

Discretization of Timed Automata in Timed μ CRL à la Regions and Zones

Jan Friso Groote Michel A. Reniers Yaroslav S. Usenko

Laboratory for Quality Software, Department of Mathematics and Computer Science,
Technical University of Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

1 Introduction

We present the first step towards combining the best parts of the real-time verification methods based on timed automata (the use of regions and zones), and of the process-algebraic approach of languages like LOTOS and μ CRL. μ CRL targets the specification of system behavior in a process-algebraic (ACP) style and deals with data elements in the form of abstract data types.

A timed automata specification is a parallel composition of timed automata. We use the existing results to translate it to a parallel composition of timed μ CRL processes. This translation uses a very simple sort *Time* to represent the real-time clock values. As a result we obtain a semantically equivalent specification in timed μ CRL.

As the next step in our scheme, we aim at replacing all parameters of sort *Time* occurring in the resulting process equation by parameters of discrete sorts. To achieve this goal we apply process-algebraic transformations and abstraction techniques to the given process equation. As a result we obtain a process equation that is closely related to the given one in the following sense. If we abstract from the fractional parts of the time stamps in the actions, both of the equations will be timed bisimilar.

2 Discretization Steps

2.1 Representing Timed Automata in Timed μ CRL

Timed automata [1, 2] can be represented in timed μ CRL by associating a recursion variable with each location of the automaton as follows (see [6] for the initial idea). Consider a timed automaton $A = \langle L, l^0, \Sigma, C, i, E \rangle$, where L is a finite set of locations, $l^0 \in L$ is the initial location, Σ is a finite set of edge labels, C is a finite set of clocks, i is a mapping that assigns to each location an invariant, and E is a set of edges. An edge is a quintuple $(l, a_e, \phi_e, \lambda_e, l_e)$ with l and $l_e \in L$ the start and end location of the edge, $a_e \in \Sigma$ the label of the edge, ϕ_e the guard associated with the edge and $\lambda_e \subseteq C$ the set of clocks that are to be reset by the transition. All $\phi(e)$ and $i(l)$ are formulas with the following syntax: $c \equiv n \mid c_1 - c_2 \equiv n \mid \phi_1 \wedge \phi_2$, where $\equiv \in \{<, \leq, =, \geq, >\}$ and $n \in \text{Nat}$.

The following timed μ CRL process equation for X_l is a translation of a location $l \in L$ of a timed automaton A :

$$\begin{aligned} X_l(t^a:Time, v:CIVals) = & \\ & \sum_{e \in E_l} \sum_{t^r:Time} a_e \cdot (t^a + t^r) \cdot X_{l_e}(t^a + t^r, v') \\ & \quad \triangleleft sat_inv_l(v) \wedge sat_inv_l(v'') \wedge sat_cond_e(v'') \wedge sat_inv_{l_e}(v') \triangleright \delta \cdot \mathbf{0} \\ & + \sum_{t^r:Time} \delta \cdot (t^a + t^r) \triangleleft sat_inv_l(v) \wedge sat_inv_l(v'') \triangleright \delta \cdot \mathbf{0} \end{aligned}$$

where $t^a:Time$ represents the current *absolute* time; $v:CIVals \subseteq C \rightarrow Time$ represents the current values of the clocks (in *relative* time); $E_l \subseteq E$ is the set of outgoing edges from l with the elements of the form $e = (l, a_e, \phi_e, \lambda_e, l_e)$; $v''(c) = v(c) + t^r$ represents the values of the clocks after time t^r ; $v'(c) = if(c \in \lambda_e, \mathbf{0}, v''(c))$ represents the values of the clocks after time t^r and resetting the clocks from λ_e . The condition $sat_inv_l : CIVals \rightarrow Bool$ is defined as $sat_inv_l(v) = i(l)[\vec{c} := v(\vec{c})]$; and the condition $sat_cond_e : CIVals \rightarrow Bool$ is defined as $sat_cond_e(v) = \phi_e[\vec{c} := v(\vec{c})]$. In these two conditions the values $v(c)$ are substituted for the clock variables c . This substitution is applied to both the location invariant formula $i(l)$ and the guard formula ϕ_e .

The conditions $sat_inv_l(v)$ and $sat_inv_l(v'')$ express that the invariant of location l has to hold in the start state of the transition and in the state just before the edge is taken. Condition $sat_cond_e(v'')$ expresses that the guard of the transition has to be satisfied at the moment the edge is taken, and condition $sat_inv_{l_e}(v')$ means that the invariant of the end location of the edge has to be satisfied (after the clock resets are applied).

2.2 Splitting the Parameters into Integral and Fractional Parts

First we split the parameters t^a and v and the bound variable t^r in two parts: integral and fractional. We make it a bit different from what an obvious split would be as: t^r is the offset since t_i^a , not since t^a . The parameter v is split into v_i and l_f^r that represent an (approximate) integral value and the fractional part of the reset time of the clocks, respectively. To be more precise, this step can be characterized as the following coordinate transformation: $t_i^a = fl(t^a)$, $t_f^a = fr(t^a)$, $v_i = fl(v) + if(fr(t^a) \geq fr(v), \mathbf{0}, 1)$, and $l_f^r = fr(t^a - v)$, where fl and fr are the floor and the fraction functions.

In the other direction: $t^a = t_i^a + t_f^a$ and $v = v_i + t_f^a - l_f^r$. The correspondence between the two t^r is the following: $t^r = t_i^r + t_f^r - t_f^a$, and t_i^r and t_f^r are the integral and the fractional parts of $t^r + fr(t^a)$, respectively. The resulting process will look as:

$$\begin{aligned} X'_l(t_i^a:Nat, t_f^a:Time, v_i:CIValsN, l_f^r:CIVals) = & \\ & \sum_{e \in E_l} \sum_{t_i^r:Nat} \sum_{t_f^r:Time} a_e \cdot (t_i^a + t_i^r + t_f^r) \cdot X'_{l_e}(t_i^a + t_i^r, t_f^r, v'_i, l_f^r) \\ & \quad \triangleleft t_f^r < 1 \wedge (t_f^r \geq t_f^a \vee t_i^r > \mathbf{0}) \wedge sat_inv_l(v) \wedge sat_inv_l(v'') \\ & \quad \quad \quad \wedge sat_cond_e(v'') \wedge sat_inv_{l_e}(v') \triangleright \delta \cdot \mathbf{0} \\ & + \sum_{t_i^r:Nat} \sum_{t_f^r:Time} \delta \cdot (t_i^a + t_i^r + t_f^r) \triangleleft t_f^r < 1 \wedge (t_f^r \geq t_f^a \vee t_i^r > \mathbf{0}) \wedge sat_inv_l(v) \wedge sat_inv_l(v'') \triangleright \delta \cdot \mathbf{0} \end{aligned}$$

where $v_i: CIValsN \subseteq C \rightarrow Nat$ and $l_f^r: CIVals$ are as defined above; $v_i''(c) = v_i(c) + t_i^r$ represents the value of v_i after time t_i^r ; $v_i'(c) = if(c \in \lambda_e, 0, v_i''(c))$ represents the value of v_i after time t_i^r taking into account the clock resets; $l_f^{r'}(c) = if(c \in \lambda_e, t_f^r, l_f^r(c))$ represents the new fractional values of the times the clocks were last reset; $v''(c) = (v_i(c) + t_i^r) + t_f^r - l_f^r(c)$ is the value of $v''(c)$ using the new coordinates, and $v'(c) = if(c \in \lambda_e, \mathbf{0}, v''(c))$ is calculated in the same way as in the previous section.

Given the specific form of the clock constraints and the specific forms of v , v' and v'' , the functions in the conditions can be expressed as the conjunctions of the following formulas (some cases for the function $sat_inv_l(v)$):

- for the case of $c < n$ constraint, substituting the value of $v(c)$ we get $v_i(c) + t_f^a - l_f^r(c) < n$ which is equivalent to $v_i(c) < n \vee (v_i(c) = n \wedge t_f^a < l_f^r(c))$;
- for the case of $c \leq n$ constraint we get $v_i(c) + t_f^a - l_f^r(c) \leq n$ which is equivalent to $v_i(c) < n \vee (v_i(c) = n \wedge t_f^a \leq l_f^r(c))$;
- for the case of $c_1 - c_2 < n$ constraint we get $(v_i(c_1) + t_f^a - l_f^r(c_1)) - (v_i(c_2) + t_f^a - l_f^r(c_2)) < n$ which is equivalent to $(v_i(c_1) - v_i(c_2)) - l_f^r(c_1) + l_f^r(c_2) < n$, or equivalently $(v_i(c_1) - v_i(c_2)) < n \vee ((v_i(c_1) - v_i(c_2)) = n) \wedge l_f^r(c_1) > l_f^r(c_2)$.

For the functions $sat_inv_l(v'')$ and $sat_cond_e(v'')$ we will get similar constraints, with $v_i(c) + t_i^r$ in place of $v_i(c)$ and t_f^r in place of t_f^a (due to the fact that $v''(c) = (v_i(c) + t_i^r) + t_f^r - l_f^r(c)$). For the function $sat_inv_{l_e}(v')$ we apply a similar reasoning.

We claim that $X_l(t^a, v)$ and $X'_l(fl(t^a), fr(t^a), fl(v) + if(fr(t^a) \geq fr(v), 0, 1), fr(t^a - v))$ have the same solutions in every model of timed μ CRL.

2.3 Splitting the Conditions into Integral and Fractional Parts

It is visible from the conditions that the actual values of the real-valued parameters (t_f^a and l_f^r) and the bound variable t_f^r are not important, but the relations between pairs of them may be. Therefore we introduce an abstraction of these parameters and try to use this abstraction instead of the real-valued parameters in the conditions. This corresponds to the use of regions in timed automata ([1]).

Let the set of clocks C be $\{1, \dots, n\}$, and C^0 be $C \cup \{0\}$. Each region will be characterized by an ordering $p_0 <_1 p_1 <_2 \dots <_n p_n$, where $<_k$ is either $<$ or $=$, and p_k is either $l_f^r(p_k)$, or it is t_f^a in case $p_k = 0$, and all p_k are unique. We assume to have such a data type called *Ord* and the functions $is_cond: Ord \times C^0 \times C^0 \rightarrow Bool$ for every possible condition $<, \leq, =, \geq, >$.

It is also important to know the relation between t_f^r and the values of l_f^r . We assume a data type *Pos* to indicate the position of t_f^r in the ordering *ord*. We use the function $fits: Nat \times Pos \times Ord \rightarrow Bool$ to check that the position fits within the given ordering, and if the first parameter is 0, then it checks whether $t_f^r \geq t_f^a$. We can assume that l_f^r and t_f^a conform to *ord* in the initial state of X'_l and prove that it will be an invariant. The condition $conform: Ord \times Pos \times Time \times CIVals \times Time \rightarrow Bool$ says that $conform(ord, pos, t_f^a, l_f^r, t_f^r)$ implies that t_f^r is less than 1 and has indeed the position *pos* in the ordering *ord* w.r.t. t_f^a and l_f^r . The resulting

process X_l'' will look as:

$$\begin{aligned}
X_l''(t_i^a: \text{Nat}, v_i: \text{ClValsN}, \text{ord}: \text{Ord}, t_f^a: \text{Time}, l_f^r: \text{ClVals}) = & \\
\sum_{e \in E_l} \sum_{t_i^r: \text{Nat}} \sum_{\text{pos}: \text{Pos}} \left(\sum_{t_f^r: \text{Time}} a_e^c(t_i^a + t_i^r \boxed{+t_f^r}) \cdot \right. & \\
X_{l_e}''(t_i^a + t_i^r, v_i^r, \text{upd_ord}(\text{ord}, \text{pos}, \lambda_e), t_f^r, l_f^r) \triangleleft \text{conform}(\text{ord}, \text{pos}, t_f^a, l_f^r, t_f^r) \triangleright \delta \cdot \mathbf{0} & \\
\triangleleft \text{fits}(t_i^r, \text{pos}, \text{ord}) \wedge \text{sat_inv}_l^r(v_i, t_i^r, \text{ord}) \wedge \text{sat_inv}_l''(v_i, t_i^r, \text{ord}, \text{pos}) & \\
\wedge \text{sat_cond}_e^r(v_i, t_i^r, \text{ord}, \text{pos}) \wedge \text{sat_inv}_l'''(v_i, t_i^r, \text{ord}, \text{pos}, \lambda_e) \triangleright \delta \cdot \mathbf{0} & \\
+ \sum_{t_i^r: \text{Nat}} \sum_{t_f^r: \text{Time}} \sum_{\text{pos}: \text{Pos}} \left(\sum_{t_f^r: \text{Time}} \delta^c(t_i^a + t_i^r \boxed{+t_f^r}) \triangleleft \text{conform}(\text{ord}, \text{pos}, t_f^a, l_f^r, t_f^r) \triangleright \delta \cdot \mathbf{0} \right) & \\
\triangleleft \text{fits}(t_i^r, \text{pos}, \text{ord}) \wedge \text{sat_inv}_l^r(v_i, t_i^r, \text{ord}) \wedge \text{sat_inv}_l''(v_i, t_i^r, \text{ord}, \text{pos}) \triangleright \delta \cdot \mathbf{0} &
\end{aligned}$$

where $\text{upd_ord}(\text{ord}, \text{pos}, \lambda_e)$ gives the new ordering based on the old one, the position of t_f^r and the clock resets. The order of the clocks that are not reset do not change; the new position of t_f^a and the clocks that are reset will be the position of t_f^r .

The *sat* formulas (some cases for the function $\text{sat_inv}_l(v)$) have the constraints that are defined as follows:

- for the case of $c < n$ constraint: if *ord* implies $t_f^a < l_f^r(c)$, then $v_i(c) \leq n$, else $v_i(c) < n$: thus $\text{if}(\text{is_le}(\text{ord}, 0, c), v_i(c) \leq n, v_i(c) < n)$;
- for the case of $c \leq n$ constraint: $\text{if}(\text{is_leq}(\text{ord}, 0, c), v_i(c) \leq n, v_i(c) < n)$;
- for the case of $c_1 - c_2 < n$: $\text{if}(\text{is_le}(\text{ord}, c_2, c_1), v_i(c_1) - v_i(c_2) \leq n, v_i(c_1) - v_i(c_2) < n)$.

We claim without further proof that $X_l'(t_i^a, t_f^a, v_i, l_f^r)$ and $X_l''(t_i^a, v_i, \text{ord}, t_f^a, l_f^r)$ are timed bisimilar for all parameters *ord* that conform with the actual values of t_f^a and l_f^r .

2.4 Abstraction from Fractional Parts

Suppose we are not interested in the fractional parts of the action and the delta time stamps. E.g. we replace $a_e^c(t_i^a + t_i^r + t_f^r)$ by $a_e^c(t_i^a + t_i^r)$ in X_l'' (we get rid of the boxed parts). The resulting process variable we call Y_l .

Now we apply sum elimination to Y_l (cf. [5]) in order to get rid of the summation with t_f^r and the condition *conform*. For this we use the fact that the *Time* domain is dense and for every $t_i^r, \text{pos}, \text{ord}$ such that $\text{fits}(t_i^r, \text{pos}, \text{ord})$ and for every admissible t_f^a and l_f^r there exists a $t_f^r < 1$ such that $\text{conform}(\text{ord}, \text{pos}, t_f^a, l_f^r, t_f^r)$. As a result we obtain the process equation for Y_l' .

Finally, we apply the parameter elimination to the last two parameters. As a result we get the following process equation for Y_l'' :

$$\begin{aligned}
Y_l''(t_i^a: \text{Nat}, v_i: \text{ClValsN}, \text{ord}: \text{Ord}) = & \\
\sum_{e \in E_l} \sum_{t_i^r: \text{Nat}} \sum_{\text{pos}: \text{Pos}} a_e^c(t_i^a + t_i^r) \cdot Y_{l_e}''(t_i^a + t_i^r, v_i^r, \text{upd_ord}(\text{ord}, \text{pos}, \lambda_e)) & \\
\triangleleft \text{fits}(t_i^r, \text{pos}, \text{ord}) \wedge \text{sat_inv}_l^r(v_i, t_i^r, \text{ord}) \wedge \text{sat_inv}_l''(v_i, t_i^r, \text{ord}, \text{pos}) & \\
\wedge \text{sat_cond}_e^r(v_i, t_i^r, \text{ord}, \text{pos}) \wedge \text{sat_inv}_l'''(v_i, t_i^r, \text{ord}, \text{pos}, \lambda_e) \triangleright \delta \cdot \mathbf{0} & \\
+ \sum_{t_i^r: \text{Nat}} \sum_{\text{pos}: \text{Pos}} \delta^c(t_i^a + t_i^r) & \\
\triangleleft \text{fits}(t_i^r, \text{pos}, \text{ord}) \wedge \text{sat_inv}_l^r(v_i, t_i^r, \text{ord}) \wedge \text{sat_inv}_l''(v_i, t_i^r, \text{ord}, \text{pos}) \triangleright \delta \cdot \mathbf{0} &
\end{aligned}$$

The transformations we apply here are known to be standard for μCRL equations [4]. We claim without further proof that Y_l , Y'_l and Y''_l are timed bisimilar.

3 Conclusions and Future Work

In this paper we transformed a timed μCRL process equation representing a timed automaton into a closely related timed μCRL process equation with discrete parameters and bound variables only. This could enable simulation and verification via enumeration of reachable states. As a result, some of the existing untimed analysis tools in the μCRL Toolset [3] could become applicable to the analysis of real-time systems.

As the future step in our scheme we would like to make the parameters and the bound variables finite. To this end we apply a *relativization* technique to get rid of the absolute time parameter t_i^a . Due to the presence of the greatest constant in timed automata we can apply the abstract interpretation technique to limit both the integer values of the clocks v_i and the integer relative time step t_i^r .

As the next step we would like to factorize the remaining time-related parameters to be able to deal with them like with zones. Both regions and zones, as well as the operations on them could be specified as the abstract data types *Region* or *Zone* in μCRL , either as clock constraints or as difference-bound matrices. We could even go further, analyze where exactly we use the fact that we are dealing with timed automata and extend some of the results to a more general setting.

References

- [1] R. Alur. Timed automata. In *Proc. CAV'99*, LNCS 1633, pages 8–22, 1999.
- [2] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. Technical Report 316, UNU-IIST, P.O.Box 3058, Macau, September 2004.
- [3] S. Blom, W. J. Fokkink, J. F. Groote, I. A. v. Langevelde, B. Lissner, and J. C. v. d. Pol. μCRL : A toolset for analysing algebraic specifications. In G. Berry, H. Comon, and A. Finkel, editors, *Proc CAV'01*, volume 2102 of *LNCS*, pages 250–254. Springer, 2001.
- [4] J. F. Groote and B. Lissner. Computer assisted manipulation of algebraic process specifications. *SIGPLAN Notices*, 37(12):98–107, 2002.
- [5] J. F. Groote and M. A. Reniers. Algebraic process verification. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, chapter 17, pages 1151–1208. Elsevier, 2001.
- [6] T. A. C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing*. PhD thesis, Eindhoven University of Technology, 2003.