Analysis of slope-parametric hybrid automata

Frédéric Boniol[1], Augusto Burgueño[1] ⋆, Olivier Roux[2] ⋆⋆, and Vlad Rusu[2]

[1] ONERA-CERT, Département d'Informatique,
2 av. E. Belin BP4025, 31055 Toulouse Cedex 4, France.
[2] LAN (URA CNRS N. 823, Ecole Centrale de Nantes, Université de Nantes),
1 rue de la Noë - BP92101 44321 Nantes Cedex 3, France.

Abstract. This paper addresses the analysis of slope-parametric hybrid automata: finding conditions on the slopes of the automaton variables, for some safety property to be verified. The problem is shown decidable in some practical situations (e.g. finding the running speeds of tasks in a real time application, for all tasks to respect their deadlines). The resolution technique generalizes polyhedral-based symbolic analysis and it involves reasoning about polyhedra with parametric shapes.

Keywords: real-time system, hybrid automata, parametric polyhedra.

1 Introduction

The parametric analysis of a real-time system consists in computing some parameters of the system to insure its 'correct' behaviour. Among the system parameters, some important ones are the running speeds of the tasks that compose it. We define here Slope-Parametric Hybrid Automata to model real-time systems whose running speeds should be adjusted for meeting some safety requirement. Then, parametric analysis consists in finding conditions on the slopes of the automaton variables, for some given property to be true. This is a new approach, since (up to our knowledge) only delay-parametric analysis has been studied and implemented (cf. [AHV93] and the HyTech tool [HH94]).

We describe a procedure which performs parametric analysis on slope-parametric hybrid automata. The procedure does not terminate in general; however, in the case of *time-bounded reachability* properties and *uniformly low-bounded* slope-parametric hybrid automata, the procedure terminates and the obtained conditions are necessary and sufficient. We believe that these automata are powerful enough to model many real-time applications. This leads for instance to a (theoretical) means for finding the running speeds, for all tasks in an application to respect their deadlines. While costly in practice, we think that the main interest of the procedure lies in setting a first solution, and in extending the usual polyhedral calculus [Zie95] with computation on parametric-shaped polyhedra.

⋆ Supported by Research Grant of the Spanish Ministry of Education. This research was carried out while the author was visiting the LAN.

⋆⋆ Corresponding author. Email: roux@lan.ec-nantes.fr. Institut Universitaire de France.

2 Slope-Parametric Hybrid Automata

Definition. Slope-Parametric Hybrid Automata are a generalization of Multirate Automata [ACH+95]. A multirate automaton is a hybrid automaton whose continuous evolution laws are defined by differential equations of the form $\dot{x} = c$ where x is a variable ranging over $\mathbb{R}$ and c is a constant in $\mathbb{R}$ (the slope). Thus, for a multirate automaton the slopes are constant. A slope-parametric hybrid automaton allows evolution laws of the form $\dot{x} = k$, where k is a parameter in $\mathbb{R}$: the slopes may be either constants or parameters. Invariants and transition conditions of slope-parametric hybrid automata are expressed in the usual way ($\bigwedge_i x_i \sim c_i$, where x_i is a variable, c_i is a constant in $\mathbb{R}$ and $\sim \in \{<, \leq, >, \geq\}$) and the only allowed updates in variables when crossing an edge are reinitializations to zero.

Example. The slope-parametric hybrid automaton of figure 1 has two vertices, L_1 and L_2, and two variables, x and y. For vertex L_1 the evolution law of x is parametric ($\dot{x} = k_1$) while the one for y is constant ($\dot{y} = 1$). Same case for vertex L_2 with a different parameter for x ($\dot{x} = k_2$). The guard of the edge from vertex L_1 to vertex L_2 is ($x \geq 1 \wedge x \leq 2 \wedge y \geq 3 \wedge y \leq 8$). Only one reinitialization takes place on the edge from L_1 to L_2 ($x := 0$). The invariant for both vertices is *true* (not shown in the figure).

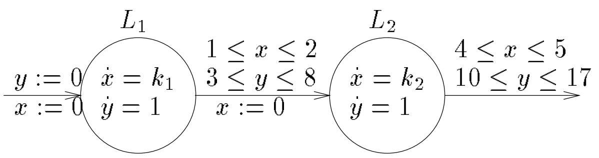

Figure 1. A slope-parametric hybrid automaton

Parametric analysis. The problem of parametric analysis is formulated as follows: given a *slope-parametric hybrid automaton*, a *temporal logic formula expressing a reachability property* and a *set of intervals in* $\mathbb{R}$ (one for each parameter), find the relations among parameters (in the corresponding intervals) such that the formula is true. For instance, in the automaton of figure 1, find values for parameters $k_1 \in [-10, 10], k_2 \in [0, 5]$, for the TCTL formula $\varphi = [L_1 \wedge x = 0 \wedge y = 0] \wedge \exists true \mathcal{U} [L_2 \wedge x \geq 4 \wedge x \leq 5 \wedge y \geq 10 \wedge y \leq 17]$ to be true. This formula means that, starting from the initial set of states defined by vertex L_1 with $x = 0$ and $y = 0$, it is possible to reach the final set of states, defined by vertex L_2 with $x \in [4, 5]$ and $y \in [10, 17]$.

3 Computing parameter values

Parametric analysis comes to operating with parametric polyhedra. The latter are described by sets of linear equations and inequations whose coefficients can be either *constants* or *symbolic expressions on the parameters*; for instance, $x \geq 0 \wedge y \geq 0 \wedge k_1 \cdot y - x \geq 0 \wedge x - k_1 \cdot y \geq 0$, where k_1 is a parameter. The operations involved in parametric analysis are *extension*, *restriction* and *projection*, that incrementally generate conditions on the parameters, at each transition from a vertex to the next one. We first demonstrate their use on an example and then describe them in more detail.

Example. Consider the slope-parametric hybrid automaton in figure 1 and the problem of finding values for parameters $k_1 \in [-10, 10], k_2 \in [0, 5]$, for the formula $\varphi = [L_1 \wedge x = 0 \wedge y = 0] \wedge \exists true \mathcal{U} [L_2 \wedge x \geq 4 \wedge x \leq 5 \wedge y \geq 10 \wedge y \leq 17]$ to be true. For this, we *extend* the initial region $x = 0 \wedge y = 0$ at vertex L_1, in the parametric direction given by $\dot{x} = k_1, \dot{y} = 1$, and then *restrict* the values of parameter k_1 such that the previous extension intersects the guard $1 \leq x \leq 2$, $3 \leq y \leq 8$. Next, we *project* the obtained intersection on plane $x = 0$ (this corresponds to reinitializing x on the transition). Likewise, we continue by *extending* the obtained projection in the direction given by $\dot{x} = k_2, \dot{y} = 1$ at vertex L_2, and terminate by *restricting* the values of parameters k_1, k_2 such that the last extension intersects the final region $x \geq 4 \wedge x \leq 5 \wedge y \geq 10 \wedge y \leq 17$.
Extension and projection are operations on parametric polyhedra, and restriction means finding the values of parameters such that a given parametric polyhedron is non-void.
Extension means: given a parametric polyhedron $P = \bigwedge_{j=1}^{m} (\Sigma_{i=1}^{n} a_{i,j} \cdot x_i \succ b_j)$ where $a_{i,j}, b_j$ can be constants or symbolic expressions and $\succ \in \{>, \geq\}$, find the polyhedron: $\overrightarrow{P} = \exists t \geq 0. \left[\bigwedge_{j=1}^{m} (\Sigma_{i=1}^{n} a_{i,j} \cdot (x_i - k_i \cdot t) \succ b_j) \right]$ where each k_i is the slope for variable x_i (constant or parameter). This is *forward continuous simulation* [ACH+95] except that we consider parametric polyhedra and directions. This imposes to consider several cases for eliminating the $\exists$ quantifier in the equivalent expression $\overrightarrow{P} = \exists t \geq 0. \left[\bigwedge_{j=1}^{m} (\Sigma_{i=1}^{n} a_{i,j} \cdot x_i - t \cdot (\Sigma_{i=1}^{n} a_{i,j} \cdot k_i) \succ b_j) \right]$. The cases to consider are, for each sum-of-products $\Sigma_{i=1}^{n} a_{i,j} \cdot k_i$, the possibility that it is negative, positive or 0. As in [AHH93] we eliminate the existential quantifier by dropping the inequations that correspond to negative sums-of-products ($\Sigma_{i=1}^{n} a_{i,j} \cdot k_i < 0$), keeping the inequations that correspond to positive or zero sums-of-products ($\Sigma_{i=1}^{n} a_{i,j} \cdot k_i \geq 0$), and linearly combining pairs of inequations that correspond to one negative and one positive sum-of-products.
The point here is that the sums-of-products are symbolic, so in general we will not be able to tell at sight the sign of the expressions $\Sigma_{i=1}^{n} a_{i,j} \cdot k_i$. We must then consider all the possible cases for these signs, so the extended polyhedron $\overrightarrow{P}$ has at most 3^m different forms, following the possible signs of the m expressions $\Sigma_{i=1}^{n} a_{i,j} \cdot k_i$. For the example of figure 1, vertex L_1, the extension of the region $P = \{x = 0 \wedge y = 0\}$ gives three cases: 1) for $k_1 = 0, \overrightarrow{P_1} = \{x = 0 \wedge y \geq 0\}$,

2) for $k_1 > 0$, $\vec{P}_2 = \{x \geq 0 \wedge y \geq 0 \wedge k_1 \cdot y - x \geq 0 \wedge x - k_1 \cdot y \geq 0\}$, and 3) for $k_1 < 0, \vec{P}_3 = \{-x \geq 0 \wedge y \geq 0 \wedge -k_1 \cdot y + x \geq 0 \wedge -x + k_1 \cdot y \geq 0\}$. The possible signs of sums-of-products (like $k_1 > 0$, $k_1 = 0$, $k_1 < 0$) constitute relations on the parameters (slopes) that should be remembered when further propagating the extended polyhedron $\vec{P}$. Note also that the parametric slopes have become coefficients in the inequations of the extended polyhedron (i.e. $k_1 \cdot y - x \geq 0$).

Restriction is: given a parametric polyhedron $P = \bigwedge_{j=1}^{m} (\Sigma_{i=1}^{n} a_{i,j} \cdot x_i \succ b_j)$ (remember that $a_{i,j}, b_j$ are constants or symbolic expressions on the parameters) find the possible values of parameters such that P is non-empty. Here is a simple example of restriction (figure 1, vertex L_1): for the three parametric polyhedra obtained at previous step (one for each case $k_1 = 0, k_1 > 0$ and $k_1 < 0$) we compute their intersection with the guard $\{1 \leq x \leq 2 \wedge 3 \leq y \leq 8\}$ and find the values of k_1 such that the intersection is non-empty. For 1) $k_1 = 0$ it is empty: $\vec{P}_1 \cap \{1 \leq x \leq 2 \wedge 3 \leq y \leq 8\} = \emptyset$. For 2) $k_1 > 0$, $\vec{P}_2 \cap \{1 \leq x \leq 2 \wedge 3 \leq y \leq 8\}$ is non-empty iff $\frac{1}{8} \leq k_1 \leq \frac{2}{3}$. For 3) $k_1 < 0$ the intersection is always empty. The general case can be treated as follows: the polyhedron P is non-empty iff the expression $\exists x_1.\exists x_2 \ldots \exists x_n.P$ is 'true'. The formal elimination of all variables in the previous expression generates a symbolic condition on the parameters, *that precisely constitutes the condition for P to be non-empty.* Variables $x_1 \ldots x_n$ can be eliminated one by one by successively applying the Fourier-Motzkin elimination algorithm [Zie95] that we now describe.

The Fourier-Motzkin elimination algorithm. This algorithm computes, given a system of linear inequations $P = \bigwedge_{j=1}^{m} (\sum_{i=1}^{n} a_{i,j} x_i \succ b_j)$, the system obtained by eliminating a variable say x_k : $P \updownarrow_k = \exists x_k. \bigwedge_{j=1}^{m} (\sum_{i=1}^{n} a_{i,j} x_i \succ b_j)$. The idea is to consider the possible signs of the coefficients $a_{k,j}$: as in the case of extension, eliminating variable x_k leads to at most 3^m possible forms of the result, depending on the signs of the m coefficients $a_{k,j}$. For a given combination of signs, denote $J_>$ (respectively, $J_=$, $J_<$) the subsets of indices of $\{1, \ldots, m\}$ such that $a_{k,j} > 0$ (respectively $= 0$, < 0). Then, $P \updownarrow_k$ is obtained by keeping the inequations indexed by $J_=$, by eliminating the inequations indexed by $J_<$ and $J_>$ (i.e. keep only the inequations where x_k does not occur), and by linearly combining pairs of inequations (one indexed by some $j_> \in J_>$, the other indexed by some $j_< \in J_<$) to eliminate variable x_k. For all $j_> \in J_>$ and $j_< \in J_<$, generate the following linear combination: $a_{k,j_>} [\sum_{i=1}^{n} a_{i,j_<} x_i \succ b_{j_<}] - a_{k,j_<} [\sum_{i=1}^{n} a_{i,j_>} x_i \succ b_{j_>}]$ which is equivalent to $\sum_{i=1}^{n} [a_{k,j_>} a_{i,j_<} - a_{k,j_<} a_{i,j_>}] x_i \succ [a_{k,j_>} b_{j_<} - a_{k,j_<} b_{j_>}]$. In this last inequation, the coefficient of x_k is 0 so variable x_k has been eliminated.

Projection on the $x_k = 0$ plane means: given a parametric polyhedron $P = \bigwedge_{j=1}^{m} (\Sigma_{i=1}^{n} a_{i,j} \cdot x_i \succ b_j)$, find the parametric polyhedron $P|_{x_k=0}$ obtained by projecting P on the plane $x_k = 0$. Consider again the example of figure 1. In the previous step (restriction) we had obtained only one valid case ($k_1 > 0$) for which the parametric polyhedron is non-empty: $\vec{P}_2 = \{x \geq 1 \wedge x \leq 2 \wedge y \geq 3 \wedge y \leq 8$

$\wedge k_1 \cdot y - x \geq 0 \wedge x - k_1 \cdot y \geq 0\}$ when $\frac{1}{8} \leq k_1 \leq \frac{2}{3}$. The projection of $\vec{P_2}$ on the plane $x = 0$ is $\vec{P_2}\,|_{x=0} = \{x \geq 0 \wedge x \leq 0 \wedge y \geq 3 \wedge y \leq 8 \wedge k_1 \cdot y \geq 1 \wedge k_1 \cdot y \leq 2\}$. In general, to obtain the projected parametric polyhedron we use the Fourier-Motzkin elimination algorithm and the identity $P|_{x_k=0} = \exists x_k.P \wedge (x_k = 0)$.

The operations at work. To obtain the conditions on the parameters for a reachability formula to be true, combine the three operations as follows. First, choose a vertex path that links a vertex from the initial region, to a vertex from the final region (as defined by the reachability formula). Then, iterate the three operations on that vertex path, to generate a *tree* whose nodes are pairs (vertex, parametric polyhedron), and whose edges are labeled by symbolic conditions on the parameters. Starting from the pair (initial vertex, polyhedron defined by the initial values of variables) as the root, apply the *extension* procedure to the initial polyhedron to generate all the possible extended polyhedra, and associate a node (successor of the root) to each one. The branch leading to a node is labeled with the condition under which the node's extended polyhedron was obtained. Likewise, for each new node, generate its successors by applying the *restriction* procedure, and label the new branches correspondingly. For the lastly obtained successors, continue with the *projection* procedure. Iterate the sequence of procedures in this order until the final vertex is reached, and terminate by a restriction to intersect the final region.

At this point, any sequence of branches from the root to a leaf defines a sufficient condition on the parameters, for the final region to be reachable from the initial one (it is the conjunction of the conditions on all branches). However, two problems arise. First, such a condition might be unsatisfiable. Second, there might be an infinity of vertex paths in the automaton, from a formula's initial vertex to a final one; this means that in order to generate all the conditions (necessary and sufficient) on the parameters for the reachability to hold, we might need to iterate the above operations on an infinite number of vertex paths. This is why we consider a restricted class of automata and formulas, for which we have to analyze only a finite number of vertex paths.

Uniformly Low-bounded Hybrid Automata, Time-bounded Reachability. A relaxed hybrid automaton is a rectangular hybrid automaton [HKPV95] obtained from a slope-parametric hybrid automaton by replacing each parameter by the corresponding interval. For instance, a relaxed hybrid automaton is obtained from the slope-parametric hybrid automaton of figure 1 by replacing '$= k_1$' with '$\in [-10, 10]$' and '$= k_2$' with '$\in [0, 5]$'. If the relaxed hybrid automaton is *uniformly low-bounded* (i.e. there is a uniform, strictly positive low-bound on the durations of cyclic runs) then any time-bounded reachability formula is decidable and furthermore *there are only a finite number of vertex paths by which reachability is possible* [RR96]. In consequence, we have to iterate the three above-mentioned operations only on a finite number of vertex paths, because if there exists a solution to parametric analysis, it will be found on some of these paths. Thus, we can find the necessary and sufficient conditions on the parameters such that the time-bounded reachability formula is true.

As an application, consider a real-time system composed of a number of tasks running at adjustable speeds. We consider the problem of finding the necessary and sufficient conditions on the speeds for all tasks to terminate within given deadlines. Such a real-time system can be modeled by a uniformly low-bounded hybrid automaton [RR96], and the property that a task terminates within a (constant) delay c can be expressed by the negation of a time-bounded reachability formula (expressing that a task cannot be continuously running for a time c). Thus, we have a theoretical means to solve the given parametric problem.

4 Conclusion

We have presented a new approach to the parametric analysis of hybrid automata, which focuses on computing slopes of variables (rather than delays) for some safety requirement to be respected. The method is an extension of the polyhedra-based symbolic analysis [ACH+95] of hybrid automata. The main operation is the Fourier-Motzkin's algorithm [Zie95] to deal with parametric polyhedra, which imposes to consider a large number of cases. We are currently studying several ways of improving its efficiency. This work is a first step towards the study of the more general case of hybrid automata with evolution laws of the form $\dot{x} + \dot{y} = k$.

Acknowledgments. We thank the anonymous referees for their useful comments on the previous version of this paper.

References

ACH+95. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

AHH93. R. Alur, T. A. Henzinger, and P-H. Ho. Automatic symbolic verification of embedded systems. In *Proc. 14th Annual Real-time Systems Symposium, RTSS'93*, pages 2–11. IEEE Computer Society Press, 1993.

AHV93. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing, STOC'93*, pages 592–601, 1993.

HH94. T. A. Henzinger and P.-H. Ho. HyTech: the cornell HYbrid TECHnology tool. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 265–294. Springer-Verlag, 1994.

HKPV95. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. of the 27th Annual ACM Symposium on Theory of Computing, STOC'95*, pages 373–382, 1995.

RR96. O. Roux and V. Rusu. Uniformity for the decidability of hybrid automata. In *Internat. Static Analysis Symposium, LNCS 1145*, pages 301–316, 1996.

Zie95. G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.