Enhancing the Inverse Method with State Merging

Étienne André[†], Laurent Fribourg[‡], Romain Soulat[‡]

[†]LIPN, CNRS UMR 7030, Université Paris 13, France
[‡]LSV, ENS Cachan & CNRS, France

Abstract. Keeping the state space small is essential when verifying real-time systems using Timed Automata (TA). In the model-checker Uppaal, the merging operation has been used extensively in order to reduce the number of states. Actually, Uppaal's merging technique applies within the more general setting of Parametric Timed Automata (PTA). The *Inverse Method* (*IM*) for a PTA $\mathcal{A}$ is a procedure that synthesizes a zone around a given point π^0 (parameter valuation) over which $\mathcal{A}$ is guaranteed to behave in an equivalent time-abstract manner. We show that the integration of merging into *IM* leads to the synthesis of larger zones around π^0. It also often improves the performance of *IM*, both in terms of computational space and time, as shown by our experimental results.

1 Introduction

A fundamental problem in the exploration of the reachability space in Timed Automata (TA) is to compact as much as possible the generated space of symbolic states. In [11], the authors show that, in a network of TAs, all the successor states can be merged together when all the interleavings of actions are possible. In [7, 8], A. David proposed to replace the union of two states by a unique state when this union is convex. More precisely, if the union of two states is included into their convex hull, then one can replace the two states by their hull. This technique is applied to timed constraints represented under the form of "Difference Bound Matrices" (DBMs). Actually, such a merging technique applies as well in the more general setting of *parametric* timed automata (PTA), where parameters can be used instead of constants, and timed constraints are represented under the form of polyhedra.

The *Inverse Method* (*IM*) for a PTA $\mathcal{A}$ is a procedure that synthesizes a zone around a given point π^0 (parameter valuation) over which $\mathcal{A}$ is guaranteed to behave in an equivalent time-abstract manner [2]. We show that the integration of merging into *IM* often leads to the synthesis of larger zones around π^0. More surprisingly, our experiments show that even a simple implementation of merging often improves the performance of *IM*, not only in terms of computational space but also in time.

2 Background and Definition

2.1 Timed Automata

Given a finite set X of n non-negative real-valued variables (called "clocks"), a *timed constraint* is a conjunction of linear inequalities of the form $x_i \prec c$, $-x_i \prec c$ or $x_i - x_j \prec c$ with $\prec \in \{<, \leq\}$, $x_i, x_j \in X$ and $c \in \mathbb{Z}$.

A Timed Automaton (TA) is a tuple $(\Sigma, Q, l_0, X, I, \rightarrow)$, with Σ a finite set of *actions*, Q a finite set of *locations*, $l_0 \in Q$ the *initial location*, X a set of *clocks*, I the *invariant* assigning to every $l \in Q$ a constraint over X, and $\rightarrow$ a *step relation* consisting of elements (l, g, a, ρ, l'), where $l, l' \in Q$, $a \in \Sigma$, g is a timed constraint (*guard*) and ρ is a subset of X (set of clocks reset to 0).

A *state* is a couple (l, v) where l is a location of Q and v a valuation of X.

The operational semantics of TA is informally given as follows: given two states $s = (l, v)$ and $s' = (l', v')$ with $l, l' \in Q$, v, v' two valuations of X, the step $s \xrightarrow{a} s'$ means that, for some $(l, g, a, \rho, l') \in \rightarrow$ and some $\delta \in \mathbb{R}_+$:

$$(l, v) \overset{g,a,\rho}{\Rightarrow} (l', v') \xrightarrow{\delta} (l', v' + \delta),$$

where $(l, v) \overset{g,a,\rho}{\Rightarrow} (l', v')$ means that discrete transition $(l, g, a, \rho 0, l')$ can take place (i.e. v satisfies g, and v' is obtained from v by resetting the clocks of ρ to zero), and $(l', v') \xrightarrow{\delta} (l', v' + \delta)$ means that time can pass during δ units in location l' (i.e., $v' + \delta'$ satisfies the invariant $I(l')$ for all $0 \leq \delta' \leq \delta$).

A *run* is a sequence of the form $(l_0, v_0) \xrightarrow{a_1} (l_1, v_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (l_n, v_n)$. A *trace* (or time-abstracted run) associated to a run is a sequence of the form $l_0 \overset{a_1}{\Rightarrow} l_1 \overset{a_2}{\Rightarrow} \cdots \overset{a_n}{\Rightarrow} l_n$. A trace can be seen as an alternating sequence of locations and actions. Given a TA $\mathcal{A}$, we denote by $Tr(\mathcal{A})$ the set of traces associated to all possible runs of $\mathcal{A}$. When two TAs have the same set of traces, we say that they behave in an equivalent time-abstract manner.

Given a set of states S, one defines $Post_{\mathcal{A}}(S)$ as the set of states reachable from S in one step, i.e.:

$$\begin{aligned} Post_{\mathcal{A}}(S) = \{s' = (l', v') \in Q \times \mathbb{R}_+^n \mid s = (l, v) \xrightarrow{a} s', \\ \text{for some } s \in S,\ l \in Q,\ v \in \mathbb{R}_+^n \text{ and } (l, g, a, \rho, l') \in \rightarrow\}. \end{aligned}$$

Likewise, $Post^i_{\mathcal{A}}(S)$ is the set of states reachable from S in exactly i steps and let $Post^*_{\mathcal{A}}(S) = \bigcup_{i \geq 0} Post^i_{\mathcal{A}}(S)$.

2.2 Parametric Timed Automata

We assume now given a finite set P of symbols (called "parameters"). A *parametric term* is a linear combination of parameters and integer constants. A *parametric timed constraint* is a conjunction of linear inequalities of the form $x_i \prec e$, $-x_i \prec e$ or $x_i - x_j \prec e$ where $\prec \in \{<, \leq\}$, $x_i, x_j \in X$ and e is a parametric term. A *constraint over* P is a conjunction of inequalities of the form $e_1 \prec e_2$ with $\prec \in \{<, \leq\}$ and e_1, e_2 two parametric terms. Given a parametric timed constraint C, the expression $(\exists X : C)$ denotes the constraint over P obtained

from C by eliminating the variables of X. Given a parametric timed constraint C and a valuation π over P (i.e. a function from P to $\mathbb{N}$), we denote by $C[\pi]$ the result of replacing every parameter in C by its π-valuation. We write $\pi \models C$ to express that $\exists X : C[\pi]$ is true. A *(symbolic parametric) state* is a couple (l, C) where l is in Q and C is a parametric timed constraint. A Parametric Timed Automaton (PTA) is a TA where some constants appearing in the guard and invariant inequalities have been replaced by parameters. Given a PTA $\mathcal{A}$, we denote by $\mathcal{A}[\pi]$ the TA obtained from $\mathcal{A}$ by replacing the parameters by their π-valuations. Given a parametric constraint K, we denote by $\mathcal{A}(K)$ the PTA where the parameters are assumed to satisfy K.

2.3 Inverse Method

Given a PTA $\mathcal{A}$ and a valuation π^0 over P, the goal of *IM* introduced in [2] is to synthesize a constraint K^0 over P such that: $\pi^0 \models K^0$ and $Tr(\mathcal{A}[\pi^0]) = Tr(\mathcal{A}[\pi])$, for all $\pi \models K^0$. This implies that for every $\pi \models K^0$, $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi^0]$ have the same time-abstracted behavior. The size of K^0 gives us a measure of the "robustness" (see [10]) of the behavior of $\mathcal{A}$ around π^0. The larger K^0 is, the more robust $\mathcal{A}$ is guaranteed to be. The algorithm *IM* is given below (where s_0 denotes the set of states of location l_0 whose clocks are equal and satisfy the invariant $I(l_0)$). The idea of the procedure is to refine iteratively a current constraint K over P by adding inequalities J in order to eliminate all the generated π^0-incompatible states (i.e., states (l, C) such that $\pi^0 \not\models (\exists X : C)$).

Algorithm 1: Algorithm $IM(\mathcal{A}, \pi^0)$

input : PTA $\mathcal{A}$ of initial state s_0, parameter valuation π^0
output: Constraint K^0 on the parameters

$i \leftarrow 0$; $K \leftarrow \texttt{true}$; $S \leftarrow \{s_0\}$
while `true` **do**
  while *there are π^0-incompatible states in S* **do**
    Select a π^0-incompatible state (l, C) of S (i.e., s.t. $\pi^0 \not\models (\exists X : C)$) ;
    Select a π^0-incompatible inequality J in $(\exists X : C)$ (i.e., s.t. $\pi^0 \not\models J$) ;
    $K \leftarrow K \wedge \neg J$; $S \leftarrow \bigcup_{j=0}^{i} Post^j_{\mathcal{A}(K)}(\{s_0\})$;
  if $Post_{\mathcal{A}(K)}(S) \sqsubseteq S$ **then return** $K^0 = \bigcap_{(l,C)\in S}(\exists X : C)$
  $i \leftarrow i + 1$; $S \leftarrow S \cup Post_{\mathcal{A}(K)}(S)$; // $S = \bigcup_{j=0}^{i} Post^j_{\mathcal{A}(K)}(\{s_0\})$

3 Enhancement of *IM* with Merging

Let us recall the notion of merging, following the lines of [7].

Definition 1 (Merging). *We say that two states $s = (l, C)$ and $s' = (l', C')$ are* mergeable *iff $l = l'$ and $C \cup C'$ is convex; then, $(l, C \cup C')$ is their* merging.

In [7], the main technique for merging two timed constraints C, C' consists of comparing their convex hull H with their union. If the hull and the union are equal (or alternatively, if $(H \setminus C) \setminus C' = \emptyset$ where $\setminus$ is the operation of *convex difference*), then C and C' are mergeable into H. In [7, 8], this technique is specialized to the case where the timed constraints are represented as DBMs. Actually, as mentioned in the introduction, such a merging technique based on convex difference is more general and still applies in the setting of PTA, where parametric timed constraints are represented under the form of polyhedra.

Given a set of (symbolic parametric) states S, let $Merging(S)$ denote the result of applying iteratively the merging of a pair of states of S (using convex difference) until no further merging applies. We define $Post_{merge}(S)$ as $Merging(Post(S))$. Let us denote by IM_{merge} the algorithm obtained from IM (see algorithm 1) by replacing the $Post$ operator by $Post_{merge}$. Let $K^0_{merge} = IM_{merge}(\mathcal{A}, \pi^0)$ and $K^0 = IM(\mathcal{A}, \pi^0)$. It is easy to see that we have always

$$K^0 \subseteq K^0_{merge}.$$

Informally, this is because the merging of a π^0-incompatible state with a π^0-compatible state gives a π^0-compatible state. Therefore, there are less π^0-incompatible states generated. Accordingly, the current set K is less often refined with inequalities J in IM_{merge}. The property of trace preservation still holds with IM_{merge}:

Proposition 1. *Given a PTA $\mathcal{A}$ and a valuation π^0, let $K^0_{merge} = IM_{merge}(\mathcal{A}, \pi^0)$. We have: $\pi^0 \models K^0_{merge}$; furthermore: $\forall \pi \models K^0_{merge}$, $Tr(\mathcal{A}[\pi]) = Tr(\mathcal{A}[\pi^0])$.*

The proof of this proposition is similar to its counterpart in [2].

We have implemented IM_{merge} through a simple extension of the tool IMITATOR [4] using the operation of convex difference on polyhedra from the Parma Polyhedra Library (PPL) [6]. We give in Table 1 some experimental results obtained with IM_{merge} compared with those obtained with IM. The experiments have been done on a 2.4 GHz Intel single-core processor with 4 GB of RAM memory.

PTA	X	P	IM				IM_{merge}				$K^0 \subseteq K^0_{merge}$
			t	States	Trans.	M	t	States	Trans.	M	
AndOr	4	12	0.112	16	17	1,262	0.101	13	14	1,187	=
Flip-Flop	5	12	0.183	14	13	1,692	0.227	14	13	1,762	=
Latch	8	13	1.18	18	68	3,686	0.621	12	40	2,662	$\subsetneq$
BRP	7	6	4.29	428	474	25,483	7.015	426	473	25,845	=
WLAN	2	8	220.157	7,038	11,052	733,044	286.141	6,020	9,538	1,408,702	=
SPSMALL_1	10	26	1.578	31	35	5,098	1.642	31	35	5,442	=
SPSMALL_2	28	62	-	-	-	overflow	593	397	499	180,888	-
SIMOP	8	7	18.959	1,108	1,404	43,333	5.179	239	347	14,371	$\subsetneq$
CSMA/CD	3	3	0.801	240	383	6,580	0.947	240	383	7,049	=
Jobshop	3	8	1.865	253	387	10,658	1.147	118	179	5,221	$\subsetneq$
Mutex 3	3	2	0.802	307	1,060	14,598	0.671	241	811	11,934	=
Mutex 4	4	2	22.373	4,769	19,873	373,900	22.03	3,287	13,459	260,962	=

Table 1. Comparison between IM and IM_{merge}

The models of PTA in Table 1 are described in [5], except Jobshop which corresponds to the jobshop scheduling problem with 2 jobs and 4 tasks of [1] (Table 1), and the Mutex i model ($i = 3, 4$) which corresponds to Fisher's mutual

exclusion protocol with i tasks of [9]. In Table 1, column X (resp. P) denotes the number of clocks (resp. parameters) of the PTA. Column t (resp. M) denotes the computational time in seconds (resp. the memory used in KB), column States (resp. Trans.) the number of states (resp. transitions) of the generated reachability graph. The last column indicates if $K^0 = K^0_{merge}$ or $K^0 \subsetneq K^0_{merge}$.

We can see that K^0_{merge} is strictly larger than K^0 on 3 examples. Furthermore, the reachability graphs produced with IM_{merge} are always smaller than the corresponding graphs produced with IM (as illustrated in Appendix). Let us also point out that IM_{merge}, unlike IM, is able to treat the SPSMALL_2 example (which contains no less than 62 parameters). Finally, the experiments are often faster with IM_{merge}, in spite of the simplicity of our implementation.

4 Final Remarks

We have shown that the integration of a general technique of state merging into IM often increases the size of the synthesized constraint while reducing the computation space. Surprisingly, in spite of our simple implementation of merging, the extended procedure is often faster than the basic procedure on our experiments. We presently study the combined integration into IM of the general technique of state merging with specific improvements presented in [3].

Acknowledgment. We are grateful to T. Chatain for helpful discussions.

References

1. Y. Abdeddaim and O. Maler. Job-shop scheduling using timed automata. In *CAV'01*, volume 2102 of *LNCS*, pages 478–492. Springer Berlin / Heidelberg, 2001.
2. É. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *IJFCS*, 20(5):819–836, 2009.
3. É. André and R. Soulat. Synthesis of timing parameters satisfying safety properties. In *RP'11*, volume 6945 of *LNCS*, pages 31–44, Italy, 2011. Springer.
4. Étienne André. IMITATOR II: A tool for solving the good parameters problem in timed automata. In *INFINITY*, pages 91–99, 2010.
5. Étienne André. *An Inverse Method for the Synthesis of Timing Parameters in Concurrent Systems*. Thèse de doctorat, ENS Cachan, France, 2010.
6. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
7. Alexandre David. Merging DBMs efficiently. In *17th Nordic Workshop on Programming Theory*, pages 54–56. DIKU, University of Copenhagen, 2005.
8. Alexandre David. Uppaal DBM Library Programmer's Reference. `http://people.cs.aau.dk/~adavid/UDBM/manual-061023.pdf`, 2006.
9. T.A. Henzinger, P.H. Ho, and H. Wong-Toi. A user guide to HyTech. In *TACAS'95*, pages 41–71, 1995.
10. Nicolas Markey. Robustness in real-time systems. In *SIES'11*, pages 28–34, Sweden, 2011. IEEE Computer Society Press.
11. R. Ben Salah, M. Bozga, and O. Maler. On interleaving in timed automata. In *CONCUR '06*, volume 4137 of *LNCS*, pages 465–476. Springer, 2006.

Appendix: Compared Reachability Graphs for Jobshop and SIMOP examples

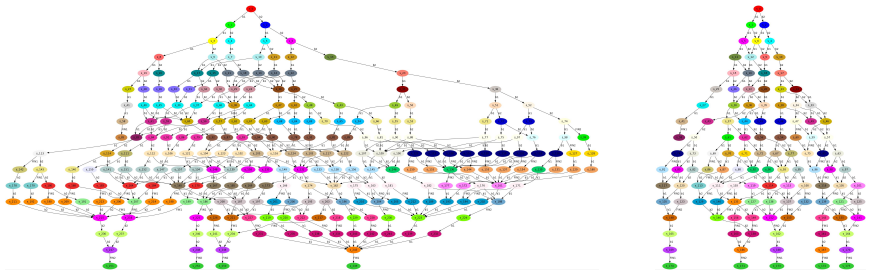

Fig. 1. Reachability graph of the jobshop example with IM (left) and IM_{merge} (right)

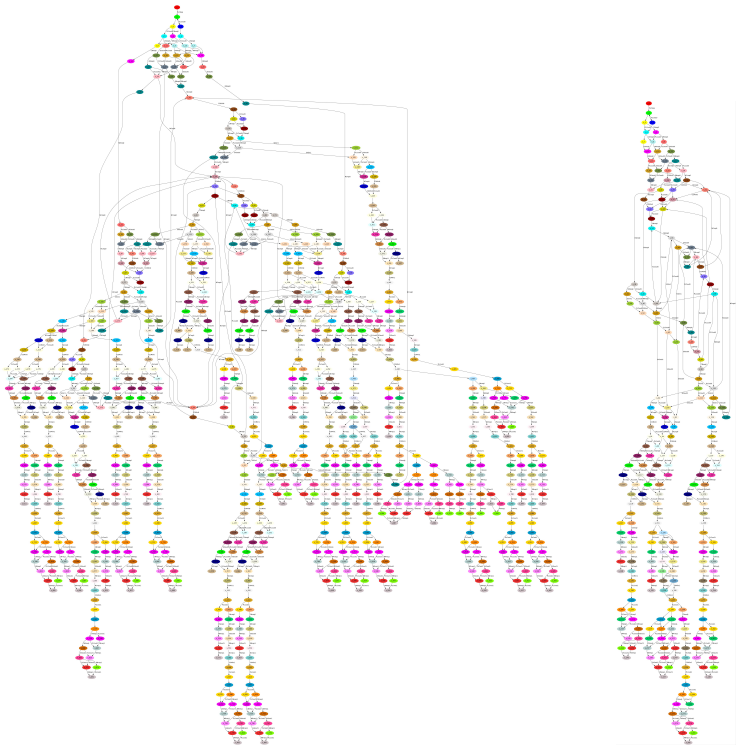

Fig. 2. Reachability graph of the SIMOP example with IM (left) and IM_{merge} (right)