Submodule Construction for Timed Systems

Jawad Drissi
Southwest Texas State University, Computer Science Depart., San Marcos, Texas. USA
jd30@swt.edu

Ahmed Khoumsi
Université de Sherbrooke, Dept. GEGI, Sherbrooke, Québec, CANADA
khoumsi@gel.usherbro.ca

Abstract. This paper addresses the problem of designing a submodule of a given timed system. For representing specifications, we use the model of communicating timed input/output automata. The problem may be formulated mathematically by the equation \((C|X) \mathcal{R} A\) under the constraint \(I_\varepsilon\), where \(C\) represents the specification of the known part of the system, \(X\) represents the context, \(A\) represents the specification of the desired whole system, \(\mathcal{R}\) represents the specification of the submodule to be constructed, \(||\) is a composition operator, \(\mathcal{R}\) is a conformance relation and \(I_\varepsilon\) is the required set of inputs for \(X\). As conformance relation, we consider the safe realization relation. This relation is implied by all the well-known criteria of trace equivalence, complete trace equivalence, quasi equivalence and reduction. We propose an algorithm for solving the problem with respect to the safe realization and we characterize the set of solutions.

1 Introduction

One common problem, encountered in the hierarchical design of complex systems, in the synthesis of controllers and in the reuse of components, is the submodule construction problem, also called factorization problem or equation solving problem. The submodule construction problem (SCP) is to construct the specification of a submodule \(X\) when the specification of the whole system and all submodules except \(X\) are given. Such a problem may be formulated mathematically by the equation \((C|X) \mathcal{R} A\) under the constraint \(I_\varepsilon = I_\varepsilon\), where \(C\) represents the specification of the known part of the system, \(A\) represent the specification of the whole system, \(||\) is a composition operator and \(\mathcal{R}\) is a conformance relation and \(I_\varepsilon\) is the required set of inputs for \(X\). The SCP was first formulated and treated in [Merlin 83], where specifications are expressed in terms of execution sequences, and trace equivalence was used as conformance relation. In [Shields 89], the author uses Milner's Calculus of Communicating Systems to model the same problem. Many other works [Haghverdi 96][Qin 1991] have been done using labeled transition systems as a model for the specifications and the strong and/or the observational equivalences as conformance relations.

In [Drissi 99a], we consider this problem in the context of partial input/output automata for systems specification for various conformance relations. Real-time programs such as airplane controllers, real-time operating systems, switching software and process controllers in manufacturing plants are inherently reactive, and their interaction with the environment must occur in real-time. Others systems in which the explicit notion of time plays an important role are communication systems and particularly communication protocols; the performance of such systems vitally depends on the value of timers, which control message retransmission. The correct operation of such systems is more than logical consistency in terms of event sequences, and extends to the satisfaction of real-time constraints. Timed models have been introduced for the specification and verification of real-time systems [Alur 94a]. In [Maler 95], the authors show that the control synthesis problem is solvable when a timed automaton gives the plant specification. The solution is obtained by solving fixed-point equations involving both discrete transition relations and linear inequalities.

In the above work [Maler 95], it is assumed that the controller can precisely observe the whole configuration of the plant. However, in realistic situations the plant can be observed only up to some equivalence relation on its states, and the controller has to operate under some uncertainty. We generalize our previous work on submodule construction in the case of partial observation [Drissi 99a] by dealing with partial timed input/output automata for systems specification. The conformance relation used is the safe realization relation that is an adaptation of trace inclusion to the case of partial timed input/output automata.

The rest of the paper is structured as follows. In Section 2, we define basic notions. Section 3 presents the submodule construction problem and the architecture in which this problem will be solved. In Section 4, we propose an algorithm which takes as input: a timed input/output automaton \(A\), a timed input/output automaton \(C\), and a set \(I_\varepsilon\) and produces as output a timed input/output automaton \(Sol_\varepsilon\) (if it exists) with \(I_\varepsilon = I_\varepsilon\) such that the composition of \(C\) with \(Sol_\varepsilon\) is conform to \(A\) with respect to the safe realization relation. Then using the previous solution \(Sol_\varepsilon\), we characterize the set of all such solutions. Finally, in Section 5 we conclude the paper. More details about the algorithm can be found in [Drissi 00].
2 Timed input/output Automata

2.1 Basic notions and definitions

In this paper, a timed I/O automaton (briefly TIOA) $A$, is a tuple $(S, I, O, X, M, T, s_0, T_{r})$ where $S$ is a finite set of states with $s_{0}$ as the initial state, $I$ is a non-empty finite set of inputs, $O$ is a non-empty finite set of outputs with $I \cup O = \emptyset$, $X$ is a finite set of clocks, $M$ is a mapping that labels each state $s$ in $S$ with some clock constraints in $\Phi(X(s))$, called state invariant and denoted $Inv(s)$, and $T(s) \subseteq S \times (I \cup O)$ is the transition set. An element $(s, u, x, s') \in T(s)$ represents a transition from state $s$ to state $s'$ on symbol $u$, the set $x \in X$ gives the clocks to be reset with this transition and $x$ is a clock invariant constraint.

A clock interpretation $v$ for a set $X_v$ of clocks assigns a real value to each clock; that is, it is a mapping from $X_v$ to the set $\mathbb{R}$ of nonnegative reals. We say that a clock interpretation $v$ for $X_v$ satisfies a clock constraint $\phi$ over $X_v$ if the valuation of each clock $x \in X_v$ under $v$ is within the bounds given by $\phi$.

If for each $s \in S$ and all $u \in I$ and any clock interpretation $v$ that satisfies $Inv(s)$ there exists $s \in S$ such that $v(x) \leq a$ for all $x \in X$, then $A$ is said to be deterministic. A TIOA $A$ is said to be nondeterministic if there exists a state $s \in S$ such that $\exists s' \in S$ such that $v(x) \leq a$ and $v(x) > a$ for some state $s \in S$ and $v$ is nondeterministic (i.e., $s\in S$). A TIOA $A$ is said to be initially connected if the set $S$ is connected.

The connectedness of a TIOA containing the initial state is the TIOA $CC(A) = (S, I, O, X, M, T, s_0)$ such that $O = \emptyset$ and $X = \emptyset$. The composition of $A_1$ and $A_2$ denoted $A_1 \circ A_2$, is defined as the connected component of the TIOA $A = (S, I, O, X, M, T, s_0)$ such that $O = \emptyset$ and $X = \emptyset$. The composition $A_1 \circ A_2$ is said to be initially connected if the set $S$ is connected.

The composition of TIOA is commutative and associative. This composition allows a number of composing TIOA to accept the same input simultaneously.

We define a safety property that formalizes the non-occurrence of an unspecified reception in the composition $A$ of a collection of TIOA $(A = (S_1, I_1, O_1, X_1, M_1, T_1, s_0))$, $1 \leq \nu \leq \mu$. We denote by $\epsilon$ the empty word.

Definition 2: Given a collection of TIOA $(A = (S_1, I_1, O_1, X_1, M_1, T_1, s_0))$, $1 \leq \nu \leq \mu$, the composition $A = \{A_i\}_{i=1}^{\mu}$ is safe, written $Safe(A)$, if for any timed trace $s$ in $(I \cup O)^{\ast}$ such that $Tr_A(s)$ is in $T_{r}$ for all $i$, $i \in \{1, 2, \ldots, \mu\}$, $A$ is safe.
2.3 Removing $\epsilon$-transitions from a TIOA

When a transition is considered as an invisible action we call it an $\epsilon$-transition. An example is an action that is internal in a composition. It is well known that in the untimed case such transitions do not increase the expressive power of finite automata. In the case of timed automata, the situation is different. The $\epsilon$-transitions with clock resets strictly increase the power of timed automata [Bérard 98].

In the case of $\epsilon$-transitions without reset, i.e. $\epsilon$-transitions which do no reset clocks, we find in [Bérard 96] an algorithm to construct, given a TIOA, an equivalent one without such internal transitions. This algorithm has two steps; it first suppresses the cycles of $\epsilon$-transitions and then the remaining ones. The purpose of an $\epsilon$-transition without reset is only to check that the clock constraint carried by the $\epsilon$-transition is satisfied at some point in time. The main idea to remove this $\epsilon$-transition is to shift this verification either forward or backward. This is done by using the forward closure or the backward closure of a clock constraint and by adding some new clocks.

**Definition 3:** Let $\varphi$ be a clock constraint. The forward closure of $\varphi$, denoted by $f_cl(\varphi)$, is a formula that is satisfied by a clock valuation $\nu$ if $\varphi$ is satisfied by the clock valuation $\nu+\delta$ for some $\delta \geq 0$: $\nu = f_cl(\varphi)$ if $\exists \delta > 0, \nu+\delta = \varphi$.

**Definition 4:** Let $\varphi$ be a clock constraint and let $\lambda$ be a set of clocks. The backward closure of $\varphi$ with respect to $\lambda$, denoted by $b_cl(\varphi)^\lambda$, is a formula that is satisfied by a clock valuation $\nu$ if $\varphi$ is satisfied by the clock valuation $\nu_\lambda = 0 + \delta$ for some $\delta \geq 0$:

$$\nu = b_cl(\varphi)^\lambda$$

if $\exists \delta > 0, \nu_\lambda = 0 + \delta = \varphi$.

**Example:**

- $A = \langle u_1, \varphi_1, \lambda_1, \varphi_2, \emptyset \rangle$
- $B = \langle u_1, \varphi_1, \lambda_1 \cup \{x_0\}, \varphi_2, \emptyset \rangle$
- $C = \langle u_1, \varphi_1, \lambda_2, \emptyset \rangle$
- $D = \langle u_1, \varphi_1 \land b_cl(\varphi_2)^\lambda_1, \emptyset \rangle$

![Figure 1: Two automata and their equivalent without $\epsilon$-transitions](image)

In this example we illustrate how we remove an $\epsilon$-transition without reset. For the automaton $A$ we use the forward closure of the constraint $\varphi_1$ to remove the $\epsilon$-transition and we obtain the automaton $B$. Note that a new clock $x_0$ and an associated constraint are necessary. This method cannot be used when the $\epsilon$-transition is not followed by a visible action. In order to remove the $\epsilon$-transition in the automaton $C$ we use the backward closure of the constraint $\varphi_1$ and we obtain the automaton $D$.

2.4 Minimization of the number of clocks of a Timed I/O automaton

The number of clocks used in a specification mainly grows for two reasons. First, specifications are often written in high-level description languages and later compiled into timed automata having a number of clocks proportional to the number of time-outs that appear in the description. However, these time-outs are rarely active at the same time, and hence, the number of clocks can be reduced. Second, complex systems are described as the composition of simpler components each having a small number of clocks. It turns out that, due to the synchronization of transitions, many clocks are simultaneously reset and therefore they will be equal for some time since they all proceed at the same speed. Clearly in this case, only one of these clocks is really necessary.

Taking into account these observations, the authors in [Daws 96] propose a method for reducing the number of clocks of a timed automaton by combining two algorithms. The first one consists of detecting active clocks. Intuitively, a clock is active at some state if its value at the state may influence the future evolution of the system. This may happen whenever the clock appears in the invariant condition of the state or is tested in the condition of some of the outgoing edges. The values of inactive clocks at state $s$ are not relevant to the evolution of the system at $s$. This means that the number of clocks required is equal to the greatest number of active clocks in all states. The second algorithm consists of detecting pairs of clocks that are always equal. Two clocks are equal in a state if for every incoming edge they are both reset or are equal in the source location and none is reset.

2.5 Tightening the constraints

Given a TIOA $A$, some transitions will never be fired since the constraints associated with such transitions are always false when we reach the corresponding states. Removing these transitions from $A$ will not change the set $T_T(s)$. There have been numerous works dealing with this problem [Courcoubetis 91][Somé 97]. The region graph [Alur 94a] corresponding to the TIOA $A$ can be used to remove such transitions but the complexity is polynomial in the number of states and edges of $A$ and exponential in the number of clocks and the binary length of the constants that appear in the constraints. In [Somé 97] the author propose an algorithm based on the propagation of clock relations that allows the detection of such transitions. Intuitively, we will find for each state the set of clock interpretations that are satisfied in the state and update the constraints associated with each outgoing transition. Then each transition for which the updated constraint is false will be removed.

2.6 The safe realization relation

**Definition 5:** For a TIOA $A$ and a composite TIOA $B=|B_1||B_2||...||B_n$ with $I_s=I_B$, we say that $B$ realizes $A$ with safety, written $B \leq_s A$, if:

- for every TIOA $E$, with $I_E = O_1$ and $O_E = I_A$, $S(E)|A$ implies $S(I_E)|B_1||B_2||...||B_n$.

Definition 5 means that for any environment $E$ over the alphabet of $A$, if the composition of $A$ and $E$ is safe then the composition of $B_1$, $B_2$, ..., $B_n$ and $E$ must also be safe, i.e. in any reachable state of the composition $E||B_1||B_2||...||B_n$, there is no unspecified reception.

The reflection of a TIOA $A$, denoted $R(A)$ represents the most liberal timed environment in which $A$ is safe, i.e. $R(A)$ must accept all the outputs produced by $A$ and nothing more, and produces all the inputs accepted by $A$. Since $A$ may be non-deterministic, each state $s$ of $R(A)$ must have the following properties:

- the sets of outputs of $s$ must be equal to the intersection of the sets of outputs of all states of $A$ which are reachable by any timed trace leading to $s$.
- the set of inputs of $s$ must be equal to the union of the sets of outputs of all the states of $A$ which are reachable by a timed trace leading to $s$.

For the construction of $R(A)$, we use two functions $RestrictOutputs$ and $AugmentInputs$. The first function takes a
The design of a submodule

3.1 The architecture

We consider the class of systems that can be represented by two TIOA that communicate with one another and with an environment (Figure 2). One TIOA, called the context C, models the known part of the system, the behavior of which is given, while the other deterministic TIOA, called the new component Comp, represents the behavior of a certain component of the system (the submodule to be designed). The set of inputs accepted by the system from the environment can be divided into three disjoint sets. The first, denoted \( I_1 \), is the set of inputs of the context C non-observable by the component. The second, denoted \( I_2 \), is the set of inputs of the context observable by the component. The third, denoted \( I_3 \), represents the set of inputs observable by the component, but not observable by the context.

![Figure 2: The composition of two communicating components C and Comp](image)

Similarly, the set of outputs delivered by the system to the environment can be divided in four disjoint sets. The first, denoted \( O_1 \), is the set of outputs delivered by the context to the environment that are not observable by the new component. The second, denoted \( O_2 \), is the set of outputs delivered by the context to the environment that are observable by the component. The third, denoted \( O_3 \), represents the set of outputs of the component accepted by the environment and not delivered to the context. The fourth, denoted \( O_4 \), is the set of outputs of the component accepted by the environment and observable by the context. In addition, the context and the component can communicate with internal actions not observable by the environment.

3.2 The problem

The problem is known as the problem of submodule construction, redesign or equation solving, where an appropriate conformity criterion should hold between a designed system and its given specification \( A \) and where the system consists of a given component C and a new component X to be designed. In this paper, we consider this problem as a problem of equation solution in the realm of TIOA for the equation \((C)\langle X\rangle \leq A\) under the constraint \( I_X=In \) with \( X \) being a free variable and \( In \) a given set of inputs, which satisfies the constraint \((I_X)\langle O_C\rangle \leq In \subset \langle I_C\rangle\langle O_C\rangle\).
**Step 1: Completing automata and tightening their constraints**

We construct the timed input enabled forms of $A'$ and $C'$, where $A'$ is obtained from $A$ by tightening the constraints and then removing all the invariants and $C'$ is obtained from $C$ by tightening the constraints.

**Step 2: Composition**

We constructed the composed automaton $R := Tief(C') \cup Tief(R(A'))$. Each time we reach a state of $R$ which contains $Fail_C$ or $Fail_{R(A')}$ we replace it by $Fail_{R'}$.

**Step 3: Clock minimization**

We replace all the invariants in $R$ by true then we minimize the number of clocks in $R$.

**Step 4: Eliminating ε-transitions**

We replace in $R$ all the actions which are not in $Inv\sqcap (IC\sqcap OC)$ by the invisible action $ε$. Then we transform $R$ into an equivalent TIOA without ε-transitions. For this purpose, we first remove all the ε-transitions that lead to a silent state- a state without outgoing transitions-, and for each state $p$ where there exists an outgoing transition labeled by an ε-transition we add a new clock $x_p$ that is reset by any transition entering the state $p$. Then,

- for each path $(p_1, ε, ∅, φ_1, p_2) ... (p_k, ε, ∅, φ_k, p_{k+1})$, where $(p_i)_{i≤k+1}$ are distinct states, we add the transition: $(p_1, u, λ, f_C(ε) ... f_C(ε) \sqcap f_C(x_{p_1}(\text{false}) ... x_{p_k}(\text{false}) \sqcap φ_1) ... \sqcap φ_k) + (p_{k+1}, u, λ, φ_k, p_{k+2})$,

- for each path $(p_1, u, λ, φ_1, p_2) ... (p_k, u, λ, φ_k, p_{k+1})$, where $(p_i)_{i≤k+1}$ are distinct states, we add the transition: $(p_1, u, ∅, φ_1 \sqcap f_C(ε) ... f_C(ε) \sqcap f_C(x_{p_1}(\text{false}) ... x_{p_k}(\text{false}) \sqcap φ_1) ... \sqcap φ_k)$.

Finally, we remove all the ε-transitions and we replace $Fail_{R'}$ by $Fail_{R''}$. The obtained automaton is denoted $R_1$.

**Step 5: Eliminating transitions leading to Fail**

In this step we will remove from $R_1$ all the traces that lead to $Fail_{R_1}$. Unlike the untimed case, a transition leading to $Fail_{R_1}$ can be avoided if it is possible to put in the state an invariant that disables this transition by leaving the state before the time of occurrence of the transition.

**Part 1:** In this part, we will use the function $RemoveE$ which takes as input a TIOA and returns, if possible, a TIOA without $E$-transitions; else it returns "NO SOLUTION". An $E$-transition is due to a sequence of interactions -non observable by the component to be designed- between the environment and the context leading to the Fail state. The function $RemoveE$ tries to find an invariant to put in the initial state of the component to be designed such that its earliest interaction with the environment or the context will prohibit the completion of the sequence of interaction leading to the Fail state.

**Part 2:** Since the component to be designed will be obtained in the form of a non deterministic TIOA, the environment and the context cannot precisely observe the state of the component. All states of the component that are reachable by a given timed trace must accept the same set of inputs. For this purpose we use the previously defined function $AugmentInputs$ that will take as input the TIOA obtained at the end of Part 1 and return a TIOA that has the above property.

**Part 3:** Now we are ready to deal with the transitions leading to the Fail state. We use the function $RemoveFail$ that takes as input the TIOA obtained at the end of Part 2 and removes from it all timed traces leading to the Fail state. For each transition $(s, u, ∅, φ_s, Fail_{R_1})$ we propagate this transition, i.e. we transform our automata to obtain one where all the states reachable by a timed trace leading to $s$, have a transition which leads with action $u$ and under constraint $φ_s$ to $Fail_{R_1}$.

If we obtain an automaton at the end of this step we call it $Sol_5$.

4.4 The set of solutions

**Theorem 1:** Given two TIOA $A$ and $C$, and given a set $I_{A}$ such that $(I_{A}\sqcap IC) \sqcap OC \sqsubseteq I_{A} \sqcap OC$, if Algorithm 1 produces a TIOA $Sol_3$ then it is a solution to the submodule construction problem, i.e. $(C||Sol_3) \subseteq A$ for the case $I_{A} = In$. Otherwise there is no solution to this problem.

The solution obtained by Algorithm 1 is generic, which means that we can derive from it all the solutions for the equation $(C||X) \subseteq A$ under the constraint $I_{A} = In$.

**Theorem 2:** Given two TIOA $A$ and $C$, and given a set $I_{A}$ such that $(I_{A}\sqcap IC) \sqcap OC \sqsubseteq I_{A} \sqcap OC$, if Algorithm 1 produces a TIOA $Sol_3$ then for any TIOA $B$, with $I_{B} = In$ and $O_{B}(I_{A}\sqcap IC) \sqcap OC$, the following propositions are equivalent:

- $B \subseteq Sol_3$
- $B$ is a solution of the equation $(C||X) \subseteq A$.

5 Conclusion

We have presented in this paper an approach to solve the problem of submodule construction in the realm of timed I/O automata. This problem may be formulated mathematically by the equation $(C||X) \subseteq A$ under the constraint $I_{A} = In$, where $C$ represents the specification of the known part of the system, $A$ represents the specification of the whole system, $X$ represents the specification of the submodule to be constructed, $||$ is a composition operator, $R$ is a conformance relation and $In$ is the required set of inputs for $X$. The conformance relation considered is the safe realization relation. We consider the case where the non-observable interactions between the context and the environment do not reset clocks. The set of solutions to the equation (if they exist) can be represented as the set of timed safe realizations of a timed I/O automaton $Sol_3$. An algorithm for finding $Sol_3$ is given.

The approach considered in this paper may be extended to treat the problem of submodule construction for other conformance relations, like for example the conforming implantation relation [Drissi 99a], which imposes constraints on the outputs produced in each state of the whole system. Another interesting problem is the case where the non-observable interactions between the context and the environment may reset clocks. We plan also to extend the tool developed for untimed automata [Drissi 99b], by implementing the algorithms for timed automata. Further work will be done in these directions.

**References**


