A Semantic-Preserving Transformation from the Compositional Interchange Format to UPPAAL [★]

D.E. Nadales Agut [*] M.A. Reniers [*] R.R.H. Schiffelers [*,]**
K.Y. Jørgensen [***] D.A. van Beek [*]

[*] *Department of Mechanical Engineering*
Eindhoven University of Technology, P.O.Box 513, 5600 MB
Eindhoven, The Netherlands
{d.e.nadales.agut, m.a.reniers, d.a.v.beek}@tue.nl
[**] *ASML, P.O. Box 324, 5500 AH Veldhoven, The Netherlands*
ramon.schiffelers@asml.com
[***] *Department of Computer Science*
Aalborg University, Selma Lagerlöfs Vej 300, 9220
Aalborg East, Denmark
kyrke@cs.aau.dk

Abstract:
The Compositional Interchange Format (CIF), is a modeling formalism for hybrid systems, that aims to establishing interoperability of a wide range of tools by means of model transformations to and from CIF. UPPAAL is a very successful tool for the specification and analysis of timed systems.

It is interesting, both from a theoretical and a practical perspective, to be able to translate CIF models to UPPAAL models, since this makes it possible to model check properties of timed CIF models. This requires a semantic preserving transformation, which ensures that properties validated in UPPAAL models also hold for their CIF counterparts. In addition, by providing such a translation we are, at the same time, providing translations for a wider set of languages that can be transformed to CIF.

This paper presents a semantic-preserving transformation from a subset of CIF models to UPPAAL. The transformation described in this work constitutes the cornerstone for transformations of a broader subset of CIF.

Keywords: Automata, formal languages, specification, verification, transformations, hybrid systems, timed systems

1. INTRODUCTION

In solving particular control design problems many different tools are used. These are tools for

- rigorous simulation of large hybrid systems consisting of complex switched continuous dynamics and distributed, hierarchically organised supervisory controls defined by intuitive, graphical, or textual formalisms.
- model simplification/abstraction which derive consistent over-approximations of complex hybrid dynamic systems for analysis and synthesis purposes.
- identification of hybrid models from simulated or real data.
- system analysis (e.g., reachability analysis, logic verification), controller synthesis, and on-line optimization of classes of simple hybrid systems.
- real-time simulation for validation by hardware-in-the-loop simulation.
- code generation to test controllers in the implementation environment, and to use this code to control the hardware.

Providing interoperability of such software tools has thus far not been realized in a systematic manner. As a first step into this direction, the Compositional Interchange Format (CIF) (Beek et al., 2008; Baeten et al., 2010) has been defined. The purpose of the Compositional Interchange Format is to establish interoperability of a wide range of tools by means of model transformations to and from CIF. The application domain of the CIF consists of languages and tools from computer science and from dynamics and control for modeling, simulation, analysis, controller synthesis, and verification in the area of hybrid, timed, and untimed systems. Using the CIF as intermediate, the implementation of many bi-lateral translators between specific formalisms can be avoided.

[★] This work was partially done as part of the European Community's Seventh Framework Programme (FP7/2007-2013) project MULTIFORM.

CIF has a formal and compositional semantics that is based on (hybrid) transition systems. This means that, in case the other formalism involved also has a formal semantics, it is possible to prove that certain model transformations are property-preserving.

UPPAAL (Larsen et al., 1997) is a very successful tool for the specification and analysis of timed systems. It is therefore very interesting, both from a theoretical and a practical perspective to be able to translate CIF models to networks of UPPAAL automata, since this makes it possible to perform model checking of timed CIF models. In addition, by enabling CIF to UPPAAL transformations we are also enabling transformations from a wider set of languages such as those used for supervisory control synthesis (Wonham, 2007; Ma and Wonham, 2006).

Providing a mathematically correct transformation is crucial. If no proof of correctness is provided, the whole transformation becomes pointless, since we have no guarantee that the property established in the UPPAAL model holds in the original CIF model. UPPAAL has a clear and unambiguous formal semantics in which a timed transition system is associated to a network of UPPAAL automata (Möller, 2002). Consequently, we are able to prove that the transformation from CIF to UPPAAL is semantic-preserving.

We are able to translate a subset of CIF to UPPAAL. This should not come as a surprise, given that CIF features a rich set of concepts for modeling hybrid systems, whereas UPPAAL is based on timed automata, and it provides no corresponding counterparts for some of the concepts supported in CIF, such as action synchronization, time can progress predicates or urgency. The translated subset includes parallel composition, unidirectional communication via channels, shared variables, clocks, and restricted forms of urgency.

The main contribution of this paper is a model transformation of an expressive subset of the CIF formalism to UPPAAL. We show that this model transformation preserves the semantics of the translated CIF models (after some abstraction). Admittedly, once the translatable subset of CIF is identified, the translation is rather simple, which has the nice side effect of being structure preserving. However the difficulties lie in identifying this subset, and providing the proof of correctness. On the other hand, this work constitutes the cornerstone for transformations of a broader subset of CIF, that can be obtained via process algebraic linearization (Usenko, 2002) to a translatable form.

The class of CIF models that can be translated to UPPAAL, can be extended considerably in two ways: (1) many CIF models that do not fit syntactically in the class of translatable CIF models can be transformed in a semantics-preserving way to a CIF model that is translatable, and (2) using the template and instantiation features of UPPAAL, CIF models with hierarchy and process instantiation may also be translated. A problem with the later extension is that there seems to be no formal semantics for these UPPAAL features which makes it impossible to formally prove the correctness of such a translation. In this way, even though we make some restrictions on the target CIF models, it is still possible to deal with a large class of real world models.

The remainder of this work is structured as follows. Sections 2 and 3 introduce CIF and UPPAAL, respectively. Section 5 characterizes the subset of CIF that can be translated to UPPAAL, and the transformation function is defined in Section 6. Section 4 presents the semantics of CIF and UPPAAL, which is required to discuss the proof of correctness sketched in Section 7.

2. CIF IN A NUTSHELL

This section presents a concise syntax and formal semantics of CIF (Beek et al., 2009).

The basic building blocks of CIF are hybrid automata, which consist of a set of locations taken from the set $\mathcal{L}$ and a set of edges that connect them. A location specifies a (discrete) state of the system, and it can have equations associated to it in the form of *time can progress* (tcp) predicates or *invariants*. These equations define the continuous behavior of the automaton: they determine when time can pass, and how the value of the variables change as time elapses. In addition, invariants also restrict the action behavior of the automaton, since this predicate must hold when a new location is entered. The values of the variables belong to the set Λ. Guarded labeled transitions (edges) specify the discrete behavior of the system. These transitions are labeled with actions originating from $\mathcal{A} \cup \{\tau\}$, where $\tau \notin \mathcal{A}$ is the silent action, and they contain an *update expression* that determines how the values of variables change after performing the action.

In the mathematical definition of CIF predicates and variables are abstract entities (see (Beohar et al., 2010) for a discussion). Initialization predicates, invariants, tcp, and guards originate from the set $\mathcal{P}_t$, which represents the set of all predicates having free variables (i.e., variables that are not bound to a quantifier). Variables are taken from $\mathcal{V}$. Reset predicates belong to the set $\mathcal{P}_r$, which represents the set of all predicates having free variables from the set $\mathcal{V} \cup \{x^+ \mid x \in \mathcal{V}\}$. Dotted variables are also supported by CIF but they are not relevant in the context of the present work.

Besides these basic elements, automata contain:

(1) A set of controlled variables (Frehse, 2005). A controlled variable cannot be assigned by any other automaton but the one that declares it, and arbitrary jumps are forbidden (in principle all variables can change in an arbitrary manner in an action transition if they are not restricted by the update expression).
(2) A set of synchronizing actions. If an action is synchronizing, then it must be executed synchronously in a parallel composition. In CIF all actions are non-synchronizing by default.
(3) A dynamic type mapping, which assigns to each variable a *dynamic type* (Lynch et al., 2001). In the context of the present work we omit the dynamic type from CIF automata, since we only consider declaration of variables and clocks at the model level. This helps to keep the presentation simple.

Given these preliminaries, a CIF automaton can be formally defined as follows.

Definition 1 (CIF automaton). *An automaton is a tuple*

$$(V, \mathrm{init}, \mathrm{inv}, \mathrm{tcp}, E, \mathrm{var}_C, \mathrm{act}_S)$$

where $V \subseteq \mathcal{L}$ is the set of locations; $\mathrm{init}, \mathrm{inv}, \mathrm{tcp} \colon V \to \mathcal{P}_t$ *are functions that associate to each location its corresponding initialization predicate, invariant, and tcp predicate, respectively; $E \subseteq V \times \mathcal{P}_t \times (\mathcal{A} \cup \{\tau\}) \times (2^{\mathcal{V}} \times \mathcal{P}_r) \times V$ is the set of edges;* $\mathrm{var}_C \subseteq \mathcal{V}$ *is the set of controlled variables; and* $\mathrm{act}_S \subseteq \mathcal{A}$ *is the set of synchronizing actions.*

Automata can be composed using several operators. We describe only those operators that are relevant in the context of the current work. In general, CIF models are referred to as *compositions*, which we assume to be contained in the set $\mathcal{C}$. The parallel composition operator, denoted as $p \parallel q$, for p and q in $\mathcal{C}$, allows the parallel execution of p and q, which can communicate via shared variables and synchronizing actions. Global variables can be declared at the top level, along with an initialization predicate that assigns initial values to the variables, and urgency declarations that expresses which actions are urgent in the model. The (concrete) syntax for these declarations is given below.

Definition 2 (CIF model). *A CIF model is of the form*

$|[$ disc control $x_0, \ldots, x_{m-1}$
; disc $y_0, \ldots, y_{k-1}$
; clock $c_0, \ldots, c_{t-1}$
; init u
; urgent $a_0, \ldots, a_{v-1}$
:: p $]|$

where $x_0, \ldots, x_{m-1}$ and $y_0, \ldots, y_{k-1}$ are sequences of controlled and uncontrolled discrete variable declarations, respectively; $c_0, \ldots, c_{t-1}$ is a list of clock declarations, $u \in \mathcal{P}_t$ is a predicate, $a_0, \ldots, a_{v-1}$ is a list of urgent action declarations; and $p \in \mathcal{C}$ is a CIF composition where all free variables (i.e., variables that are not bound to an operator) and clocks are declared in the model.

Besides the syntax just described, edges can be annotated with send and receive actions, which have the form $a!$ and $a?$, with $a \in \mathcal{A}$. These actions do not affect CIF semantics, and they are interpreted (that is, the semantics ignores these annotations) as regular actions. They provide a convenient way to identify, in the formal definition of the translation, when a given action is intended to be used as a send or as a receive action.

In what follows we use the following notations related to lists: $[A]$ denotes the set of lists whose elements belong to the set A; $[\ ]$ is the empty list; $[x_0, \ldots, x_{n-1}]$ denotes a sequence of n elements, namely $x_0, \ldots, x_{n-1}$; and given lists xs and ys, $xs +\!\!+\, ys$ denotes the concatenation of xs and ys.

3. UPPAAL IN A NUTSHELL

We have chosen the formal description of UPPAAL syntax presented in (Bortnik et al., 2005), with some minor modifications for the current work. Next, UPPAAL timed automata and networks of automata are defined.

Definition 3 (Timed automaton). *A UPPAAL timed automaton is a tuple $\langle L, l_0, E, \mathrm{inv}, T_L \rangle$ where:*

(1) $L \subseteq \mathcal{L}$ is a set of locations, and l_0 is the initial location.
(2) $E \subseteq L \times \mathcal{P}_u \times \mathcal{A}_u \times \Sigma_l \times L$ is the set of edges of the automaton. The set $\mathcal{P}_u$, contains the allowed predicates in guards (Bortnik et al., 2005). The actions of the automaton are taken from the set $\mathcal{A}_u$. Actions can be simple actions a ($a \in \mathcal{A}$), the silent action τ, send actions $h!$, or receive actions $h?$, where h is a channel (the set of channels is given in Definition 4). We use notation Σ_l to refer to the set of lists of assignments of the form $[x_0 = e_0, \ldots, x_{n-1} = e_{n-1}]$, where $x_i \in \mathcal{V}$, and e_i are integer expressions.
(3) $\mathrm{inv} \in L \to \mathcal{P}_t$ *is a function that assigns an invariant predicate with each location. For a description of the allowed invariants see (Bortnik et al., 2005).*
(4) $T_L \in L \to \{o, u, c\}$ is the function that assigns to each location the type ordinary (o), urgent (u), or committed (c).

In UPPAAL assignments are executed sequentially, from left to right. For instance, given an initial valuation $\sigma = \{(x, 0), (y, 1)\}$, assignment $[y = 2, x = y+1]$ changes σ into $\{(x, 3), (y, 2)\}$. Given an assignment as, and a valuation σ, $as(\sigma)$ represents the valuation that is obtained after updating the values of the variables according to as. For our previous example, $[y = 2, x = y + 1](\{(x, 0), (y, 1)\}) = \{(x, 3), (y, 2)\}$.

We allow simple actions in UPPAAL automata, which originate from CIF models, to be able to relate these actions to the original CIF models. This simplifies the development of our theoretical work.

Note that the above definition lacks local variables, whereas the UPPAAL tool-set provides support for this. However we could not find a corresponding formalization in literature, which forces us to avoid the treatment of this feature.

Definition 4 (Network of timed automata). *A network of automata is a tuple*

$$\langle \overline{A}, \bar{l}, V, C, H, T_H, \mathrm{init} \rangle$$

where $\overline{A} = [A_0, \ldots, A_{n-1}]$ is a sequence of timed automata; each A_i is of the form $\langle L_i, l_{0_i}, E_i, \mathrm{inv}_i, T_{L_i} \rangle$, and for each assignment $[x_0 = e_0, \ldots, x_{n-1} = e_{n-1}]$, if $x_i \in C$ then $e_i = 0$; $\bar{l} = [l_{0_0}, \ldots, l_{0_{n-1}}]$ is the initial location sequence; $V \subseteq \mathcal{V}$ and $C \subseteq \mathcal{V}$ are disjoints sets of global variables and clocks, respectively, and H is a set of channels. Function $T_H \in H \to \{o, u\}$ determines whether a given channel is urgent (u) or not. Assignment $\mathrm{init} \in \Sigma_l$ *determines the initial values of* all *variables in V, and for all $c \in C$,* $\mathrm{init}(c) = 0$.

4. CIF AND UPPAAL SEMANTICS

Before talking about the correctness of the translation schema presented in Section 6, it is necessary to present the semantic models for the two formalisms we are relating in this paper.

The semantics of CIF models is defined through a hybrid transition system (Cuijpers and Reniers, 2008). The states of the hybrid transition system are of the form $\langle p, \sigma \rangle$, where $p \in \mathcal{C}$ is a composition and $\sigma \in \mathcal{V} \to \Lambda$ is a valuation (function that assigns value to variables). There are three kinds of transitions between these states, namely, action, time, and environment transitions. We describe them in detail next.

Action transitions are of the form $(p, \sigma) \xrightarrow{a,b,X} (p', \sigma')$, and they model the execution of an action a by composition p in an initial valuation σ, which changes composition p into p' and results in a new valuation σ'. Label b is a boolean that indicates whether action a is synchronizing, and label X is the set of controlled variables.

Time behavior is captured by *time transitions.* Time transitions are of the form $(p, \sigma) \overset{\rho,A,\theta}{\longmapsto} (p', \sigma')$, and they model the passage of time in composition p, in an initial valuation σ, which results in a composition p' and valuation σ'. Function $\rho : \mathbb{T} \rightharpoonup \Sigma$ is the variable trajectory that models the evolution of variables during the time delay, where $\mathbb{T}$ is used to refer to the set of all time points and $A \rightharpoonup B$ denotes the set of partial functions from A to B. Function $\theta : \mathbb{T} \rightharpoonup 2^{\mathcal{A}}$ is called a *guard trajectory* (Beek et al., 2010), and it models the evolution of enabled actions during time delays. For each time point $s \in \mathrm{dom}(\theta)$, the function application $\theta(s)$ yields the set of enabled actions of composition p at time s. For all time transition $\mathrm{dom}(\rho) = [0, t]$, for some time point $t \in \mathcal{T}$, and $\mathrm{dom}(\rho) = \mathrm{dom}(\theta)$. Finally, label A contains the set of synchronizing actions of p and p' [1].

In order to model parallel execution of compositions, we need the notion of *environment transitions*, which are transitions of the form $(p, \sigma) \overset{A}{\dashrightarrow} (p', \sigma')$, and they model the fact that composition p (p') is consistent in valuation σ (σ'). Label A is the set of synchronizing actions of p and p', and the relation between these two components is the same as for time transitions.

The semantics of a UPPAAL network of automata N is defined through a timed transition system (Bengtsson and Yi, 2004). The states of the timed transition system are of the form $(\bar{l}, \sigma)$, where $\bar{l}$ is a sequence of locations taken from the sequence of automata in N, and σ is a valuation, as before. There are two kinds of transitions (or relations) in the induced timed transition system [2] : actions and time transitions. We explain them next.

An action transition is of the form $(\bar{l}, \sigma) \xrightarrow{a} (\overline{l'}, \sigma')$, where label a is either an action label or a channel. Time transitions are of the form $(\bar{l}, \sigma) \overset{t}{\longmapsto} (\overline{l'}, \sigma')$, where $t \in \mathbb{T}$ denotes the duration of the delay.

Having defined the semantic frameworks for the two formalisms, the next section relates the hybrid transition system of a CIF model with the timed transition system of its translation to UPPAAL.

5. DESCRIPTION OF THE TRANSLATED SUBSET

This section describes the subset of CIF that we can currently translate to UPPAAL. We do so by specifying syntactic constraints on CIF compositions.

Not every CIF automaton can be translated to UPPAAL because there are constructs, such as urgency or synchronization, that cannot be expressed, in their full generality, in the latter. The subset of CIF automata, which we are able to translate, can be informally described as follows.

Initial locations Only automata having a unique initial location are translated.

Local urgency UPPAAL provides a limited mechanism for expressing urgency in the models: either time can progress in a location, or no time can pass. Thus, the tcp conditions in CIF automata are restricted to true or false.

Controlled variables The only controlled variables that can be declared in an automaton are those used to receive values via channels.

Synchronizing actions Synchronizing actions are used to implement binary channels. The set of synchronizing actions that can be declared in the automaton must coincide with the set of send and receive actions used in an automaton.

Actions Action labels can be used for *only one* of the following purposes: as a non-synchronizing action, send action, or receive action. This is due to the fact that we implement CIF synchronization using UPPAAL channels, and the latter is limited to channel synchronization.

Predicates The predicates that can be used in the guards and invariants are restricted to those supported by UPPAAL.

Edges The edges of the translatable CIF automata can perform *regular actions* (of the form a, with $a \in \mathcal{A}_\tau$), send actions ($a!$), or receive actions ($a?$). CIF allows a richer set of update predicates than UPPAAL, which solely supports *sequential* assignments to variables and clock resets. Thus, we can translate CIF updates only if they produce a state change equivalent to some *sequential assignment.* Definition 5 characterizes such assignments.

In order to formalize the restrictions above, we need some definitions.

Definition 5 (Translatable updates)**.** *An update predicate is translatable if it is written as a conjunction of equalities*

$$x_0^+ = e_0 \wedge \ldots \wedge x_{n-1}^+ = e_{n-1}$$

such that the following conditions hold.

(1) $\langle \forall i, j : 0 \leq i \leq j < n : x_j{}^+ \notin FV(e_i) \rangle$
(2) $\langle \forall i, j : 0 \leq i < j < n : x_i \notin FV(e_j) \rangle$
(3) $\langle \forall i, j : 0 \leq i < j < n : x_i \neq x_j \rangle$

where $FV(e)$ is the set of all free variables occurring in expression e.

An update is translatable if it is of the form

$$(\{x_0, \ldots, x_{n-1}\}, x_0^+ = e_0 \wedge \ldots \wedge x_{n-1}^+ = e_{n-1})$$

and the update predicate is translatable.

Note that a given update predicate may not be translatable, but it can be rewritten into an equivalent form such that it is translatable.

Definition 6 (Ordinary, send, and receive actions)**.** *Given an automaton α with a set of edges E we define* [3] *:*

[1] The set of synchronizing actions are not changed by time transitions

[2] This is the least timed transition system that satisfies the operational semantics rules.

[3] Symbol $\triangleq$ denotes "equal by definition" (as opposed to semantic equality).

$$\begin{aligned} ord(\alpha) &\triangleq \{a \mid a \in \mathcal{A} \wedge (l, g, a, u, l') \in E\} \\ snd(\alpha) &\triangleq \{a \mid a \in \mathcal{A} \wedge (l, g, a!, u, l') \in E\} \\ rcv(\alpha) &\triangleq \{a \mid a \in \mathcal{A} \wedge (l, g, a?, u, l') \in E\} \end{aligned}$$

Using the definitions above, the set of CIF automata that can be translated can be formally defined as follows.

Definition 7 (Translatable automaton). *A CIF automaton $(L, \text{init}, \text{inv}, \text{tcp}, E, \text{var}_C, \text{act}_S)$ is said to be translatable if the following conditions are met.*

(1) init *is of the form* $\mapsto l_0 \triangleq \{(l, l = l_0) \mid l \in L\}$ *for some* $l_0 \in L$.
(2) $\text{ran}(\text{tcp}) = \{\text{true}, \text{false}\}$, *where* $\text{ran}(f)$ *denotes the range of function* f.
(3) *Each edge in* E *is of the form* (l, g, x, u, l'), *where* x *is of the form* τ, a, $a!$, *or* $a?$; $a \in \mathcal{A}$, *and* u *is a translatable update.*
(4) $ord(\alpha)$, $snd(\alpha)$, *and* $rcv(\alpha)$ *are pairwise disjoint.*
(5) $snd(\alpha) \cup rcv(\alpha) = \text{act}_S$

The next step is to characterize the set of CIF compositions that can be translated to UPPAAL. Informally, the *translatable compositions* are parallel compositions of n automata. All variables in the used composition must be declared, and controlled (either at the top level or controlled by a single automaton). The initialization predicate must assign unique initial values to model variables. Synchronization between the automata is restricted to point to point synchronization (i.e., synchronization where exactly two parties are involved).

The restrictions presented below, allow us to get CIF compositions that can be translated to UPPAAL. All variables are required to be controlled because if a variable is allowed to jump arbitrarily, then there is an infinite set of possible values that it can receive, and this cannot be modeled in the target language. Synchronization can take place only between two automata. Moreover, if two automata synchronize via some shared action a, then one of them can send data using a, and the other can only receive data when it synchronizes in a. If we do not have this restriction, then, given the way channels are implemented, a send action can be synchronized with a send action in the CIF model. Finally, an automaton cannot modify variables that are declared as controlled in the other automata. This is required because the concept of controlled variables is not supported in UPPAAL.

Formally, the class of CIF models that can be translated to UPPAAL is defined as follows.

Definition 8 (Translatable model). *A CIF model is said to be translatable if it is of the form*

$$\begin{aligned} &\|[\text{ disc control } x_0, \dots, x_{m-1} \\ &; \text{disc } y_0, \dots, y_{k-1} \\ &; \text{clock } c_0, \dots, c_{t-1} \\ &; \text{init } \bigwedge_{0 \le i < m} x_i = e_i \wedge \bigwedge_{0 \le i < k} y_i = d_i \wedge \bigwedge_{0 \le i < t} c_i = 0 \\ &; \text{urgent } a_0, \dots, a_{v-1} \\ &:: \alpha_0 \parallel \dots \parallel \alpha_{n-1} \text{ }]| \end{aligned}$$

where

(1) *Automaton* $\alpha_i = (L_i, \text{init}_i, \text{inv}_i, \text{tcp}_i, E_i, \text{var}_{Ci}, \text{act}_{Si})$ *is translatable, for all* i *such that* $0 \le i < n$.
(2) $\text{var}_{C0} \cup \ldots \cup \text{var}_{Cn-1} = \{y_0, \dots, y_{k-1}\}$.
(3) *Guards of urgent actions cannot contain clock variables.*
(4) $e_i, d_j \in \mathbb{Z}$, *for all* i, j *such that* $0 \le i < m$ *and* $0 \le j < k$.
(5) $\langle \forall i, j : 0 \le i < j < n : snd(\alpha_i) \cap snd(\alpha_j) = \emptyset \rangle$.
(6) $\langle \forall i, j : 0 \le i < j < n : rcv(\alpha_i) \cap rcv(\alpha_j) = \emptyset \rangle$.
(7) $\langle \forall i, j : 0 \le i, j < n \wedge i \ne j : (rcv(\alpha_i) \cup snd(\alpha_i)) \cap ord(\alpha_j) = \emptyset \rangle$.
(8) *For each send action* $a!$ *in* α_i *with update* (W_s, u_s), *and for each receive* $a?$ *action in* α_j, *with update* (W_r, u_r), $u_s \wedge u_r$ *must be a translatable update predicate (note that the order of the conjunction is important).*
(9) $\langle \forall i, j : 0 \le i, j < n \wedge i \ne j : \text{var}_{Ci} \cap jmp(\alpha_j) = \emptyset \rangle$, *where* $jmp(\alpha)$ *returns the set of all variables that are assigned in the edges of* α.
(10) *The values of clocks can only be changed to* 0.

6. TRANSLATION SCHEME

In this section we describe how transformation of CIF models to UPPAAL networks of automata can be performed.

Given a translatable update u, $\mathcal{T}_=(u)$ is the translation of the update to a UPPAAL assignment. The translation is formalized as follows.

Definition 9 (Translation of updates). *For any translatable update, we define:*

$$\mathcal{T}_=((\{x_0, \dots, x_{n-1}\}, x_0^+ = e_0 \wedge \ldots \wedge x_{n-1}^+ = e_{n-1})) \triangleq [x_0 = e_0^-, \dots, x_{n-1} = e_{n-1}^-]$$

where given an expression e, *notation* e^- *is used to refer to the expression obtained after replacing all occurrences of free variables of* e *having the form* x^+, *by* x.

The following lemma is useful in proving the correctness of Theorem 1. It states that a translatable CIF update and its corresponding translation result in the same change in the valuation.

Lemma 1 (Translatable updates and assignments). *For any translatable update of the form* (W, u), *for any valuations* σ, σ', *such that* $\text{dom}(\sigma) = \text{dom}(\sigma')$ *and* $\langle \forall x : x \in \text{dom}(\sigma) : \sigma(x) = \sigma'(x) \rangle$, *we have that*

$$\sigma'^+ \cup \sigma \models u \Leftrightarrow (\mathcal{T}_=(u))(\sigma) = \sigma'$$

where $\sigma'^+ = \{(x^+, v) \mid (x, v) \in \sigma'\}$, *and relation* $\sigma \models u$ *expresses that predicate* $u \in \mathcal{P}_t \cup \mathcal{P}_r$ *is satisfied (i.e., it is true) in valuation* σ.

In a similar way, we can define the translation of initialization predicates.

Definition 10 (Translation of initializations). *Given variables* $x_0, \dots, x_{n-1}$, *and integer values* $v_0, \dots, v_{n-1}$ *we define:*

$$\mathcal{T}_i(x_0 = v_0 \wedge \ldots \wedge x_{n-1} = v_{n-1}) = [x_0 = v_0, \dots, x_{n-1} = v_{n-1}]$$

The translation of tcp predicates to a function that assigns to each location the *urgent* or *ordinary* attribute is defined as follows.

Definition 11 (Translation of tcp predicates). $\mathcal{T}_{o,u}(\text{tcp}) \triangleq \{(l, o) \mid l \in \text{dom}(\text{tcp}) \wedge \text{tcp}(l)\} \cup \{(l, u) \mid l \in \text{dom}(\text{tcp}) \wedge \neg\text{tcp}(l)\}$

The transformation from CIF automata to UPPAAL automata is given next.

Definition 12 (Translation of CIF automata). *For any translatable automaton, function $\mathcal{T}_a$ is defined by means of the following equation.*

$$\mathcal{T}_a((L, \mapsto l_0, \mathrm{inv}, \mathrm{tcp}, E, \mathrm{var}_C, \mathrm{act}_S)) = \\ \langle L, l_0, E', \mathrm{inv}, \mathcal{T}_{o,u}(\mathrm{tcp})\rangle$$

where $E' = \{(l, g, x, \mathcal{T}_{=}(u), l') \mid (l, g, x, (W, u), l') \in E\}$.

As we saw in Section 5, a limited form of non-synchronizing urgent actions is supported in the translatable CIF models. To implement these, we need to use channels in UPPAAL. The idea is to transform every non-synchronizing action, that is declared as urgent, into a send action over an urgent channel (in UPPAAL). In this way, by introducing an extra automaton that is always ready to perform receive actions over these channels, we can implement urgent actions. For achieving this, we need the following definition.

Definition 13 (Receptive UPPAAL automaton). *Given a set of actions A, we define the receptive automaton $P(A)$ as the UPPAAL automaton*

$$\langle\{l_u\}, l_u, E_A, \{(l_u, \mathrm{true})\}, \{(l_u, o)\}\rangle$$

where $E_A = \{\ (l_u, \mathrm{true}, a?, [\ ], l_u)|\ a \in A\}$.

Next we provide a formalization of the transformation function, which translates CIF models to UPPAAL networks of automata. Once the constraints on translatable models are identified, the transformation is relatively straightforward. Note that the receptive automaton needs to be constructed with the urgent actions that are not used as channels. Otherwise we would get a different behavior in the UPPAAL model.

Definition 14 (Translation of CIF models). *Function $\mathcal{T}$ is defined by means of the following equation.*

$$\begin{aligned}\mathcal{T}(|[\ & \text{disc control } x_0, \ldots, x_{m-1} \\ & ;\text{ disc } y_0, \ldots, y_{k-1} \\ & ;\text{ clock } c_0, \ldots, c_{t-1} \\ & ;\text{ init } u \\ & ;\text{ urgent } a_0, \ldots, a_{v-1} \\ & ::\ \alpha_0 \parallel \ldots \parallel \alpha_{n-1}\ ]|) = \\ & \langle \overline{A}, [l_0, \ldots, l_{n-1}], V, C, H, T_H, \mathcal{T}_i(u)\rangle\end{aligned}$$

where

- $\overline{A} = [A_0, \ldots, A_{n-1}, P(\{a_0, \ldots, a_{v-1}\} \setminus ch)]$
- $ch = \bigcup_{0 \le i < n} \mathrm{act}_{S\,i}$
- $\mathcal{T}_a(\alpha_i) = \langle L_i, l_i, E'_i, \mathrm{inv}_i, \mathcal{T}_{o,u}(\mathrm{tcp}_i)\rangle$
- *A_i is equal to $\mathcal{T}_a(\alpha_i)$ except that every urgent action a is replaced by $a!$.*
- $V = \{x_i \mid 0 \le i < m\} \cup \{y_i \mid 0 \le i < k\}$
- $C = \{c_i \mid 0 \le i < t\}$
- $H = ch \cup \{a_0, \ldots, a_{v-1}\}$
- $T_H = \{(a, u) \mid a \in \{a_0, \ldots, a_{v-1}\}\} \cup \{(a, o) \mid a \in H \setminus \{a_0, \ldots, a_{v-1}\}\}$

7. ABOUT THE CORRECTNESS

The ultimate goal of the transformation described in the previous section is to enable the verification of temporal properties of CIF models using UPPAAL model checking capabilities. For this we prove that the hybrid transition system induced by a CIF model and its corresponding translation are bisimilar (modulo certain abstraction), which means that for each CIF composition p, there is a transition in the hybrid transition system induced by p if and only if there is a corresponding transition in the timed transition system of $\mathcal{T}(p)$. Bisimilarity respects most logics for expressing behavioral properties. This guarantees that if a property is asserted as valid in a given model $\mathcal{T}(p)$ by the UPPAAL model checker, then p satisfies the same property.

To be able to formalize the claim above we need to define some notations beforehand, and to tweak the notion of stateless bisimulation to compare two formalisms having different semantics.

We use notation $[\![p_0]\!] \vdash (p, \sigma) \rightarrow (p', \sigma')$ to express the fact that the a transition in the right hand side of the $\vdash$ symbol belongs to the transition system induced by the composition p_0, where $\rightarrow$ can be an action or a time transition (environment transitions are not considered to obtain the induced transition system), and the domain of the valuations in the transition system coincides with the variables and clocks used in the model. Making some abuse of notation, the term $[\![N]\!] \vdash (\bar{l}, \sigma) \rightarrow (\bar{l}', \sigma')$ denotes that the transition on right hand side of the symbol $\vdash$ is in the transition system induced by the UPPAAL network of automata N. Given a UPPAAL network of automata N of the form $\langle \overline{A}, \bar{l}, V, C, H, T_H, \mathrm{init}\rangle$, we use notation $actv(N)$ to refer to the vector on initial locations of N, that is $\bar{l}$.

Next, we define the notion of *bisimulation* (Milner, 1971) between CIF and UPPAAL models.

Definition 15 (Bisimilarity). *A relation $R \subseteq \mathcal{C} \times [\mathcal{L}]$ is a bisimulaton relation with regards to CIF composition p_0 and UPPAAL network of automata N if and only if for all p, l, and σ such that $(p, l) \in R$ the following holds:*

(1) $\langle \forall a, b, X, p', \sigma' : [\![p_0]\!] \vdash (p, \sigma) \xrightarrow{a,b,X} (p', \sigma') : \langle \exists l' :: [\![N]\!] \vdash (l, \sigma) \xrightarrow{a} (l', \sigma') \wedge (p', l') \in R\rangle\rangle$

(2) $\langle \forall a, l', \sigma' : [\![N]\!] \vdash (l, \sigma) \xrightarrow{a} (l', \sigma') : \langle \exists b, X, p' :: [\![p_0]\!] \vdash (p, \sigma) \xrightarrow{a,b,X} (p', \sigma') \wedge (p', l') \in R\rangle\rangle$

(3) $\langle \forall \rho, t, A, \theta, p', \sigma' : [\![p_0]\!] \vdash (p, \sigma) \overset{\rho,A,\theta}{\longmapsto} (p', \sigma') \wedge \mathrm{dom}(\rho) = [0, t] : \langle \exists l' :: [\![N]\!] \vdash (l, \sigma) \overset{t}{\longmapsto} (l', \sigma') \wedge (p', l') \in R\rangle\rangle$

(4) $\langle \forall t, l', \sigma' : [\![N]\!] \vdash (l, \sigma) \overset{t}{\longmapsto} (l', \sigma') : \langle \exists \rho, A, \theta, p' :: [\![p_0]\!] \vdash (p, \sigma) \xrightarrow{\rho,A,\theta} (p', \sigma') \wedge \mathrm{dom}(\rho) = [0, t] \wedge (p', l') \in R\rangle\rangle$

Given a CIF composition p and a UPPAAL network of automata N, we say that they are bisimilar, denoted by $p \underline{\leftrightarrow} N$, if and only if there exists a bisimulation relation R, w.r.t. p and N, such that $(p, actv(N)) \in R$.

Having defined bisimulation, we can state our main theorem as follows.

Theorem 1. *For any translatable CIF composition p_0 it holds that $p_0 \underline{\leftrightarrow} \mathcal{T}(p_0)$.*

Proof. We define relation R as follows.

$$\begin{aligned} R \triangleq & \{(p_0, actv(p_0))\} \cup \\ & \{(p, actv(p)) \mid [\![p_0]\!] \vdash (p_1, \sigma) \overset{\rho,A,\theta}{\longmapsto} (p, \sigma')\} \cup \\ & \{(p, actv(p)) \mid [\![p_0]\!] \vdash (p_1, \sigma) \xrightarrow{a,b,\emptyset} (p, \sigma')\}\end{aligned}$$

where given a composition p, function application $actv(p)$ returns a sequence $xs{+\!\!+}[l_u]$, where xs is the sequence of all the active locations of the automata contained in p, and l_u is the unique location of the receptive automaton.

It can be shown that R is a bisimulation relation.

The full details of the proof are described in (Nadales Agut and Reniers, 2010). □

8. CONCLUSION

We have identified a subset of CIF that can be translated to UPPAAL, providing a proof of correctness that guarantees that all properties validated in UPPAAL for a translated model are valid in the original CIF model as well.

There are many conditions that a CIF model has to fulfill to be regarded as translatable to a UPPAAL model. However we believe that the set of translatable CIF models can be broadened significantly by performing partial elimination of CIF operators such as urgency and synchronization. In this way, the modeler will not have to be aware of such constraints. Providing an algorithm for this is an interesting subject for further research.

The UPPAAL tool-set provides several other constructs, such as local variable declarations, and channel broadcast. However the semantics of these concepts is not formally defined. In (Gómez, 2009), a semantics for broadcast in UPPAAL is given, but only in a limited form. Thus, we would like to investigate whether we can extend this semantics to fully formalize broadcast, and then we could study how the addition of broadcast in the target language can broaden the set of translatable CIF models. As soon as a formalization of these concepts is available we would like to extend the translation presented here.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Alexandre David for his valuable comments while developing this work. Also we would like to thank the anonymous reviewers for their insightful and detailed remarks which helped to improve the quality of this paper.

REFERENCES

Baeten, J.C.M., Beek, D.A.v., Hendriks, D., Hofkamp, A.T., Nadales Agut, D.E., Rooda, J.E., and Schiffelers, R.R.H. (2010). Report describing the extended CIF functionality. `http://cms.multiform.bci.tu-dortmund.de/images/stories/multiform/deliverables/multiform\_d112.pdf`.

Beek, D.A.v., Collins, P., Nadales, D.E., Rooda, J.E., and Schiffelers, R.R.H. (2009). New concepts in the abstract format of the Compositional Interchange Format. In A. Giua, C. Mahuela, M. Silva, and J. Zaytoon (eds.), *3rd IFAC Conference on Analysis and Design of Hybrid Systems*, 250–255. Zaragoza, Spain.

Beek, D.A.v., Reniers, M.A., Rooda, J.E., and Schiffelers, R.R.H. (2008). Concrete syntax and semantics of the Compositional Interchange Format for hybrid systems. In *17th Triennial World Congress of the IFAC*, 7979–7986. Seoul, Korea.

Beek, D., Cuijpers, P., Markovski, J., Nadales Agut, D., and Rooda, J. (2010). Reconciling urgency and variable abstraction in a hybrid compositional setting. In K. Chatterjee and T. Henzinger (eds.), *FORMATS'10*, volume 6246 of *Lecture Notes in Computer Science*, 47–61. Springer Berlin / Heidelberg.

Bengtsson, J. and Yi, W. (2004). Timed automata: Semantics, algorithms and tools. In J. Desel, W. Reisig, and G. Rozenberg (eds.), *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, 87–124. Springer.

Beohar, H., Nadales Agut, D.E., van Beek, D.A., and Cuijpers, P.J.L. (2010). Hierarchical states in the Compositional Interchange Format. *Electronic Proceedings in Theoretical Computer Science*, 32, 42–56.

Bortnik, E.M., Beek, D.A.v., Mortel-Fronczak, J.M.v.d., and Rooda, J.E. (2005). Verification of timed Chi models using Uppaal. In J. Filipe, J.A. Cetto, and J.L. Ferrier (eds.), *ICINCO 2005, Second International Conference on Informatics in Control, Automation and Robotics*, 486–492. INSTICC Press, Barcelona.

Cuijpers, P.J.L. and Reniers, M.A. (2008). Lost in translation: Hybrid-time flows vs. real-time transitions. In M. Egerstedt and B. Mishra (eds.), *HSCC'08*, volume 4981 of *Lecture Notes in Computer Science*, 116–129. Springer.

Frehse, G. (2005). *Compositional Verification of Hybrid Systems using Simulation Relations*. Ph.D. thesis, Universiteit of Nijmegen.

Gómez, R. (2009). A compositional translation of timed automata with deadlines to uppaal timed automata. In J. Ouaknine and F.W. Vaandrager (eds.), *FORMATS'09*, volume 5813 of *Lecture Notes in Computer Science*, 179–194. Springer-Verlag, Berlin, Heidelberg.

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

Lynch, N.A., Segala, R., and Vaandrager, F.W. (2001). Hybrid i/o automata revisited. In *HSCC'01*, volume 2034 of *Lecture Notes in Computer Science*, 403–417. Springer.

Ma, C. and Wonham, W.M. (2006). *Nonblocking supervisory control of state tree structures*, volume 51 of *IEEE Transactions on Automatic Control*. IEEE.

Milner, R. (1971). An algebraic definition of simulation between programs. In *Proceedings of the 2nd international joint conference on Artificial intelligence*, 481–489. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

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

Nadales Agut, D.E. and Reniers, M.A. (2010). Proof of correctness for the main theorem of the paper "a semantic-preserving transformation from the Compositional Interchange Format to UPPAAL". `http://se.wtb.tue.nl/sewiki/cif/publications2`.

Usenko, Y.S. (2002). *Linearization in μCRL*. Ph.D. thesis, Eindhoven University of Technology.

Wonham, W. (2007). *Supervisory control of discrete-event systems*. Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada. URL `http://www.control.toronto.edu/DES/`.