Closed-Form Evaluation and Abstraction for Parametric Timed Automata (Abstract)

Béatrice Bérard and Laurent Fribourg *

Ecole Normale Supérieure Cachan/CNRS - 61 av. Pdt. Wilson - 94235 Cachan - France
email: {berard,fribourg}@lsv.ens-cachan.fr

Abstract. We present here a method for performing backward reachability analysis of Alur-Dill's timed automata, which is inspired from Revesz's procedure for computing least fixed-points of logic programs with arithmetical constraints. The method extends to timed automata which use parameters instead of constants within clock constraints. In order to guarantee the termination of the procedure in this case, we perform an abstraction of the reachability relation, and the procedure does not yield any longer the exact set of predecessor states, but just an overapproximation. The method also extends to control points with data variables ("memory cells"). The procedure has been successfully experimented for proving automatically the correctness of a sophisticated reactive program that controls dataflow rates on ATM networks.

1 Introduction

Reachability analysis is a classical method used for proving "safety properties" of concurrent systems modeled by automata [14]. Given an automaton $\mathcal{A}$, the *successor relation* $\rightarrow_{\mathcal{A}}$ is defined as follows: $q \rightarrow_{\mathcal{A}} q'$ iff there exists some transition of $\mathcal{A}$ going from state q to state q'. Given a set of states Q, the successor operator $Post(Q)$ is the set of states q' such that $q \rightarrow_{\mathcal{A}} q'$ for some $q \in Q$. Likewise the predecessor operator $Pre(Q)$ is the set of states q' such that $q' \rightarrow_{\mathcal{A}} q$ for some $q \in Q$. The set of iterated successors $Post^*(Q)$ (resp. set of iterated predecessors $Pre^*(Q)$) characterizes the set of states q' which are forward reachable (resp. backward reachable) from some state $q \in Q$ via some arbitrarily long sequence of transitions of $\mathcal{A}$. In order to prove that some safety property Π holds for $\mathcal{A}$ it suffices to show that $Post^*$ applied to the initial state is included into the set of states Q_{Π} corresponding to Π. [1] This is called the *forward reachability construction.* Alternatively it is possible to prove that $\mathcal{A}$ satisfies Π by showing that the initial state does not belong to $Pre^*(Q_{\neg\Pi})$ where $Q_{\neg\Pi}$ denotes the set of

* Supported by Action *FORMA* (Programme DSP-STTC/CNRS/MENRT)

[1] In order to characterize Π under the form of a set of states, it is often necessary to compose $\mathcal{A}$ with an "observer" automaton (which observes the behaviour of $\mathcal{A}$ and detects any violation of Π) and to construct $Post^*$ for the resulting product [9].

states corresponding to $\neg\Pi$. This is called the *backward reachability construction.*

In the case of finite state automata, the space of states is finite, and the construction of $Post^*$ and Pre^* is guaranteed to terminate. We are interested here in *timed automata* [4] which are finite state systems equipped with a set of *clocks*, used to measure time. Clocks evolve continuously and at the same rate as time progresses. A transition is enabled only if a set of *clock constraints*, is satisfied. In the case of a timed automaton $\mathcal{A}$ with n clocks, states are vectors of n nonnegative reals corresponding to clock values, and are therefore in infinite number. The *symbolic approach* consists in representing states under the form of a vector of n variables constrained by some relations. A typical symbolic form is a "zone", that is a finite conjunction of inequalities of the form $x \ll c, c \ll x, x - y \ll c$ or $c \ll x - y$ where x, y are clock variables, c is a nonnegative integer, and $\ll$ is either $<$ or $\leq$. The operators $Post$ and Pre extend to symbolic states (zones) in the natural way. The interesting property of zones is their *closure* with respect to $Post$ and Pre operations: the successor (resp. predecessor) of a zone is a finite set of zones. One can therefore iterate $Post$ and Pre and try to construct $Post^*$ and Pre^*. Unfortunately the process is not guaranteed to terminate for $Post$ because the constants c involved in the linear inequalities of zones may increase indefinitely as far as $Post$ is iterated, so that the number of forward reachable symbolic states may be infinite. An abstraction $\leadsto_{\mathcal{A}}$ of the successor relation $\rightarrow_{\mathcal{A}}$ has then been introduced in [4], for which the induced abstract space of symbolic states, called "region graph", is finite. As a consequence the iteration of $Post$ and Pre for this abstraction always terminates. Besides, this abstraction does not entail any loss of information: roughly speaking, two symbolic states are related via $\rightarrow_{\mathcal{A}}$ iff their abstract forms are related via $\leadsto_{\mathcal{A}}$ (see e.g. [6] for a study of timed abstractions). An alternative way to perform reachability analysis is to consider the original relation $\rightarrow_{\mathcal{A}}$ (instead of the abstraction $\leadsto_{\mathcal{A}}$), but to iterate Pre instead of $Post$. It can be shown that, when computing iteratively Pre starting from a zone ϕ', the constants c involved in the generated zones are bounded by the largest constant appearing in ϕ' or in the clock constraints enabling transitions of $\mathcal{A}$. Therefore the number of possible zones backward reachable is finite, and the procedure Pre^* is guaranteed to terminate. We sketch here a formal proof of this "folk" result, using a change of variables. We also propose an abstract form of relation $\rightarrow_{\mathcal{A}}$ for performing reachability analysis with *parametric* timed automata, i.e. timed automata where clock constraints of transitions involve parameters instead of integer constants. These results extend to a class of automata with *data variables* (or "memory cells"), which are variables that do not evolve in time, but are nondeterministically updated through transitions. Finally we explain how such backward reachability constructions have been implemented and discuss experiments done for proving the correctness of a reactive program controlling dataflow rates for ATM networks.

Plan of the paper is as follows: section 2 recalls definitions and results for timed automata; section 3 presents an extended procedure dealing with parameters and data variables; section 4 discusses experimental results.

2 Timed Automata

2.1 Definitions

We consider here (a variant of) Alur-Dill's timed automata [4,1]. As finite state automata, timed automata have a finite set S of locations (denoted by $s, s', ...$). In addition, they have a finite set X of variables $x_1, ..., x_n$, called "clocks", which all evolve continuously in time and at the same rate. A transition is enabled only if some relation ψ, called "transition guard", is satisfied by the current values of clocks $x_1, ..., x_n$. When a transition is fired, some clocks are reset to fixed values, while the other ones remain unchanged. Additionally, for each location s, variables $x_1, ..., x_n$ are required to satisfy a relation I_s, called "location invariant". We also assume that a **universal clock**, denoted by τ, is given. The value of such a clock increases at the same rate as clocks x_i, but is **never reset**. We finally assume given a sequence of $m+1$ natural numbers $c_0, c_1, ..., c_m$ such that $0 = c_0 < c_1 < ... < c_m$ and we write $\overline{c}$ for the vector $(c_0, c_1, ..., c_m)$. Formally, for a set X of clock variables, the set $\Gamma(X)$ of *clock constraints* γ is defined by the grammar

$\gamma := \quad x \leq d \mid d \leq x \mid x < d \mid d < x \mid \gamma_1 \wedge \gamma_2$

with $x \in X$ and $d \in \{c_0, c_1, ..., c_m\}$.

A *timed automaton* T is a tuple $\langle S, s_{init}, X, \tau, I, E\rangle$, where

- S is a finite set of locations,
- s_{init} is an element of S called *initial location*,
- X is a finite set of clocks,
- τ is a real variable, called *universal clock*,
- I is a mapping that labels each location s in S with some clock constraint in $\Gamma(X)$, and
- E is a set of tuples $\langle s, \psi, \rho, s'\rangle$, where $s, s' \in S$, $\psi \in \Gamma(X)$, and ρ is a mapping over X such that, for all $x \in X$, either $\rho(x) = x$ or $\rho(x) = -d_x$ with $d_x \in \{c_0, ..., c_m\}$.

The clock constraint $I(s)$ associated with $s \in S$ via I, is called *location invariant of s*, and will be simply denoted I_s in the following.
A tuple $\langle s, \psi, \rho, s'\rangle$ in E is called an *action transition* where s is the *source location*, s' the *target location*, ψ the *guard*, and ρ the *reset operation*.

Let n be the number of elements of X. Without loss of understanding, we will interpret x as a real variable and $\overline{x}$ as a vector $(x_1, ..., x_n)$ of n real variables. We will denote by σ a variable taking its values on S. We now define formulas of the form $\phi(\overline{q})$ or $\phi(\overline{q}, \overline{q}')$, where $\overline{q}$ (resp. $\overline{q}'$) stands for the vector of variables $(\sigma, \tau, \overline{x})$ (resp. $(\sigma', \tau', \overline{x}')$):

- The *initial formula* $\phi_{init}(\sigma, \tau, \overline{x})$ is:
  $\sigma = s_{init} \wedge \tau = 0 \wedge \overline{x} = -\overline{c}_{init}$, with $\overline{c}_{init} \in (\{c_0, c_1, ..., c_m\})^n$.
- The *action successor relation via* $e : \langle s, \psi, \rho, s' \rangle$, denoted $\rightarrow_e$, is such that:
  $(\sigma, \tau, \overline{x}) \rightarrow_e (\sigma', \tau, \overline{x}')$ iff
  $\sigma = s \ \wedge \ \sigma' = s' \ \wedge \ I_\sigma(\overline{x}) \ \wedge I_{\sigma'}(\overline{x}') \ \wedge \ \psi(\overline{x}) \ \wedge \ \overline{x}' = \rho(\overline{x})$.
- The *delay successor relation*, denoted $\rightarrow_\delta$, is such that:
  $(\sigma, \tau, \overline{x}) \rightarrow_\delta (\sigma, \tau', \overline{x}')$ iff
  $\exists \varepsilon \in \mathbb{R}^+ \ (I_\sigma(\overline{x}) \ \wedge \ I_\sigma(\overline{x}') \ \wedge \ \tau' = \tau + \varepsilon \ \wedge \ \overline{x}' = \overline{x} + \varepsilon)$.
- The *action successor relation via* E, denoted $\rightarrow_E$, is such that:
  $(\sigma, \tau, \overline{x}) \rightarrow_E (\sigma', \tau, \overline{x}')$ iff $(\sigma, \tau, \overline{x}) \rightarrow_e (\sigma', \tau, \overline{x}')$ for some $e \in E$.
- The *successor relation via automaton* T, denoted $\rightarrow_T$, is $\rightarrow_E \ . \rightarrow_\delta$.
- The *iterated successor relation*, denoted $\rightarrow_T^*$, is $\rightarrow_\delta \ . \ (\rightarrow_E \ . \ \rightarrow_\delta)^*$.

Remark

Note that, for all $1 \leq i \leq n$, x_i is initially set to a nonpositive value $-c_i^{init}$ (with $c_i^{init} \in \{c_0, c_1, ..., c_m\}$) while τ is initially set to 0. Furthermore x_i and τ evolve with the same rate as time, and x_i is sometimes reset to some nonpositive value $-d_i$ (with $d_i \in \{c_0, c_1, ..., c_m\}$) while τ is never reset. It follows that, for all $1 \leq i \leq n$, we have $-c_m \leq x_i \leq \tau$.

Example 1

We consider the timed automaton in Figure 1, inspired from the train-controller example in [12], p.404. We thus add action names on the transition labels, to make interpretation easier. Recall that, in this example, actions *app*, *in* (or *in'*) and *out* from the train indicate respectively approach, entering and exiting a railroad crossing. Actions *down* and *up* from the controller indicate closing and opening of the gate. As usual, the locations are represented in circles, including their invariants, with 00 as initial location. The transitions are represented by arrows, where each label contains an action transition name with, above, the guard (which is *true* when missing) and, below, the parts of the reset operations for which the clock is changed. We focus here on (big circled) locations 00, 11, 12 and 22, and transitions *app*, *down* and *in*. For instance, invariants associated with states 00 and 11 are:

$$I_{00} : \ true$$
$$I_{11} : \ x_1 \leq 0 \wedge x_2 < 0$$

while the guards and reset operations for actions *app* and *down* are:

$$\begin{array}{ll} \psi_{app} : \ true & \rho_{app}(x_1) = -5 \\ & \rho_{app}(x_2) = -2 \\ & \rho_{app}(x_3) = -5 \\ \psi_{down} : \ -1 \leq x_2 < 0 & \rho_{down}(x_1) = x_1 \\ & \rho_{down}(x_2) = x_2 \\ & \rho_{down}(x_3) = -5 \end{array}$$

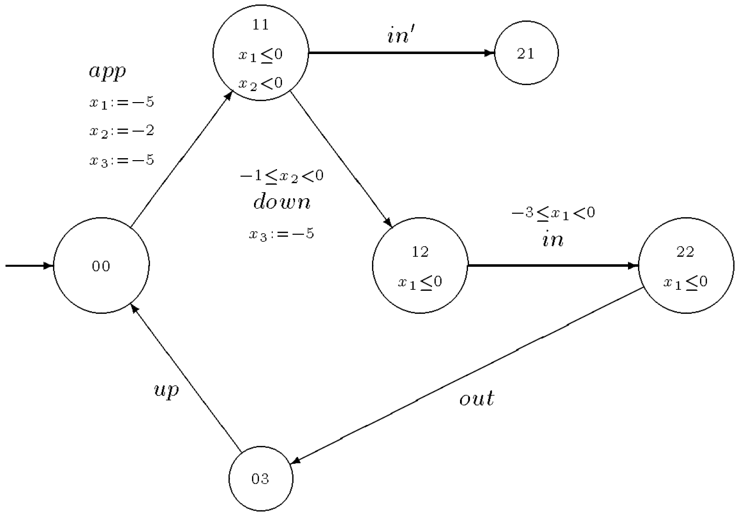

Fig. 1. Train-controller automaton

2.2 Change of variables

In order to relate reachability analysis in timed automata with Revesz's method [17], we introduce a change of variables. The new variables are y_i, defined to be equal to $\tau - x_i$, for all $1 \leq i \leq n$. Since $-c_m \leq x_i \leq \tau$, we can assume: $0 \leq y_i \leq \tau + c_m$. We also introduce τ_j as a new variable defined to be equal to $\tau + c_j$, for all $0 \leq j \leq m$. Note that we have $\tau = \tau_0 < \tau_1 < ... < \tau_m$. The expression $\overline{\tau}$ will denote the vector $(\tau_0, \tau_1, ..., \tau_m)$. Formally, for a set Y of clock variables, the set $\Gamma(\overline{\tau}, Y)$ of τ-*constraints* γ is defined by the grammar

$\gamma := \quad \tau_k \leq y \mid y \leq \tau_k \mid \tau_k < y \mid y < \tau_k \mid \gamma_1 \wedge \gamma_2$

with $y \in Y$ and $0 \leq k \leq m$.

A *timed* τ-*automaton* $\mathcal{U}$ is a tuple $\langle S, s_{init}, Y, \overline{\tau}, J, F \rangle$ where

- S is a finite set of locations,
- s_{init} is an element of S called *initial location*,
- Y is a finite set of discrete variables,
- $\overline{\tau}$ is a vector of variables $(\tau_0, \tau_1, ..., \tau_m)$,
- J is a mapping that labels each location s in S with some τ-constraint in $\Gamma(\overline{\tau}, Y)$, and
- F is a set of tuples $\langle s, \psi, \rho, s' \rangle$, where $s, s' \in S$, $\psi \in \Gamma(\overline{\tau}, Y)$, and ρ is a mapping over $\{\overline{\tau}\} \times Y$ such that, for all $y \in Y$, either $\rho(\overline{\tau}, y) = y$ or $\rho(\overline{\tau}, y) = \tau_k$ with $k \in \{0, 1, ..., m\}$.

The τ-constraint $J(s)$ associated with $s \in S$ via J, is called *location* τ-*invariant of* s, and will be simply written J_s in the rest of the paper.

A tuple $\langle s, \psi, \rho, s' \rangle$ in F is called a *τ-action transition* where s is the *source location*, s' the *target location*, ψ the *τ-guard*, and ρ the *τ-reset operation.*

We consider the following formulas:

- The *initial formula* $\phi_{init}(\sigma, \overline{\tau}, \overline{y})$ is:

$$\sigma = s_{init} \wedge \overline{\tau} = \overline{c} \wedge \overline{y} = \overline{c}_{init}, \qquad \text{with } \overline{c}_{init} \in (\{c_0, c_1, ..., c_m\})^n.$$

- The *τ-action successor relation via f:* $\langle s, \psi, \rho, s' \rangle$, written $\rightarrow_f$, is such that:

$$(\sigma, \overline{\tau}, \overline{y}) \rightarrow_f (\sigma', \overline{\tau}, \overline{y}') \quad \text{iff}$$

$$\sigma = s \ \wedge\ \sigma' = s' \ \wedge\ \ J_\sigma(\overline{\tau}, \overline{y}) \ \ \wedge J_{\sigma'}(\overline{\tau}, \overline{y}') \ \wedge\ \psi(\overline{\tau}, \overline{y}) \ \wedge\ \overline{y}' = \rho(\overline{\tau}, \overline{y}) \ .$$

- The *τ-delay successor relation*, $\rightarrow_\delta^\tau$, is defined by:

$$(\sigma, \overline{\tau}, \overline{y}) \rightarrow_\delta^\tau (\sigma, \overline{\tau}', \overline{y}) \quad \text{iff}$$

$$\exists \varepsilon \in \mathbb{R}^+ \ \ (0 \leq \overline{y} \leq \tau_m \ \ \wedge\ \ J_\sigma(\overline{\tau}, \overline{y}) \ \ \wedge\ \ J_\sigma(\overline{\tau}', \overline{y}) \ \ \wedge\ \ \tau_0' = \tau_0 + \varepsilon \ \ \wedge\ \ \overline{\tau} = \tau_0 + \overline{c} \ \wedge\ \overline{\tau}' = \tau_0' + \overline{c}).$$

or more simply: $(\sigma, \overline{\tau}, \overline{y}) \rightarrow_\delta^\tau (\sigma, \overline{\tau}', \overline{y})$ iff

$$0 \leq \tau_0 \leq \tau_0' \ \wedge\ 0 \leq \overline{y} \leq \tau_m \wedge J_\sigma(\overline{\tau}, \overline{y}) \ \wedge\ J_\sigma(\overline{\tau}', \overline{y}) \ \wedge \overline{\tau} = \tau_0 + \overline{c} \wedge \overline{\tau}' = \tau_0' + \overline{c}.$$

- The *τ-action successor relation via F*, $\rightarrow_F$, is defined by:

$$(\sigma, \overline{\tau}, \overline{y}) \rightarrow_F (\sigma', \overline{\tau}, \overline{y}') \quad \text{iff} \quad (\sigma, \overline{\tau}, \overline{y}) \rightarrow_f (\sigma', \overline{\tau}, \overline{y}') \text{ for some } f \in F.$$

- The *successor relation via automaton $\mathcal{U}$*, $\rightarrow_\mathcal{U}$, is $\rightarrow_F \ . \ \rightarrow_\delta^\tau$.
- The *iterated successor relation*, $\rightarrow_\mathcal{U}^*$, is $\rightarrow_\delta^\tau \ . \ (\rightarrow_F \ . \ \rightarrow_\delta^\tau)^*$.

It is easy to transform a given timed automaton $T : \langle S, s_{init}, X, \tau, I, E \rangle$ into a τ-automaton $\mathcal{U} : \langle S, s_{init}, Y, \overline{\tau}, J, F \rangle$ by introducing a vector $\overline{\tau}$ of $m+1$ variables $\tau_0, \tau_1, ..., \tau_m$ (with $\tau = \tau_0$) and changing all occurrence of x_i with $y_i = \tau_0 - x_i$. By construction, we have:

Property 1

$(\sigma, \tau_0, \overline{x}) \rightarrow_T^* (\sigma', \tau_0', \overline{x}')$ implies $(\sigma, \overline{\tau}, \tau_0 - \overline{x}) \rightarrow_\mathcal{U}^* (\sigma', \overline{\tau}', \tau_0' - \overline{x}')$.
$(\sigma, \overline{\tau}, \overline{y}) \rightarrow_\mathcal{U}^* (\sigma', \overline{\tau}', \overline{y}')$ implies $(\sigma, \tau_0, \tau_0 - \overline{y}) \rightarrow_T^* (\sigma', \tau_0', \tau_0' - \overline{y}')$.

Example 2

We apply the change of variables to the automaton of example 1. The constants appearing in invariants, guards and reset actions of the automaton are $1, 2, 3$ and 5. Therefore $m = 4$ and c_0, c_1, c_2, c_3, c_4 are respectively $0, 1, 2, 3, 5$. We have implicitly: $\tau_1 = \tau_0 + c_1 = \tau_0 + 1$, $\tau_2 = \tau_0 + c_2 = \tau_0 + 2$, $\tau_3 = \tau_0 + c_3 = \tau_0 + 3$, $\tau_4 = \tau_0 + c_4 = \tau_0 + 5$. The resulting $\overline{\tau}$-invariants, $\overline{\tau}$-guards and reset $\overline{\tau}$-operations are illustrated in Figure 2. For instance, the $\overline{\tau}$-invariants of states 00 and 11 are:

$$\begin{array}{ll} J_{00}: & true \\ J_{11}: & \tau_0 \leq y_1 \wedge \tau_0 < y_2 \end{array}$$

and the $\overline{\tau}$-guards and reset $\overline{\tau}$-operations corresponding to actions *app* and *down* are:

$$\begin{array}{ll} \psi_{app} : \ true & \rho_{app}(\overline{\tau}, y_1) = \tau_4 \\ & \rho_{app}(\overline{\tau}, y_2) = \tau_4 \\ & \rho_{app}(\overline{\tau}, y_3) = \tau_4 \\ \psi_{down} : \ \tau_0 < y_2 \wedge y_2 \leq \tau_1 & \rho_{down}(\overline{\tau}, y_1) = y_1 \\ & \rho_{down}(\overline{\tau}, y_2) = y_2 \\ & \rho_{down}(\overline{\tau}, y_3) = \tau_4 \end{array}$$

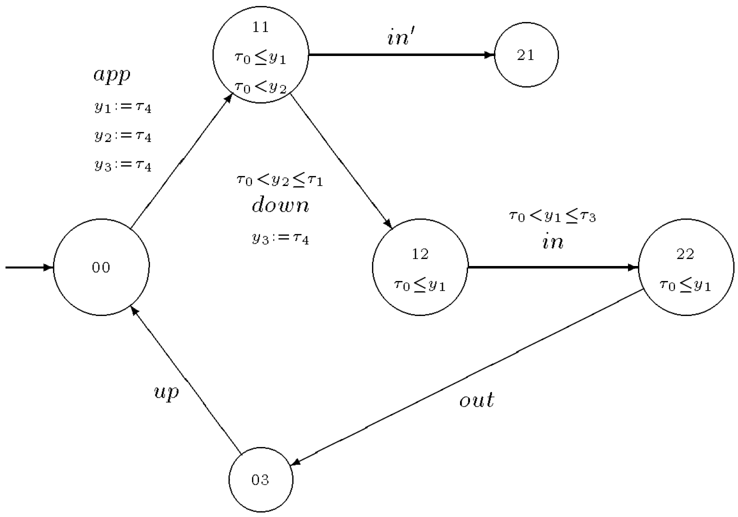

Fig. 2. Variable change in the train-controller automaton

2.3 Closed form

We propose here the definition of a class of formulas under "closed form", inspired by the notion of "gap-configurations" introduced by Revesz [17] (*cf.* section 4.1). This class is interesting because of its closure under backward application of transitions. Moreover, it contains only a finite number of formulas, so that the application of Pre^* will be guaranteed to terminate.

Definition 2.1 (τ-Zone)
A formula $\phi(\overline{y}, \overline{\tau})$ is a *bounded τ-zone* iff it is a conjunction of inequalities of the form:

$y_i + d \ll \tau_k, \quad \tau_k \ll y_i + d, \quad y_i + d \ll y_j, \quad y_i \ll y_j + d,$
with $d \in \{0, 1, ..., c_m\}$, $i, j \in \{1, ..., n\}$, $k \in \{0, 1, ..., m\}$, and $\ll \in \{<, \leq\}$.

Definition 2.2
A formula $\phi(\sigma, \overline{\tau}, \overline{y})$ *belongs to class* $\mathcal{C}$ iff it is either $false$ or a disjunction of the form $\bigvee_{k \in K} \ 0 \leq \overline{y} \leq \tau_m \wedge \overline{\tau} = \tau_0 + \overline{c} \wedge \sigma = s_k \wedge \phi_k(\overline{y}, \overline{\tau})$, where K denotes a subset of indices, s_k is in S, and $\phi_k(\overline{y}, \overline{\tau})$ is a bounded τ-zone.

Remark
The class $\mathcal{C}$ is finite. (The number of formulas $\phi(\sigma, \overline{\tau}, \overline{y})$ is hyperexponential in the number n of clocks $y_1, ..., y_n$ and exponential in the maximal constant c_m.)

We now have:

Lemma 2.1 (backward closure via $\rightarrow_\delta^\tau$)
Let ϕ' be a formula in $\mathcal{C}$. Then there exists a formula ϕ in $\mathcal{C}$, denoted by $pre_\delta(\phi')$, such that:

$$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \ ((\sigma, \overline{\tau}, \overline{y}) \rightarrow_\delta^\tau (\sigma', \overline{\tau}', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$$

Lemma 2.2 (backward closure via $\rightarrow_f$)
Let ϕ' be a formula in $\mathcal{C}$, and f a τ-action transition in F. Then there exists a formula ϕ of $\mathcal{C}$, denoted by $pre_f(\phi')$, such that:

$$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \ ((\sigma, \overline{\tau}, \overline{y}) \rightarrow_f (\sigma', \overline{\tau}', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$$

Proofs of these lemmas will be given in the full version of the paper.
Combining them yields immediately:

Lemma 2.3 (backward closure via $\rightarrow_\mathcal{U}$)
Let ϕ' be a formula of $\mathcal{C}$. Then there exists a formula ϕ of $\mathcal{C}$, written $Pre(\phi')$, such that:

$$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \ ((\sigma, \overline{\tau}, \overline{y}) \rightarrow_\mathcal{U} (\sigma', \overline{\tau}', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$$

Define now

$Pre^0(\phi') = \phi'$, and

$Pre^{l+1}(\phi') = Pre(Pre^l(\phi')) \cup Pre^l(\phi')$.

Using the facts that Pre preserves $\mathcal{C}$ (by lemma 2.3), and $\mathcal{C}$ is a finite set of formulas, we obtain that, for each formula $\phi' \in \mathcal{C}^{max}$, there exists $l \in \mathbb{N}$ such that $Pre^{l+1}(\phi') = Pre^l(\phi')$. The formula $pre_\delta(Pre^l(\phi'))$, abbreviated as $Pre^*(\phi')$, characterizes the formula $(\sigma, \tau, \overline{y})$ such that $\exists \sigma', \tau', \overline{y}' \quad ((\sigma, \tau, \overline{y}) \rightarrow_\mathcal{U}^* (\sigma', \tau', \overline{y}') \quad \wedge \quad \phi'(\sigma', \tau', \overline{y}'))$. We recapitulate as follows:

Theorem 2. (Termination of Pre^*)
Let ϕ' be a formula in $\mathcal{C}$. Then there exists a formula ϕ of $\mathcal{C}$, obtained by iteration of Pre (followed by one application of pre_δ), such that:

$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \;\; ((\sigma, \overline{\tau}, \overline{y}) \rightarrow^*_{\mathcal{U}} (\sigma', \overline{\tau}', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$
Formula ϕ is called the *iterated predecessor of* ϕ' *via* $\mathcal{U}$, and written $Pre^*(\phi')$.

Remark
Let Π be a property such that some formula $\phi_{\neg \Pi}$ in $\mathcal{C}$ corresponds to the negation of Π. To prove Π, it suffices to prove that $Pre^*(\phi_{\neg \Pi}) \Rightarrow \neg \phi_{init}$ holds (recall that ϕ_{init} is the initial formula).

Example 3
Consider the τ-automaton of example 2. We want to show that in location 22, property $\Pi: \; \tau_0 < y_3$ always holds. To do this, we express the negation of Π, viz, H: $y_3 \leq \tau_0$, and we compute the set of iterated predecessors of $\sigma = 22 \wedge H$, by applying backward delay transitions alternatively with action transitions *in*, *down* and *app*. For the sake of conciseness, we will not explicitly write the ubiquous constraints $\overline{\tau} = \tau_0 + \overline{c}$ (i.e., $\tau_1 = \tau_0 + 1, \tau_2 = \tau_0 + 2, \tau_3 = \tau_0 + 3, \tau_4 = \tau_0 + 5$), $\overline{\tau}' = \tau'_0 + \overline{c}$ and $y_i \leq \tau_4, y'_i \leq \tau'_4$ for $i = 1, 2, 3$.
We have:

$pre_\delta(\sigma = 22 \wedge H)$

$$\Leftrightarrow \quad \sigma = 22 \wedge \exists \overline{\tau}' \;\; ((22, \overline{\tau}, \overline{y}) \rightarrow^{\overline{\tau}}_{\delta} (22, \overline{\tau}', \overline{y}) \quad \wedge \quad y_3 \leq \tau'_0).$$
$$\Leftrightarrow \quad \sigma = 22 \wedge \exists \overline{\tau}' \; (\tau_0 \leq \tau'_0 \; \wedge \; \tau'_0 \leq y_1 \quad \wedge \quad y_3 \leq \tau'_0).$$
$$\Leftrightarrow \quad \sigma = 22 \wedge \; \tau_0 \leq y_1 \; \wedge \; y_3 \leq y_1.$$

Let H' be $\tau_0 \leq y_1 \wedge y_3 \leq y_1$. We have:

$pre_{in}(\sigma = 22 \wedge H')$

$$\Leftrightarrow \quad \sigma = 12 \wedge \exists \overline{y}' \;\; ((12, \overline{\tau}, \overline{y}) \rightarrow_{in} (22, \overline{\tau}, \overline{y}') \;\; \wedge \;\; \tau_0 \leq y'_1 \wedge y'_3 \leq y'_1)$$
$$\Leftrightarrow \quad \sigma = 12 \wedge \; \tau_0 < y_1 \; \wedge \; y_1 \leq \tau_3 \quad \wedge \quad \tau_0 \leq y_1 \wedge y_3 \leq y_1$$
$$\Leftrightarrow \quad \sigma = 12 \wedge \; y_1 \leq \tau_3 \; \wedge \; \tau_0 < y_1 \quad \wedge \quad y_3 \leq y_1.$$

Let G be $y_1 \leq \tau_3 \; \wedge \; \tau_0 < y_1 \; \wedge \; y_3 \leq y_1$. We have:

$pre_\delta(\sigma = 12 \wedge G)$

$$\Leftrightarrow \quad \sigma = 12 \wedge \exists \overline{\tau}' \; ((12, \overline{\tau}, \overline{y}) \rightarrow^{\overline{\tau}}_{\delta} (12, \overline{\tau}', \overline{y}) \; \wedge \; y_1 \leq \tau'_3 \wedge \tau'_0 < y_1 \wedge y_3 \leq y_1).$$
$$\Leftrightarrow \quad \sigma = 12 \wedge \exists \overline{\tau}' \; (\tau_0 \leq \tau'_0 \; \wedge \; \tau_0 \leq y_1 \quad \wedge \quad y_1 \leq \tau'_3 \; \wedge \; \tau'_0 < y_1 \wedge y_3 \leq y_1)$$
$$\Leftrightarrow \quad \sigma = 12 \; \wedge \; \tau_0 < y_1 \wedge y_3 \leq y_1.$$

Let G' be $\tau_0 < y_1 \wedge y_3 \leq y_1$. We have:

$pre_{down}(\sigma = 12 \wedge G')$

$$\Leftrightarrow \quad \sigma = 11 \wedge \exists \overline{y}' \;\; ((11, \overline{\tau}, \overline{y}) \rightarrow_{down} (12, \overline{\tau}, \overline{y}') \quad \wedge \quad \tau_0 < y'_1 \wedge y'_3 \leq y'_1)$$
$$\Leftrightarrow \quad \sigma = 11 \wedge \; y_2 \leq \tau_1 \wedge \; \tau_0 < y_2 \; \wedge \; \tau_0 \leq y_1 \quad \wedge \quad \tau_0 < y_1 \wedge \tau_4 \leq y_1$$
$$\Leftrightarrow \quad \sigma = 11 \wedge \; y_2 \leq \tau_1 \; \wedge \; \tau_0 < y_2 \quad \wedge \quad \tau_4 \leq y_1.$$

Let F be $y_2 \leq \tau_1 \; \wedge \; \tau_0 < y_2 \wedge \tau_2 \leq y_1$. We have:

$pre_\delta(\sigma = 11 \wedge F)$

$$\Leftrightarrow \quad \sigma = 11 \wedge \exists \overline{\tau}' \; ((11, \overline{\tau}, \overline{y}) \rightarrow^{\overline{\tau}}_{\delta} (11, \overline{\tau}', \overline{y}) \quad \wedge \quad y_2 \leq \tau'_1 \; \wedge \; \tau'_0 < y_2 \; \wedge \; \tau'_4 \leq y_1)$$
$$\Leftrightarrow \quad \sigma = 11 \wedge \exists \overline{\tau}' \; (\tau_0 \leq \tau'_0 \wedge \tau'_0 \leq y_1 \; \wedge \; \tau'_0 < y_2 \;\; \wedge \;\; y_2 \leq \tau'_1 \wedge \tau'_0 < y_2 \wedge \tau'_4 \leq y_1)$$
$$\Leftrightarrow \quad \sigma = 11 \wedge \; \tau_0 \leq y_1 \; \wedge \; \tau_0 < y_2 \quad \wedge \quad y_2 + 4 \leq y_1.$$

Let F' be $\tau_0 \leq y_1 \wedge \tau_0 < y_2 \wedge y_2 + 4 \leq y_1$. We have:

$pre_{app}(\sigma = 11 \wedge F')$

$$\Leftrightarrow \quad \sigma = 00 \wedge \exists \overline{y}' \;\; ((00, \overline{\tau}, \overline{y}) \rightarrow_{app} (11, \overline{\tau}, \overline{y}') \wedge \tau_0 \leq y'_1 \wedge \tau_0 < y'_2 \wedge y'_2 + 4 \leq y'_1)$$
$$\Leftrightarrow \quad \sigma = 00 \wedge \exists \overline{y}' \;\; ((00, \overline{\tau}, \overline{y}) \rightarrow_{app} (11, \overline{\tau}, \overline{y}') \;\; \wedge \;\; \tau_0 \leq \tau_4 \wedge \tau_0 < \tau_2 \;\; \wedge \;\; \tau_2 + 4 \leq$$

τ_4)
$\Leftrightarrow$ $false$. (since $\tau_2 = \tau_0 + 2$ and $\tau_4 = \tau_0 + 5$)
There are no more transition applicable backwards. The iterated predecessor is thus:

$$Pre^*(\sigma = 22 \wedge H) = (\sigma = 22 \wedge (H \vee H')) \vee (\sigma = 12 \wedge (G \vee G')) \vee (\sigma = 11 \wedge (F \vee F')).$$

Since no disjunct contains the initial location $\sigma = 00$, we can conclude that property Π is true.

3 Parametric Timed Automata

3.1 Parameters

From now on, we assume that $c_1, ..., c_m$ are no longer natural numbers, but are instead **parameters** such that $0 < c_1 < ... < c_m$. The extended timed automata we consider thus become "parametric" automata and we are interested in the *parametric verification problem* [5]: given a parametric timed automaton T and a property Π, is Π true for all values of the parameters $c_1, ..., c_m$? In other words, we wish to verify that no values of $c_1, ..., c_m$ make the initial state backward reachable from the final state (corresponding to $\neg\Pi$). It has been shown in [5] that such an "emptiness" problem is undecidable for timed automata with three parametric clocks. Therefore, for parametric timed automata, there is no hope to find a decidable class $\mathcal{C}$ of formulas characterizing the backward reachability fixed-point, as done in Section 2. However, our backward reachability construction for extended $\overline{\tau}$-automata, proposed in Section 2, can be adapted to the case of extended $\overline{\tau}$-automata with parameters by considering an **abstract** form of the successor relation. As a consequence, we obtain an overapproximation, written Pre^*_{abs}, instead of the exact iterated predecessor Pre^* (i.e. $Pre^*(\phi') \Rightarrow Pre^*_{abs}(\phi')$, for each formula ϕ'). The abstract form of successor relations is obtained by replacing constraints of the form
$0 \le \tau_0 \le \tau'_0 \ \wedge \ \overline{\tau} = \tau_0 + \overline{c} \ \wedge \ \overline{\tau}' = \tau'_0 + \overline{c}$
with $0 \le \overline{\tau} \le \overline{\tau}' \ \wedge \ \tau_0 < \tau_1 < ... < \tau_m \ \wedge \ \tau'_0 < \tau'_1 < ... < \tau'_m$.

3.2 Abstract computation of Pre^*

For an automaton $\mathcal{U}$, the definitions of action successor relations via f and F are left unchanged, but the other successor relation definitions are "abstracted" as follows:

- The *delay abstract successor relation*, written $\rightarrow^{abs}_{\delta}$ is such that:
$(\sigma, \overline{\tau}, \overline{y}) \rightarrow^{abs}_{\delta} (\sigma, \overline{\tau}', \overline{y})$ iff
$0 \le \overline{\tau} \le \overline{\tau}' \ \wedge \ 0 \le \overline{y} \le \tau_m \wedge J_\sigma(\overline{\tau}, \overline{y}) \ \wedge \ J_\sigma(\overline{\tau}', \overline{y}) \ \wedge \ \tau_0 < \tau_1 < ... < \tau_m$
$\wedge \ \tau'_0 < \tau'_1 < ... < \tau'_m$.
- The *abstract successor relation via* $\mathcal{U}$, $\rightarrow^{abs}_{\mathcal{U}}$, is

$\rightarrow_F \ . \ \rightarrow_\delta^{abs}$.
- The *iterated abstract successor relation* , $(\rightarrow_{\mathcal{U}}^{abs})^*$, is
$\rightarrow_\delta^{abs} \ . \ (\rightarrow_F \ . \ \rightarrow_\delta^{abs})^*$.

Definitions of section 2.3 become:

Definition 3.1. (Abstract τ-Zone)
A formula $\phi(\overline{y}, \overline{\tau})$ is an *abstract τ-zone* iff it is a conjunction of inequalities of the form:
$y_i \ll \tau_k, \quad \tau_k \ll y_i, \quad y_i \ll y_j,$
with $i, j \in \{1, ..., n\}$, $k \in \{0, 1, ..., m\}$, and $\ll \in \{<, \leq\}$.

Definition 3.2.
A formula $\phi(\sigma, \overline{\tau}, \overline{y})$ *belongs to class* $\mathcal{C}^{abs}$ iff it is either $false$ or a disjunction of the form $\bigvee_{k \in K} \ 0 \leq \overline{y} \leq \tau_m \wedge \tau_0 < \tau_1 < ... < \tau_m \ \wedge \sigma = s_k \wedge \phi_k(\overline{y}, \overline{\tau})$, where K is a subset of indices, s_k is in S, and $\phi_k(\overline{y}, \overline{\tau})$ is an abstract τ-zone.

Remark
The class $\mathcal{C}^{abs}$, like $\mathcal{C}$, is finite (with a number of formulas $\phi(\sigma, \overline{\tau}, \overline{y})$ hyperexponential in $n + m$).

The corresponding properties are:

Lemma 3.1.1 (backward closure via $\rightarrow_\delta^{abs}$)
Let ϕ' be a formula in $\mathcal{C}^{abs}$. Then there exists a formula ϕ in $\mathcal{C}^{abs}$, written $pre_\delta^{abs}(\phi')$, such that:
$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \ ((\sigma, \overline{\tau}, \overline{y}) \rightarrow_\delta^{abs} (\sigma', \overline{\tau}', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$

Lemma 3.1.2. (backward closure via $\rightarrow_{\mathcal{U}}^{abs}$)
Let ϕ' be a formula in $\mathcal{C}^{abs}$. Then there exists a formula ϕ in $\mathcal{C}^{abs}$, denoted by $Pre_{abs}(\phi')$, such that:
$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \ \ ((\sigma, \tau, \overline{y}) \rightarrow_{\mathcal{U}}^{abs} (\sigma', \tau', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$

Theorem 3.1. (Termination of Pre_{abs}^*)
Let ϕ' be a formula in $\mathcal{C}^{abs}$. Then there exists a formula ϕ in $\mathcal{C}^{abs}$, obtained by iteration of Pre_{abs} (followed by pre_δ^{abs}), such that:
$\phi(\sigma, \overline{\tau}, \overline{y}) \quad \Leftrightarrow \quad \exists \sigma', \overline{\tau}', \overline{y}' \ \ ((\sigma, \overline{\tau}, \overline{y}) \rightarrow_{\mathcal{U}}^{abs\ *} (\sigma', \overline{\tau}', \overline{y}') \quad \wedge \quad \phi'(\sigma', \overline{\tau}', \overline{y}')).$
Formula ϕ is called the *iterated abstract predecessor of ϕ' via $\mathcal{U}$*, and is written $Pre_{abs}^*(\phi')$.

Remark
As already mentionned above, Pre_{abs}^* is an overapproximation of Pre^*: $Pre^*(\phi') \Rightarrow Pre_{abs}^*(\phi')$, for all formula ϕ'.
This gives us a **sufficient condition** for the verification of a property Π: if $Pre_{abs}^*(\phi_{\neg\Pi}) \Rightarrow \neg\phi_{init}$, then $Pre^*(\phi_{\neg\Pi}) \Rightarrow \neg\phi_{init}$.

3.3 Locations with data variables

We also consider extended locations of the form $(\sigma, \overline{w})$ instead of standard locations σ. An extended location $(\sigma, \overline{w})$ is a couple where σ denotes a variable over S and $\overline{w}$ is a vector of p real variables satisfying a formula $\chi(\overline{w})$ made of inequalities of the form $w_i \ll w_j$ (with $i, j \in \{1, ..., p\}$, $\ll \in \{<, \leq\}$). The vector $\overline{w}$ can be seen as a vector of **data variables** associated with location s. These data are not changed during delay-transitions, but are (possibly) updated in a non deterministic way through action transitions. Hence, extended locations are useful for modeling unbounded data registers.
In [11], these variables are called "memory cells" and interpreted as special temporal variables with evolution slope equal to 0, in contrast with standard temporal variables with evolution slope 1. Besides, it has been pointed out (p. 377) that Alur and Dill results on classical timed automata can be extended to the case of timed automata with data variables.

Let W be a set of p real variables $w_1, ..., w_p$. The previous definitions extend as follows:

- The *initial formula* $\phi_{init}(\sigma, \overline{w}, \overline{\tau}, \overline{y})$ is $\sigma = s_{init} \wedge \chi_{init}(\overline{w}) \wedge \overline{\tau} = \overline{c} \wedge \overline{y} = \overline{c}_{init}$ where $\chi_{init}(\overline{w})$ is made of inequalities of the form $w_i \ll w_j$ (with $i, j \in \{1, ..., p\}$).
- A *τ-action transition* f is now of the form $\langle s, \xi, \psi, \rho, s' \rangle$. The additional component ξ, called *data update relation*, is a subset of $W \times W$ containing inequalities of the form $w_i \ll w_j, w'_i \ll w_j, w_i \ll w'_j, w'_i \ll w'_j$ (with $i, j \in \{1, ..., p\}$).
- The *action successor relation via* f, again written $\rightarrow_f$, is defined by:
$(\sigma, \overline{w}, \overline{\tau}, \overline{y}) \rightarrow_f (\sigma', \overline{w}', \overline{\tau}, \overline{y}')$ iff
$\sigma = s \wedge \sigma' = s' \wedge \xi(\overline{w}, \overline{w}') \ \wedge\ J_\sigma(\overline{\tau}, \overline{y}) \ \wedge\ J_{\sigma'}(\overline{\tau}, \overline{y}') \ \wedge\ \psi(\overline{\tau}, \overline{y}) \ \wedge\ \overline{y}' = \rho(\overline{\tau}, \overline{y})$.
- The *delay successor relation*, $\rightarrow_\delta^\tau$, is defined by
$(\sigma, \overline{w}, \overline{\tau}, \overline{y}) \rightarrow_\delta^\tau (\sigma, \overline{w}, \overline{\tau'}, \overline{y})$ iff
$0 \leq \tau_0 \leq \tau'_0 \ \wedge\ 0 \leq \overline{y} \leq \tau_m \wedge J_\sigma(\overline{\tau}, \overline{y}) \ \wedge\ J_\sigma(\overline{\tau}', \overline{y}) \ \wedge \overline{\tau} = \tau_0 + \overline{c} \wedge \overline{\tau}' = \tau'_0 + \overline{c}$.

The other successor relations are defined in the natural way. Definition 2.2 (τ-class) is extended by replacing $\phi_k(\overline{y}, \overline{\tau})$ with $\chi_k(\overline{w}) \wedge \phi_k(\overline{y}, \overline{\tau})$ where $\chi_k(\overline{w})$ contains inequalities of the form $w_i \ll w_j$ (with $i, j \in \{1, ..., p\}$). The results of section 2.3 still hold in this context. In the same manner, definitions and results for the parametric procedure (section 3.2) naturally extend to the framework of locations with data variables.

4 Experimental Results

We follow two approaches for the implementation of the procedures Pre^* described above. The first one is based on Revesz's procedure for computing least fixed-points of logic programs with arithmetical constraints [17]. The second approach is based on hybrid automata [3,2,16].

4.1 Revesz based approach

In the abstract procedure Pre^*_{abs}, the only constraints that are manipulated are constraints of the form $z_i \ll z_j$, where z_i, z_j are real variables and $\ll \in \{<, \leq\}$. More precisely, the successor relation between two symbolic states, say $\overline{q}$ and $\overline{q}'$ via an automaton $\mathcal{A}$ is such that $\overline{q} \rightarrow_{\mathcal{A}} \overline{q}'$ iff $\zeta(\overline{q}, \overline{q}')$, where ζ is a conjunction of inequalities of the form $q_i \ll q_j, q_i \ll q'_j, q'_i \ll q_j, q'_i \ll q'_j$. Revesz has shown that the class made of conjunctions of inequalities of the form $q_i \ll q_j$ is closed under such a successor relation $\rightarrow_{\mathcal{A}}$, and that $\rightarrow^*_{\mathcal{A}}$ always terminates. [2] Actually the result holds more generally in the case where all the manipulated constraints are of the form $z_i + d \ll z_j$ where d denotes a natural number. It is possible to adapt Revesz's procedure in order to handle constraints of the form $z_i \ll z_j + d$ in the special case of timed automata (see [7]). Such a system, called GAP, was obtained by modifying an existing implementation of Revesz's procedure. [3]

4.2 Hybrid automata approach

Our extended τ-automata can be considered as linear hybrid automata (i.e., extended timed automata where temporal variables may evolve at different rates [3]) where $\overline{w}$ and $\overline{y}$ are temporal variables of constant evolution slope 0, and variables $\tau_0, \tau_1, ..., \tau_m$ are temporal variables of constant slope 1, which are never reset. One can therefore compute the exact backward reachability fixed-point Pre^* using existing hybrid automata analysis tools such as HyTech [10]. In the case of **parametric** timed automata, it is also possible with HyTech to simulate our **abstract** backward reachability construction by considering $\tau_0, \tau_1, ..., \tau_m$ as temporal variables with slopes varying in $[0, +\infty[$ (instead of being equal to 1), and adding constraint $\tau_0 < \tau_1 < ... < \tau_m$ as location invariants. Note that, with HyTech, we can even try and compute the **exact** backward reachability fixed-point in the case of parametric timed automata, by assigning slopes of τ_i to be equal to 1 and declaring c_i as parameters instead of constants, but there is not any longer guarantee of termination.

4.3 ABR conformance example

We have experimented these two approaches for verifying the correctness of a reactive program , called "ABR conformance", designed for controlling dataflow rates on ATM networks. The program has been designed by Jean-Francois Monin and Francis Klay at French Center of Telecommunications (CNET), and is now part of an ITU norm [15]. The correctness property Π is formalized as a set of three properties Π_1, Π_2, Π_3. Monin and Klay have proved the correctness property Π by finding by hand a collection of auxiliary assertions $\Pi_4, ..., \Pi_{10}$

[2] In Revesz's original work, the variables domain is $\mathbb{Z}$, but his results extend easily to $\mathbb{R}$ (see [7]).

[3] The original implementation has already been used for verifying safety properties of concurrent systems, but in a different context [8].

additionally to Π_1, Π_2, Π_3, and by showing that the conjunction $\Pi_1 \wedge ... \wedge \Pi_{10}$ is a global invariant (fixed-point) of their program. (The global invariance proof has been checked using the proof-assistant Coq [13].) The program, translated under the form of an extended $\overline{\tau}$-automaton, is made of 20 transitions, and contains three temporal variables y_1, y_2, y_3, five data variables w_1, w_2, w_3, w_4, w_5 and two parameters c_1, c_2. The correctness proof is attempted by performing backward reachability analysis starting successively from three formulas $\phi_{\neg\Pi_1}, \phi_{\neg\Pi_2}, \phi_{\neg\Pi_3}$, which correspond to $\neg\Pi_1, \neg\Pi_2, \neg\Pi_3$ respectively, and by checking that the initial state is never reached. The full description of the $\overline{\tau}$-automaton and formulas $\phi_{\neg\Pi_1}, \phi_{\neg\Pi_2}, \phi_{\neg\Pi_3}$ are given in appendix. Experiments have been conducted on a SUN station ULTRA-1 with 64 Megabytes of RAM memory. We discuss hereafter the results obtained by backward reachability construction starting from $\phi_{\neg\Pi_1}$, but the results are similar for $\phi_{\neg\Pi_2}$ and $\phi_{\neg\Pi_3}$.

The **exact parametric** backward reachability construction of $Pre^*(\phi_{\neg\Pi_1})$ with HyTech (declaring c_1, c_2 as parameters, and τ_0, τ_1, τ_2 as temporal variables of slope 1), **fails** by exhaustion of memory. It fails also by exhaustion of memory if only c_1 or c_2 is declared as a parameter, or if forward reachability is performed instead of backward. The **abstract parametric** backward reachability construction of $Pre^*_{abs}(\phi_{\neg\Pi_1})$ with HyTech (declaring c_1, c_2 as parameters and τ_0, τ_1, τ_2 as temporal variables of slope in $[0, +\infty[$ with constraint $\tau_0 < \tau_1 < \tau_2$) **succeeds** and proves the property in 202 seconds after generation of a fixed-point made of 270 conjunctions of inequalities. The abstract parametric backward reachability construction also succeeds with GAP and proves the property in 2400 seconds after generation of a fixed-point made of 460 conjunctions of inequalities. Other experiences with various **instantiated values** of parameters c_1, c_2 confirm that, on this problem, HyTech is about 10 times faster than GAP. When the ratio of values c_2/c_1 is larger than 200 (resp. 20), HyTech (resp. GAP) runs out of memory.

As a recapitulation, the parametric abstraction scheme of section 3 has allowed us to prove **fully automatically** the correctness of the ABR conformance program both with HyTech and GAP, but HyTech has turned out to be much more efficient than GAP. Without abstraction, HyTech fails to prove the correctness of the parametric program by exact (backward or forward) reachability analysis. Further experiences on other protocol verification problems are in progress.

Acknowledgement. We are very grateful to François Laroussinie for numerous helpful discussions on timed automata.

References

1. R. Alur, C. Courcoubetis and D. Dill. "Model-Checking in Dense Real-Time". *Information and Computation 104:1*, 1993, pp. 2–34.

2. 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*, 1995, pp. 3–34.
3. R. Alur, C. Courcoubetis, T.A. Henzinger and P.-H. Ho. "Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems". *Hybrid Systems I*, LNCS 736, 1993, pp. 209–229.
4. R. Alur and D. Dill. "Automata for Modeling Real-Time Systems". *Proc. 17th ICALP*, LNCS 443, Springer-Verlag, 1990, pp. 322–335.
5. R. Alur, T.A. Henzinger, M. Vardi. "Parametric real-Time reasoning". *Proc. 25th Annual ACM Symp. on Theory of Computing (STOC)*, 1993, pp. 592–601.
6. C. Daws and S. Tripakis. "Model-Checking of real-time reachability properties using abstractions". *4th Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 98)*, LNCS 1384, 1998, pp. 313–329.
7. L. Fribourg. "A Closed-Form Evaluation for Extended Timed Automata". *Technical Report LSV-98-2*, CNRS & Ecole Normale Supérieure de Cachan, March 1998. (http://www.lsv.ens-cachan.fr/Publis/)
8. L. Fribourg and J. Richardson. "Symbolic Verification with Gap-Order Constraints". *Proc. 6th Intl. Workshop on Logic Program Synthesis and Transformation (LOPSTR)*, LNCS 1207, 1996, pp. 20–37.
9. N. Halbwachs, F. Lagnier and C. Ratel. "Synchronous Observers and The Verification of reactive Systems". *Proc. 3rd Conf. on Algebraic Methodology and Software Technology (AMAST)*, Workshop in Computing, Springer-Verlag, 1993.
10. T. Henzinger, P.-H. Ho and H. Wong-Toi. "A user Guide to HYTECH". *Proc. TACAS'95*, LNCS 1019, Springer-Verlag, 1995, pp. 41–71.
11. T. Henzinger, P.W. Kopke, A. Puri and P. Varaiya. "What's Decidable about Hybrid Automata?", *Proc. 27th Annual ACM Symp. on Theory of Computing (STOC)*, 1995, pp.373–382.
12. T. Henzinger, X. Nicollin, J. Sifakis and S. Yovine. "Symbolic Model Checking for Real-Time Systems", *IEEE Symp. on Logic in Computer Science*, 1992, pp. 394–406.
13. G. Huet, G. Kahn and C. Paulin-Mohring. "The Coq Proof Assistant - A tutorial, Version 6.1". *Technical Report 204*, INRIA, August 1997.
14. L. Lamport. "Proving the Correctness of Multiprocess Programs". *IEEE Trans. on Softw. Eng. 3:2*, 1977, pp. 125–143.
15. J.-F. Monin and F. Klay. "Correction of an Algorithm for ABR Conformance". *Technical Report*, CNET, Lannion, December 1996.
16. X. Nicollin, A. Olivero, J. Sifakis and S. Yovine. "An Approach to the Description and Analysis of Hybrid Systems". *Hybrid Systems I*, LNCS 736, 1993, pp. 149–178.
17. P.Z. Revesz. "A Closed-Form Evaluation for Datalog Queries with Integer (Gap)-Order Constraints", *Theoretical Computer Science*, 1993, vol. 116, pp. 117-149.

Appendix: ABR Conformance Example

We consider formulas of the form $\phi(\sigma, \overline{w}, \overline{\tau}, \overline{y})$ where $\overline{w}$ is $(w_1, w_2, w_3, w_4, w_5)$, $\overline{\tau}$ is (τ_0, τ_1, τ_2), and $\overline{y}$ is (y_1, y_2, y_3). There is only one location s_0 in S. The associated $\overline{\tau}$-invariant is:

$J_{s_0} \quad : \quad \tau_0 \leq y_1 \leq \tau_2 \ \wedge \ \tau_0 \leq y_2 \leq \tau_2 \ \wedge \ \tau_0 \leq y_3 \leq \tau_2.$

The initial formula is: $\phi_{init}(s_0, \overline{w}, \overline{c}, \overline{c}_{init})$ with constraint $\chi_{init}(\overline{w}) \Leftrightarrow w_1 = w_2 = w_3 = w_4$, and $\overline{c} = (c_0, c_1, c_2)$, $\overline{c}_{init} = (c_2, c_2, c_2)$.

The formulas $\phi_{\neg\Pi_i}$ (corresponding to the negation of properties to be proven) are:

$\phi_{\neg\Pi_1}: \quad \sigma = s_0 \wedge w_5 > w_4 \wedge \tau_0 \leq y_3 < y_1$

$\phi_{\neg\Pi_2}: \quad \sigma = s_0 \wedge w_5 > w_1 \wedge y_1 \leq y_3 < y_2$

$\phi_{\neg\Pi_3}: \quad \sigma = s_0 \wedge w_5 > w_2 \wedge y_2 \leq y_3$

There are 20 transitions, labeled $1a, 1b, 2, 3, 4, 5, 6, (9a.7), (9a.8), 9b$ and $1a', 1b', 2', 3', 4', 5', 6', (9a.7)', (9a.8)', 9b'$. The associated guards $\psi(\overline{\tau}, \overline{y})$, reset operations $\rho(\overline{\tau}, y_i)$, and data updates $\xi(\overline{w})$, are defined as follows:

$\psi_{1a} \Leftrightarrow \tau_0 < y_1 = y_2 < \tau_1 \wedge y_3 < \tau_1, \quad \xi_{1a} \Leftrightarrow \xi'_{1a} \wedge w'_5 = w_5,$
$\rho_{1a}(\overline{\tau}, y_1) = y_1,\ \rho_{1a}(\overline{\tau}, y_2) = \tau_1,\ \rho_{1a}(\overline{\tau}, y_3) = y_3.$
$\psi_{1a'} \Leftrightarrow \tau_0 < y_1 = y_2 < \tau_1 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{1a'} \Leftrightarrow \xi'_{1a} \wedge w'_5 = max(w_5, w'_2),$
$\rho_{1a'}(\overline{\tau}, y_1) = y_1,\ \rho_{1a'}(\overline{\tau}, y_2) = \tau_1,\ \rho_{1a'}(\overline{\tau}, y_3) = y_3.$
$\psi_{1b} \Leftrightarrow \tau_0 < y_1 < \tau_1 < y_2 \wedge y_3 < \tau_1, \quad \xi_{1b} \Leftrightarrow \xi'_{1b} \wedge w'_5 = w_5,$
$\rho_{1b}(\overline{\tau}, y_1) = y_1,\ \rho_{1b}(\overline{\tau}, y_2) = \tau_1,\ \rho_{1b}(\overline{\tau}, y_3) = y_3.$
$\psi_{1b'} \Leftrightarrow \tau_0 < y_1 < \tau_1 < y_2 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{1b'} \Leftrightarrow \xi'_{1b} \wedge w'_5 = max(w_5, w'_2),$
$\rho_{1b'}(\overline{\tau}, y_1) = y_1,\ \rho_{1b'}(\overline{\tau}, y_2) = \tau_1,\ \rho_{1b'}(\overline{\tau}, y_3) = y_3.$
$\psi_2 \Leftrightarrow \tau_0 < y_1 < y_2 \leq \tau_1 \wedge y_3 < \tau_1, \quad \xi_2 \Leftrightarrow \xi'_2 \wedge w'_5 = w_5,$
$\rho_2(\overline{\tau}, y_1) = y_1,\ \rho_2(\overline{\tau}, y_2) = y_2,\ \rho_2(\overline{\tau}, y_3) = y_3.$
$\psi_{2'} \Leftrightarrow \tau_0 < y_1 < y_2 \leq \tau_1 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{2'} \Leftrightarrow \xi'_2 \wedge w'_5 = max(w_5, w'_2),$
$\rho_{2'}(\overline{\tau}, y_1) = y_1,\ \rho_{2'}(\overline{\tau}, y_2) = y_2,\ \rho_{2'}(\overline{\tau}, y_3) = y_3.$
$\psi_3 \Leftrightarrow \tau_1 \leq y_1 \wedge y_3 < \tau_1, \quad \xi_3 \Leftrightarrow \xi'_3 \wedge w'_5 = w_5,$
$\rho_3(\overline{\tau}, y_1) = \tau_1,\ \rho_3(\overline{\tau}, y_2) = \tau_1,\ \rho_3(\overline{\tau}, y_3) = y_3.$
$\psi_{3'} \Leftrightarrow \tau_1 \leq y_1 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{3'} \Leftrightarrow \xi'_3 \wedge w'_5 = max(w_5, w'_2),$
$\rho_{3'}(\overline{\tau}, y_1) = \tau_1,\ \rho_{3'}(\overline{\tau}, y_2) = \tau_1,\ \rho_{3'}(\overline{\tau}, y_3) = y_3.$
$\psi_4 \Leftrightarrow \tau_1 \leq y_1 \wedge y_3 < \tau_1, \quad \xi_4 \Leftrightarrow \xi'_4 \wedge w'_5 = w_5,$
$\rho_4(\overline{\tau}, y_1) = y_1,\ \rho_4(\overline{\tau}, y_2) = y_1,\ \rho_4(\overline{\tau}, y_3) = y_3.$
$\psi_{4'} \Leftrightarrow \tau_1 \leq y_1 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{4'} \Leftrightarrow \xi'_4 \wedge w'_5 = max(w_5, w'_2),$
$\rho_{4'}(\overline{\tau}, y_1) = y_1,\ \rho_{4'}(\overline{\tau}, y_2) = y_1,\ \rho_{4'}(\overline{\tau}, y_3) = y_3.$
$\psi_5 \Leftrightarrow \tau_0 < y_1 \wedge y_3 < \tau_1, \quad \xi_5 \Leftrightarrow \xi'_5 \wedge w'_5 = w_5,$
$\rho_5(\overline{\tau}, y_1) = y_1,\ \rho_5(\overline{\tau}, y_2) = \tau_2,\ \rho_5(\overline{\tau}, y_3) = y_3.$
$\psi_{5'} \Leftrightarrow \tau_0 < y_1 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{5'} \Leftrightarrow \xi'_5 \wedge w'_5 = max(w_5, w'_2),$
$\rho_{5'}(\overline{\tau}, y_1) = y_1,\ \rho_{5'}(\overline{\tau}, y_2) = \tau_2,\ \rho_{5'}(\overline{\tau}, y_3) = y_3.$
$\psi_6 \Leftrightarrow \tau_0 < y_1 \wedge y_3 < \tau_1, \quad \xi_6 \Leftrightarrow \xi'_6 \wedge w'_5 = w_5,$
$\rho_6(\overline{\tau}, y_1) = y_1,\ \rho_6(\overline{\tau}, y_2) = y_2,\ \rho_6(\overline{\tau}, y_3) = y_3.$
$\psi_{6'} \Leftrightarrow \tau_0 < y_1 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{6'} \Leftrightarrow \xi'_6 \wedge w'_5 = max(w_5, w'_2),$
$\rho_{6'}(\overline{\tau}, y_1) = y_1,\ \rho_{6'}(\overline{\tau}, y_2) = y_2,\ \rho_{6'}(\overline{\tau}, y_3) = y_3.$
$\psi_{(9a.7)} \Leftrightarrow \tau_0 = y_1 = y_2 \wedge y_3 < \tau_1, \quad \xi_{(9a.7)} \Leftrightarrow \xi'_{(9a.7)} \wedge w'_5 = w_5,$
$\rho_{(9a.7)}(\overline{\tau}, y_1) = \tau_1,\ \rho_{(9a.7)}(\overline{\tau}, y_2) = \tau_1,\ \rho_{(9a.7)}(\overline{\tau}, y_3) = y_3.$
$\psi_{(9a.7)'} \Leftrightarrow \tau_0 = y_1 = y_2 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{(9a.7)'} \Leftrightarrow \xi'_{(9a.7)} \wedge w'_5 = max(w_5, w'_2),$
$\rho_{(9a.7)'}(\overline{\tau}, y_1) = \tau_1,\ \rho_{(9a.7)'}(\overline{\tau}, y_2) = \tau_1,\ \rho_{(9a.7)'}(\overline{\tau}, y_3) = y_3.$
$\psi_{(9a.8)} \Leftrightarrow \tau_0 = y_1 = y_2 \wedge y_3 < \tau_1, \quad \xi_{(9a.8)} \Leftrightarrow \xi'_{(9a.8)} \wedge w'_5 = w_5,$
$\rho_{(9a.8)}(\overline{\tau}, y_1) = \tau_2,\ \rho_{(9a.8)}(\overline{\tau}, y_2) = \tau_2,\ \rho_{(9a.8)}(\overline{\tau}, y_3) = y_3.$
$\psi_{(9a.8)'} \Leftrightarrow \tau_0 = y_1 = y_2 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{(9a.8)'} \Leftrightarrow \xi'_{(9a.8)} \wedge w'_5 = max(w_5, w'_2),$
$\rho_{(9a.8)'}(\overline{\tau}, y_1) = \tau_2,\ \rho_{(9a.8)'}(\overline{\tau}, y_2) = \tau_2,\ \rho_{(9a.8)'}(\overline{\tau}, y_3) = y_3.$
$\psi_{9b} \Leftrightarrow \tau_0 = y_1 < y_2 \wedge y_3 < \tau_1, \quad \xi_{9b} \Leftrightarrow \xi'_{9b} \wedge w'_5 = w_5,$
$\rho_{9b}(\overline{\tau}, y_1) = y_2,\ \rho_{9b}(\overline{\tau}, y_2) = y_2,\ \rho_{9b}(\overline{\tau}, y_3) = y_3.$
$\psi_{9b'} \Leftrightarrow \tau_0 = y_1 < y_2 \wedge \tau_1 \leq y_3 < \tau_2, \quad \xi_{9b'} \Leftrightarrow \xi'_{9b} \wedge w'_5 = max(w_5, w'_2),$
$\rho_{9b'}(\overline{\tau}, y_1) = y_2,\ \rho_{9b'}(\overline{\tau}, y_2) = y_2,\ \rho_{9b'}(\overline{\tau}, y_3) = y_3.$

with:

$$\xi'_{1a} \;\Leftrightarrow\; \xi'_{1b} \;\Leftrightarrow\; \xi'_2 \;\Leftrightarrow\; w_3 \leq w'_2 = w'_3 \wedge w'_1 = w_1 \wedge w'_4 = w_4$$
$$\xi'_3 \;\Leftrightarrow\; w_3 \leq w'_3 = w'_2 = w'_1 \wedge w_4 = w'_4$$
$$\xi'_4 \;\Leftrightarrow\; w_3 \leq w'_3 = w'_2 = w'_1 < w_4 = w'_4$$
$$\xi'_5 \;\Leftrightarrow\; w'_2 < w_3 = w'_3 = w'_1 \wedge w'_2 < w_2 \wedge w'_4 = w_4$$
$$\xi'_6 \;\Leftrightarrow\; w_2 \leq w'_2 < w_3 = w'_3 = w'_1 \wedge w'_4 = w_4$$
$$\xi'_{(9a.7)} \;\Leftrightarrow\; w'_4 = w_1 < w'_2 = w'_1 = w'_3$$
$$\xi'_{(9a.8)} \;\Leftrightarrow\; w'_2 = w'_3 = w'_1 \leq w_1 = w'_4$$
$$\xi'_{9b} \;\Leftrightarrow\; w'_1 = w'_2 = w'_3 = w_2 \wedge w'_4 = w_1$$