l_{\mathrm{f}}^q\}$, $E = E^p \cup \sigma(\sigma(E^q, l_0^q, l_0^p), l_{\mathrm{f}}^q, l_{\mathrm{f}}^p)$, and $l_{\mathrm{f}} = l_{\mathrm{f}}^p$.

If $l_{\mathrm{f}}^p = \top$ or $l_{\mathrm{f}}^q = \top$, then $L' = L^q \setminus \{l_0^q\}$, $E = E^p \cup \sigma(E^q, l_0^q, l_0^p)$, and if $l_{\mathrm{f}}^p \neq \top$ then $l_{\mathrm{f}} = l_{\mathrm{f}}^p$, otherwise $l_{\mathrm{f}} = l_{\mathrm{f}}^q$.

The function Inv is defined as follows: $\mathrm{Inv}(l_0^p) = \mathrm{Inv}^p(l_0^p) \wedge \mathrm{Inv}^q(l_0^q)$, and $\mathrm{Inv} \restriction ((L^p \cup L') \setminus \{l_0^p\}) = (\mathrm{Inv}^p \restriction (L^p \setminus \{l_0^p\})) \cup (\mathrm{Inv}^q \restriction L')$.

Finally, if $\mathrm{T_L^p}(l_0^p) = \mathrm{u}$, then $\mathrm{T_L}(l_0^p) = \mathrm{u}$, otherwise $\mathrm{T_L}(l_0^p) = \mathrm{T_L^q}(l_0^q)$. Furthermore, $\mathrm{T_L} \restriction ((L^p \cup L') = (\mathrm{T_L^p} \restriction (L^p \setminus \{l_0^p\})) \cup (\mathrm{T_L^q} \restriction L')$.

5 EXAMPLE OF THE TRANSLATION

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 (Bos and Kleijn, 2001; Bos and Kleijn, 2002; Hofkamp and van Rooy, 2003).

The turntable system consists of a round turntable, a clamp, a drill and a testing device. 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. Here, the translation of the process Tester is shown.

The tester is controlled by one actuator a_1 that is used to start or stop testing. It also has two sensors (s_1, s_2). The sensor s_1 detects whether the tester is in its initial (up) position. The sensor s_2 is used to detect a test result of a product. When the tester gets the signal to start testing it moves down. If the drilling was successful then the tester reaches the sensor s_2

within 2 time units. If the tester does not reach the sensor s_2 within 2 time units, then a hole in a product is not deep enough and a product must be drilled again. In the χ process *Tester*, possible test results are implemented by non-deterministic choice, where the skip process term models failure. The actuator a_1 and sensors s_1, s_2 are implemented as the channels *cTesterUpDown*, *cTesterUpDone*, *cTesterDownDone*, respectively. When the test result of a product is good, the process *Tester* sends a signal via the channel *cTesterDownDone*. Otherwise, it executes an internal action (skip). After this, the process waits for the command to move up to the initial position (*cTesterUpDown*) and then sends a signal via the channel *cTesterUpDone*.

$$
\begin{aligned}
&Tester(\ \mathsf{chan}\ cTesterUpDown,\ cTesterUpDone\\
&\qquad ,\ cTesterDownDone\\
&\qquad)=\\
&[\![\ *(\ [cTesterUpDown\ ??]\\
&\quad ;\ \Delta 2.0\\
&\quad ;\ (cTesterDownDone\ !!\ [\!]\ \mathrm{skip})\\
&\quad ;\ [cTesterUpDown\ ??]\\
&\quad ;\ \Delta 2.0\\
&\quad ;\ [cTesterUpDone\ !!]\\
&\quad)\\
&]\!]
\end{aligned}
$$

The result of applying the given translation scheme to the χ process *Tester* is illustrated in Figure 1.

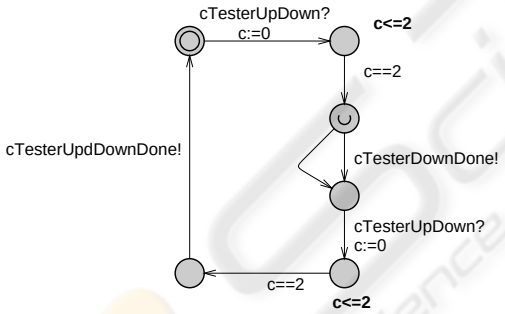

Figure 1: The *Tester* process translation.

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.

More about using UPPAAL for the verification of the turntable model written in χ can be found in (Bortnik et al., 2005).

6 CONCLUSIONS

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 χ 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 χ we suggest to translate χ models to UPPAAL timed automata and verify their properties using UPPAAL model-checking tool.

In this paper, the general translation of the subset of χ 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.

The future work includes translation of the guard operator, defining the equivalence relation between the hybrid transition system of χ and the timed transition system of the UPPAAL timed automata, and extending the χ toolset with the translator from χ to input language of UPPAAL. This will give the possibility to verify system properties such as the absence of a deadlock, as well as other liveness and safety properties.

REFERENCES

Behrmann, G., David, A., and Larsen, K. G. (2004). A Tutorial on UPPAAL. In Bernardo, M. and Corradini, F., editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer-Verlag.

Bengtsson, J. and Yi, W. (2004). Timed Automata: Semantics, Algorithms and Tools. In Reisig, W. and Rozenberg, G., editors, *Lecture Notes on Concurrency and Petri Nets*, number 3098 in LNCS. Springer-Verlag.

Bohnenkamp, H., Hermanns, H., Katoen, J.-P., and Klaren, R. (2003). The Modest Modeling Tool and Its Implementation. In *Lecture Notes in Computer Science*, volume 2794, pages 116 – 133. Springer-Verlag.

Bortnik, E., Trčka, N., Wijs, A., Luttik, B., van de Mortel-Fronczak, J., Baeten, J., Fokkink, W., and Rooda, J. (2005). Analyzing a χ model of a turntable system

using Spin, CADP and Uppaal. To appear in Journal of Logic and Algebraic Programming.

Bos, V. and Kleijn, J. (2001). Automatic Verification of a Manufacturing System. *Robotics and Computer Integrated Manufacturing*, 17:185–198.

Bos, V. and Kleijn, J. (2002). *Formal Specification and Analysis of Industrial Systems*. PhD thesis, Eindhoven University of Technology.

D'Argenio, P. R., Hermanns, H., Katoen, J.-P., and Klaren, R. (2001). MoDeST - A Modelling and Description Language for Stochastic Timed Systems. In *PAPM-PROBMIV '01: Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, pages 87–104. Springer-Verlag.

Daws, C., Olivero, A., and Yovine, S. (1995). Verifying ET-LOTOS programmes with KRONOS. In *Proceedings of the 7th IFIP WG6.1 International Conference on Formal Description Techniques VII*, pages 227–242, London, UK, UK. Chapman & Hall, Ltd.

Hofkamp, A. and van Rooy, H. (2003). *Embedded Systems Laboratory Exercises Manual*. Eindhoven University of Technology, Department of Mechanical Engineering.

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

Möller, M. (2002). *Structure and Hierarchy in Real-Time Systems*. PhD thesis, University of Aarhus.

Nicollin, X., Sifakis, J., and Yovine, S. (1992). Compiling Real-Time Specifications into Extended Automata. *IEEE Trans. Softw. Eng.*, 18(9):794–804.

van Beek, D., Man, K., Reniers, M., Rooda, J., and Schiffelers, R. (2004). Syntax and Consistent Equation Semantics of Hybrid Chi. Technical Report 04-37, Eindhoven University of Technology, Department of Computer Science.

van Beek, D., Man, K., Reniers, M., Rooda, J., and Schiffelers, R. (2005). Syntax and Semantics of Timed Chi. Technical Report 05-09, Eindhoven University of Technology, Department of Computer Science.

van Beek, D., van der Ham, A., and Rooda, J. (2002). Modelling and Control of Process Industry Batch Production Systems. In *15th Triennial World Congress of the International Federation of Automatic Control*, Barcelona, Spain, CD-ROM.

Yi, W., Pettersson, P., and Daniels, M. (1994). Automatic Verification of Real-Time Communicating Systems By Constraint-Solving. In Hogrefe, D. and Leue, S., editors, *Proc. of the 7*th *Int. Conf. on Formal Description Techniques*, pages 223–238. North–Holland.