Composing Reachability Analyses of Hybrid Systems for Safety and Stability*

Sergiy Bogomolov, Corina Mitrohin, and Andreas Podelski

University of Freiburg
Department of Computer Science
Freiburg, Germany
{bogom,mitrohin,podelski}@informatik.uni-freiburg.de

Abstract. We present a method to enhance the power of a given reachability analysis engine for hybrid systems. The method works by a new form of composition of reachability analyses, each on a different relaxation of the input hybrid system. We present preliminary experiments that indicate its practical potential for checking safety and stability.

1 Introduction

A standard technique in programming languages is to improve the precision of the analysis of a program by preceding it with an auxiliary analysis. The auxiliary analysis is in general used to infer the validity of an invariant at a specific program location. For example, an interval analysis is used to infer a lower and an upper bound on the possible values of a program variable at a given program location, which means that an assertion of the form $l \leq x \leq u$ is a valid invariant for the program location. It is hence safe to use this invariant for refining the abstract program used for a subsequent program analysis. A simple way to refine the abstract program is to insert an `assume` statement (e.g., `assume(l<= x <= u)`).

Unfortunately, it is not clear how one can directly transfer this technique from programs to hybrid systems. In a hybrid system, where the value of a variable changes not only through updates in transitions from one location to another but evolves dynamically in one location, an invariant in the form of a state assertion associated with a program location does not seem useful for improving a subsequent analysis. For one thing, an invariant must account for a continuum of states. For another, a state assertion does not seem suitable to describe the interdependence between the possible values of continuous variables x and y whose evolution is prescribed by differential equations.

In essence, the problem is to incorporate the result of the reachability analysis of one abstraction into the reachability analysis of another abstraction of a given

* This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center Automatic Verification and Analysis of Complex Systems (SFB/TR 14 AVACS). See `www.avacs.org` for more information.

A. Bouajjani and W.-N. Chin (Eds.): ATVA 2010, LNCS 6252, pp. 67–81, 2010.

hybrid system. In this paper, we propose to base the solution of this problem on the notion of *dwell time*. The concept of dwell time has already shown its usefulness for the analysis of hybrid systems (see our discussion of related work further below). It refers to the amount of time which the hybrid system can resp. must spend ('dwell') in a location between an entry and an exit. The mutual interdependence between the possible values of continuous variables x and y can be approximated via bounds on the the dwell time, i.e., bounds on the possible values of x propagate to bounds on the dwell time which again propagate to bounds on the possible values of y. We propose to use an auxiliary 'dwell time variable' d for interfacing a reachability analysis with an auxiliary analysis. We encode the result of the auxiliary analysis by bounds on the possible values of d in reachable states of an (in general coarse) abstraction of the hybrid system. We use these bounds to improve the precision of the reachability analysis of another (in some sense, complementary) abstraction as follows. We add the dwell time variable d as a continuous variable (in addition to other continuous variables, in case the abstraction is expressed by a hybrid system), and we refine the abstraction by constraints on d.

In summary, we present new methods both for inferring and for exploiting dwell time bounds. As a consequence, we obtain an approach where one can incorporate the result of a first, auxiliary reachability analysis into a second reachability analysis in order to refine its abstraction and thus improve its precision. Our approach is generic in that it uses reachability analysis as a black-box method. We present initial experiments that indicate its practical potential for checking safety and stability.

2 Hybrid Systems, Relaxation, Refinement

A *hybrid system* is formally a tuple $\mathcal{H} = (Loc, V, Init, R^{\mathsf{cont}}, R^{\mathsf{disc}}, Inv)$ defining

- the finite set of locations Loc,
- the set of continuous variables $V = \{v_1, \ldots, v_n\}$,
- the initial condition, given by the constraint $Init(\ell)$ for each location ℓ,
- the continuous transition relation, given by the expression $e = R^{\mathsf{cont}}(\ell)(v)$ for each continuous variable v and each location ℓ; the expression e (in the variables $v_1, \ldots, v_n$) is used in the differential equation $\dot{v} = e$ that defines the flow of the continuous variable v in the location ℓ,
- the discrete transition relation, given by a set R^{disc} of transitions; a transition is formally a tuple (ℓ, g, ξ, ℓ') defining
  - the source location ℓ and the target location ℓ',
  - the guard, given by a constraint g,
  - the update, given by a (possibly empty) set ξ of assignments $v := e$ of expressions to continuous variables,
- the invariant, given by the constraint $Inv(\ell)$ for each location ℓ.

An example of a hybrid system is given in Figure 1.

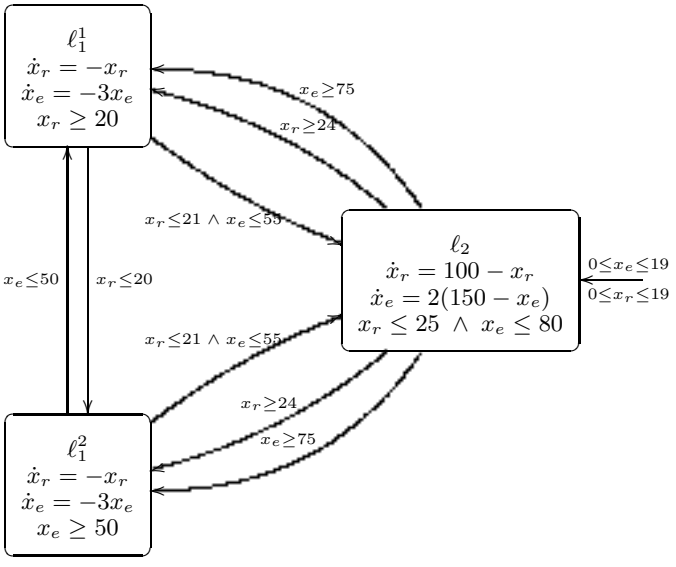

Fig. 1. Example hybrid system, modeling a temperature controller with one internal engine. The temperature of the plant is controlled through a thermostat which turns the engine off and on, depending on the temperature of the room (modeled by the continuous variable x_r) and the temperature of the engine (modeled by the continuous variable x_e). We model the mode 'off' by the two locations ℓ_1^1 and ℓ_1^2, and the mode 'on' by the location ℓ_2.

A *state* of the hybrid system $\mathcal{H}$ is a tuple $(\ell, v_1, \ldots, v_n)$ consisting of a location ℓ in *Loc* and values of the continuous variables in V.

The hybrid system can be represented by a labeled graph (as in Figure 1), where the set of nodes is the set of locations *Loc*, and the set of edges is defined by the discrete transition relation, i.e.,

$$E = \{(\ell, \ell') \mid \exists (\ell, g, \xi, \ell') \in R^{\mathsf{disc}}\}.$$

We now give the formal definition of the semantics of a hybrid system, in the form of the set of its runs. We write $\mathcal{T}$ for the set of all continuous time points which are denoted by non-negative real values, i.e., $\mathcal{T} = \mathbb{R}_0^+$. A run ρ assigns to every time point t in $\mathcal{T}$ a location and a valuation of the variables v in V. Formally, a run ρ is a tuple

$$\rho = (\hat{\ell}, \hat{v}_1, \ldots, \hat{v}_n)$$

of functions $\hat{\ell} : \mathcal{T} \to \mathit{Loc}$ (for the current location) and $\hat{v} : \mathcal{T} \to \mathbb{R}$ (for the current value of the variable v in V), such that there exists an infinite sequence of switching time points,

$$(\tau_i)_{i \in \omega} \in \mathcal{T}^\omega$$

which starts in 0 and is strictly increasing, i.e., $\tau_0 = 0$ and $\tau_i < \tau_{i+1}$, such that the following five conditions hold:

- "non-zenoness"
$$\forall t \in \mathcal{T}\ \exists i :\ t \leq \tau_i \tag{1}$$
- "switching time"
$$\forall i\ \forall t \in [\tau_i, \tau_{i+1}) : \hat{\ell}(t) = \hat{\ell}(\tau_i) \tag{2}$$
- "continuous evolution"
$$\forall v \in V\ \forall i\ \forall t \in [\tau_i, \tau_{i+1}) : \quad \frac{d}{dt}\hat{v}(t) = e[\hat{v}_1, \ldots, \hat{v}_n](t) \tag{3}$$
where $e = R^{\mathsf{cont}}(\ell)(v)$ with $\ell = \hat{\ell}(\tau_i)$,
- "invariants"
$$\forall i\ \forall t \in [\tau_i, \tau_{i+1}) :\ (\hat{v}_1(t), \ldots, \hat{v}_n(t)) \models \mathit{Inv}(\ell) \tag{4}$$
where $\ell = \hat{\ell}(\tau_i)$,
- "discrete transition firing"
$$\begin{aligned} \forall i\ \exists\ (\ell, g, \xi, \ell') \in R^{\mathsf{disc}} :\ & \\ & \hat{\ell}\,(\tau_i) = \ell \\ & \hat{\ell}\,(\tau_{i+1}) = \ell' \\ & \exists\ \sigma : V \to \mathbb{R}\ \forall v \in V : \\ & \qquad \sigma(v) = \lim_{u \to \tau_{i+1}} \hat{v}(u) \\ & \qquad \sigma \models g \\ & \qquad \hat{v}(\tau_{i+1}) = \begin{cases} \sigma(e), & \text{if } v := e \in \xi \\ \sigma(v), & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Condition (1) states that we do not allow Zeno behavior. The time sequence $(\tau_i)_{i \in \omega}$ identifies the time points where location switches may occur, which is expressed in Condition (2). Only at those points discrete transitions may be taken. Condition (3) expresses that the dynamics of the continuous variables obeys their respective differential equations. Condition (4) requires that for each location the valuation of continuous variables satisfies the local invariant while staying in that location. Condition (5) expresses that whenever a discrete transition is taken, variables may be assigned new values, obtained by evaluating the right-hand side of the respective assignment using the previous values of variables. If there is no such assignment, the variable maintains its previous value, which is determined by taking the limit of the trajectory of the variable as t converges to the switching time τ_{i+1}.

Definition 1 (Relaxation, Refinement). *A hybrid system $\mathcal{H}^{\#}$ is a* relaxation *of $\mathcal{H}$, written $\mathcal{H}^{\#} \sqsupseteq \mathcal{H}$, if it has more runs than $\mathcal{H}$, i.e.,*

$$\mathrm{Run}(\mathcal{H}^{\#}) \supseteq \mathrm{Run}(\mathcal{H}).$$

The system $\mathcal{H}$ is called, in turn, a refinement *of $\mathcal{H}^{\#}$.*

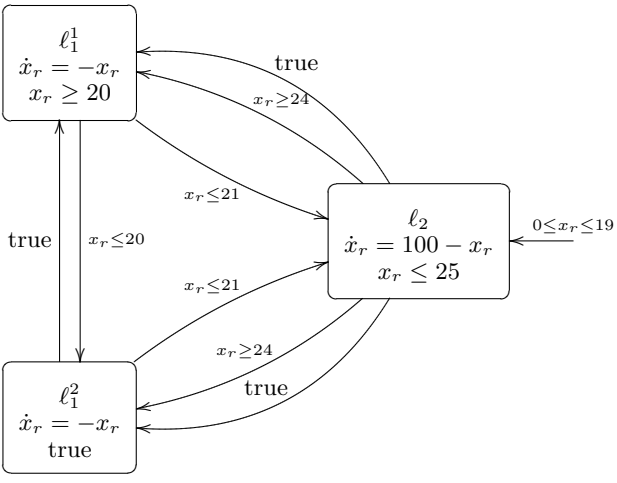

Fig. 2. The relaxation of the heating system from Figure 1 obtained by eliminating the continuous variable x_e

By definition, each (reachable) location ℓ and edge (ℓ, ℓ') of $\mathcal{H}$ belongs also to $\mathcal{H}^{\#}$. There are several ways to construct a relaxation $\mathcal{H}^{\#}$ from $\mathcal{H}$, for example, by projection or by weakening of constraints; see, e.g. [8]. Figure 2 presents the relaxation of the heating system from Figure 1 obtained by eliminating the continuous variable x_e.

3 Dwell Time Bounds

A family of *dwell time bounds* $dtb = (dtb_{\mathsf{low}}, dtb_{\mathsf{high}})$ for a hybrid system $\mathcal{H}$ is given by a constant

$$dtb_{\mathsf{low}}(\ell, \ell') = c_{(\ell,\ell')}$$

for each transition from the location ℓ to the location ℓ', and a constant

$$dtb_{\mathsf{high}}(\ell) = c_\ell$$

for each location ℓ. (Here, we index a dwell time guard by a pair of locations, and not by the transition. We assume wlog. that a transition is unique for its pair of source and target locations.)

Definition 2 (Validity of dwell time bounds). *The dwell time bound $dtb_{\mathsf{low}}(\ell, \ell') = c_{(\ell,\ell')}$ for the transition from the location ℓ to the location ℓ' is valid if in every run of $\mathcal{H}$, the time spent between the entry of the location ℓ and its exit towards the location ℓ' is always greater than or equal to $c_{(\ell,\ell')}$.*

The dwell time bound $dtb_{\mathsf{high}}(\ell) = c_\ell$ for the location ℓ is valid if in every run of $\mathcal{H}$, the time spent between the entry and the exit of the location ℓ is always smaller than or equal to c_ℓ.

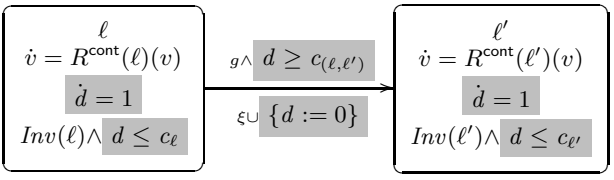

Fig. 3. $\mathsf{DTR}(\mathcal{H}, dtb)$, the dwell time refinement of $\mathcal{H}$ for a given family of dwell time bounds dtb valid for $\mathcal{H}$, is obtained from $\mathcal{H}$ by a transformation that adds the highlighted items; among those, the dwell time constraints $d \leq c_\ell$ and $d \geq c_{(\ell,\ell')}$

A family of dwell time bounds $dtb = (dtb_{\mathsf{low}}, dtb_{\mathsf{high}})$ for a given hybrid system $\mathcal{H}$ is valid if each of its dwell time bounds are valid.

That is, the valid dwell time bound $dtb_{\mathsf{low}}(\ell, \ell')$ for the transition from the location ℓ to the location ℓ' is a lower bound for the time spent in the location ℓ, or, equivalently, for the length of the interval $[\tau_i, \tau_{i+1}]$ formed by two consecutive switching time points for a discrete transition into ℓ and a discrete transition from ℓ to ℓ'.

Also, the valid dwell time bound $dtb_{\mathsf{high}}(\ell)$ is an upper bound for the time spent in the location ℓ, or, equivalently, for the length of the interval $[\tau_i, \tau_{i+1}]$ formed by two consecutive switching time points for a discrete transition into ℓ respectively away from ℓ.

4 Dwell Time Refinement

Dwell Time Refinement for $\mathcal{H}$ ($\mathsf{DTR}(\mathcal{H}, dtb)$). Given a family $dtb = (dtb_{\mathsf{low}}, dtb_{\mathsf{high}})$ of dwell time bounds valid for $\mathcal{H}$, we construct a new hybrid system $\mathsf{DTR}(\mathcal{H}, dtb)$ informally as follows (see also Figure 3).

- We add a continuous variable d.
- In each location, we set the slope of d to 1 (i.e., d evolves like a clock).
- In each discrete transition, the variable d is reset to 0 (i.e., we add the assignment $d := 0$ to update of the transition).
- We add $d \geq dtb_{\mathsf{low}}(\ell, \ell')$ as a conjunct to the guard of the transition from the location ℓ to the location ℓ' (we call this conjunct the *dwell time guard*).
- We add $d \leq dtb_{\mathsf{high}}(\ell)$ as a conjunct to the invariant of the location ℓ (we call this conjunct the *dwell time invariant*).

Formally, we define $\mathsf{DTR}(\mathcal{H}, dtb)$ as the hybrid system

$$\mathsf{DTR}(\mathcal{H}, dtb) = (Loc, V_d, Init_d, \mathsf{R}_d^{\mathsf{cont}}, \mathsf{R}_d^{\mathsf{disc}}, Inv_d)$$

where

- the set of locations Loc is the same as in $\mathcal{H}$,

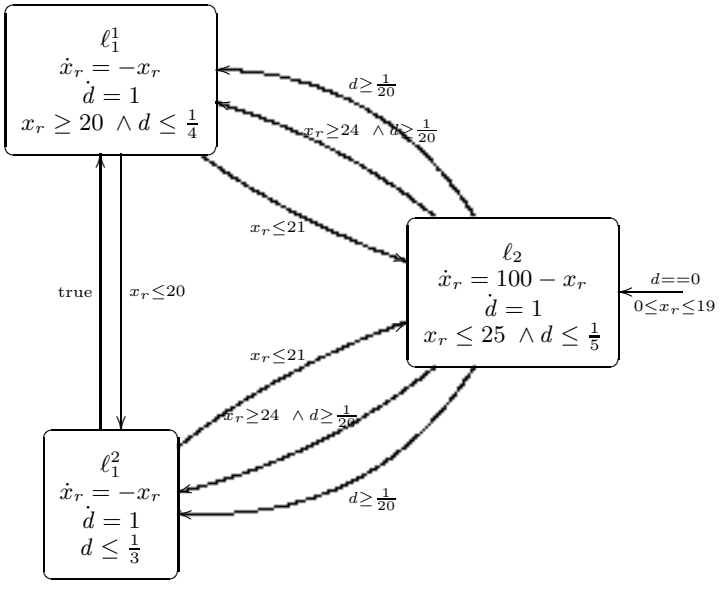

Fig. 4. $\mathsf{DTR}(\mathcal{H}^{\#}, dtb)$, the dwell time refinement of the relaxation $\mathcal{H}^{\#}$ in Figure 2 for a given family of dwell time bounds dtb valid for $\mathcal{H}$, where the dwell time bounds dtb are the ones inferred automatically by the method given in Section 5

- the set of continuous variables V_d contains the new variable d and all variables of $\mathcal{H}$,

$$V_d = \{d\} \cup V$$

- the initial condition $Init_d$ fixes the value of the new variable d to 0 in each location,

$$Init_d(\ell) \equiv Init(\ell) \wedge (d = 0)$$

- the continuous transition relation R_d^{cont} assigns constant 1 to the derivative of the new variable,

$$R_d^{\mathsf{cont}}(\ell)(d) = 1$$
$$R_d^{\mathsf{cont}}(\ell)(v) = R^{\mathsf{cont}}(\ell)(v) \ \text{ for } v \in V$$

- the discrete transition relation R_d^{disc} adds the assignment $d := 0$ to the update of each transition, and it adds the dwell time guard $d \geq dtb_{\mathsf{low}}(\ell, \ell')$ as a conjunct to the guard of the transition from ℓ to ℓ',

$$(\ell,\ g \wedge d \geq dtb_{\mathsf{low}}(\ell, \ell')\ ,\ \xi \cup \{d := 0\}\ ,\ \ell') \in R_d^{\mathsf{disc}} \quad \text{if} \quad (\ell, g, \xi, \ell') \in R^{\mathsf{disc}}$$

- the invariant Inv_d adds the dwell time invariant $d \leq dtb_{\mathsf{high}}(\ell)$ as a conjunct to the invariant of the location ℓ,

$$Inv_d(\ell) \equiv Inv(\ell) \wedge d \leq dtb_{\mathsf{high}}(\ell)$$

The definition of the validity of *dtb* as a family of dwell time bounds for $\mathcal{H}$ is equivalent to saying that $\mathsf{DTR}(\mathcal{H}, dtb)$ has the same set of runs as $\mathcal{H}$. Dwell time refinement becomes interesting when it is applied to a relaxation $\mathcal{H}^{\#}$ of $\mathcal{H}$.

Theorem 1. *If dtb is a valid family of dwell time bounds for* $\mathcal{H}$*, and* $\mathcal{H}^{\#}$ *is a relaxation of* $\mathcal{H}$*, then* $\mathsf{DTR}(\mathcal{H}^{\#}, dtb)$ *is also a relaxation of* $\mathcal{H}$*.* □

5 Inferring Dwell Time Bounds

Given a relaxation $\mathcal{H}^{\#}$ of a hybrid system $\mathcal{H}$, we define the operation $\mathsf{DTI}(\mathcal{H}^{\#})$, which infers

$$dtb = \mathsf{DTI}(\mathcal{H}^{\#}),$$

a family of dwell time bounds which is valid for $\mathcal{H}$. The operation DTI consists of applying a syntactic transformation ST to the hybrid system $\mathcal{H}^{\#}$ and then applying a reachability analysis to the resulting hybrid system $ST(\mathcal{H}^{\#})$, for every location ℓ. The syntactic transformation ST is very simple. Given the location ℓ, we add a dwell time variable d, which is a continuous varible. The dwell time variable is reset to 0 by every transition entering the location ℓ. It evolves like a clock variable (i.e., it has constant slope 1) in the location ℓ and it does not evolve in every other location (i.e., it has the constant slope 0). The syntactic transformation of $\mathcal{H}^{\#}$ thus consists of adding updates $d := 0$ to every incoming transition and adding the differential equation $\dot{d} = 1$ to the location ℓ and the differential equation $\dot{d} = 0$ to every other location.

Having computed a safe approximation of the set of reachable states of $ST(\mathcal{H}^{\#})$, we are interested only in the set

$$\mathsf{Reach}_d(\ell')$$

of values of the continuous variable d of states at the neighbor locations ℓ' of ℓ.

- We define the constant $c_{(\ell,\ell')}$ as the minimum of $\mathsf{Reach}_d(\ell')$.
- We define the constant c_ℓ as the maximum of the union of sets $\mathsf{Reach}_d(\ell')$ for all neighbor locations ℓ'.

By repeating the above computation of the dwell time guards and invariants for all locations ℓ of the hybrid system $\mathcal{H}$ we obtain a valid family of dwell time bounds *dtb*.

For concreteness, we illustrate the syntactic transformation for the particular relaxation that we use in our experiments (see Section 7). We consider the relaxation $\mathrm{Proj}(\mathcal{H}, \ell)$ of $\mathcal{H}$ by projecting $\mathcal{H}$ to one single location, say ℓ, and its successor locations ℓ'. While abstracting away all predecessors of ℓ, we collect the entry guards and the invariants of the predecessors of ℓ. By applying the syntactic transformation ST to this relaxation, we obtain the hybrid system $\mathrm{Proj_d}(\mathcal{H}, \ell)$. Informally (see also Figure 5), $\mathrm{Proj_d}(\mathcal{H}, \ell) = ST(\mathrm{Proj}(\mathcal{H}, \ell))$ consists of only the location ℓ and its *neighbor locations* ℓ', which are the target locations of transitions from ℓ, and an additional clock variable d.

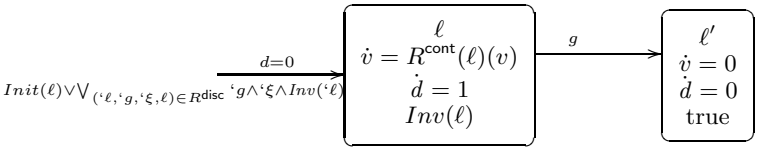

Fig. 5. Given the hybrid system $\mathcal{H}$ and the location ℓ, the figure presents the hybrid system $\mathrm{Proj_d}(\mathcal{H}, \ell)$ used for the inference of dwell time constraints by reachability analysis. The new initial condition for the location ℓ uses the disjunction of the guards $\text{`}g$ of incoming transitions in conjunction with the invariants of their source locations $\text{`}\ell$

Formally,

$$\mathrm{Proj_d}(\mathcal{H}, \ell) = (Loc_\ell, V \cup \{d\}, Init_\ell, R_\ell^{\mathsf{cont}}, R_\ell^{\mathsf{disc}}, Inv_\ell)$$

with

- the set of locations Loc_ℓ that consists of ℓ and all its successor locations ℓ',
- the set of continuous variables $V \cup \{d\}$, where d is a new clock variable,
- the initial condition for the location ℓ is augmented by the disjunction of the guards $\text{`}g$ of incoming transitions in conjunction with the invariants of their source locations $\text{`}\ell$ (pronounced "old-ell"), and the update,

$$Init_\ell(\ell) \;\equiv\; d = 0 \wedge (Init(\ell) \vee \bigvee_{(\text{`}\ell,\text{`}g,\text{`}\xi,\ell) \in R^{\mathsf{disc}}} \text{`}g \wedge \text{`}\xi \wedge Inv(\text{`}\ell))$$

- the initial condition for each neighbor location ℓ' is the Boolean constant *false*,
- the continuous transition relation R_ℓ^{cont} is given by

$$\begin{aligned} &R_\ell^{\mathsf{cont}}(\ell)(v) = R^{\mathsf{cont}}(\ell)(v) \text{ if } v \neq d \\ &R_\ell^{\mathsf{cont}}(\ell)(d) = 1 \\ &R_\ell^{\mathsf{cont}}(\ell') = 0 \quad \text{for each neighbor location } \ell' \end{aligned}$$

- the discrete transition relation R_ℓ^{disc} is the discrete transition relation R^{disc} restricted to transitions from ℓ,
- the invariant of the location ℓ is equal to $Inv(\ell)$,
- the invariant in each neighbor location ℓ' is the Boolean constant *true*.

Strictly speaking, $\mathrm{Proj}(\mathcal{H}, \ell)$ is not a relaxation of $\mathcal{H}$ (in the sense fixed in Section 2), since it has not the same set of locations and the same set of edges the set of its runs cannot be a superset of the set of runs $\mathcal{H}$. It is straightforward to extend the definition of $\mathrm{Proj}(\mathcal{H}, \ell)$ such that we obtain a relaxation in the strict sense.

6 Composing Reachability Analyses

We now define the method to sequentially compose reachability analyses. The method incorporates the result of the reachability analysis of one relaxation $\mathcal{H}_1^{\#}$

into the reachability analysis of another relaxation $\mathcal{H}_2^\#$ of a given hybrid system $\mathcal{H}$. We want to note that to get good experimental results it is crucial to choose judicious relaxations $\mathcal{H}_1^\#$ and $\mathcal{H}_2^\#$; several methods to obtain relaxations are mentioned in Section 2 and Section 5.

- The input to the method consists of two hybrid systems $\mathcal{H}_1^\#$ and $\mathcal{H}_2^\#$. The first one, $\mathcal{H}_1^\#$, embodies the auxiliary (in general, rather coarse) relaxation of $\mathcal{H}$. The second one, $\mathcal{H}_2^\#$, embodies another (in some sense, complementary) relaxation of $\mathcal{H}$. (As a consequence of our definition of relaxation, the three hybrid systems $\mathcal{H}$, $\mathcal{H}_1^\#$ and $\mathcal{H}_2^\#$ all have the same set of locations.)
- We infer $dtb_{\mathcal{H}_1^\#} = \mathsf{DTI}(\mathcal{H}_1^\#)$, a family of dwell time bounds valid for $\mathcal{H}$, by applying, repeatedly for each location ℓ, a reachability analysis to the hybrid system that is obtained from $\mathcal{H}_1^\#$ by the simple syntactic transformation described in Section 5.
- We use the family of dwell time bounds $dtb_{\mathcal{H}_1^\#}$ for the dwell time refinement of $\mathcal{H}_2^\#$; we obtain the hybrid system $\mathsf{DTR}(\mathcal{H}_2^\#, dtb_{\mathcal{H}_1^\#})$ from $\mathcal{H}_2^\#$ by the syntactic transformation described in Section 4.
- The output of the method is the result of the reachability analysis applied to the hybrid system $\mathsf{DTR}(\mathcal{H}_2^\#, dtb_{\mathcal{H}_1^\#})$.

The method is sound by Theorem 1, i.e., the output of the method is a safe approximation of the set of reachable states of the original hybrid system $\mathcal{H}$ and the following relation holds.

$$\mathcal{H} \sqsubseteq \mathsf{DTR}(\mathcal{H}_2^\#, dtb_{\mathcal{H}_1^\#}) \sqsubseteq \mathcal{H}_2^\#$$

Completeness is of theoretical interest only; trivially, for every relaxation $\mathcal{H}_1^\#$ of $\mathcal{H}$ there exists a relaxation $\mathcal{H}_2^\#$ of $\mathcal{H}$ such that the hybrid system $\mathsf{DTR}(\mathcal{H}_2^\#, dtb_{\mathcal{H}_1^\#})$ has the same runs as $\mathcal{H}$. Generally, for a given relaxation $\mathcal{H}_2^\#$ of $\mathcal{H}$, it is not possible to find a relaxation $\mathcal{H}_1^\#$ of $\mathcal{H}$ such that the inferred dwell time bounds are precise enough, i.e., such that $\mathsf{DTR}(\mathcal{H}_2^\#, dtb_{\mathcal{H}_1^\#})$ has the same runs as $\mathcal{H}$.

A family of dwell time bounds incarnates a rather conservative approximation. This is because, intuitively, a bound at a location ℓ accounts for every visit of the location by a run of the hybrid system; i.e., it ignores the history of the run. As a tradeoff, one can choose a rather coarse relaxation for $\mathcal{H}_1^\#$ and obtain (practically optimal) dwell time bounds rather efficiently (this is confirmed by our practical experiments, where the cost for the inference of the dwell time bounds is negligible).

7 Case Studies

7.1 Case Study: Stability Verification

For a proof of concept, we have implemented the sequential composition of reachability analyses and automatic computation of dwell time bounds and used it to

Table 1. Case study on three previously unsolved instances of the stability verification problem: a water tank system with a parametrized number of tanks (three and four) and a temperature controller with a parametrized number of internal engines (two). Execution times are in seconds on a Pentium 2.7 GHz, with Linux Debian 2.1.18. The regions in the specification of the stability criterion are varied for the purpose of comparison of execution times; the stability condition becomes weaker with a larger region; for the purpose of specifying correctness, the larger regions are less interesting. In each case, $\mathcal{H}_2^{\#}$ is a relaxation of $\mathcal{H}$ obtained by eliminating all variables that are not used in the specification of the region. $\mathcal{H}_1^{\#}$ is the relaxation of $\mathcal{H}$ described in Section 5. Both, plain reachability and sequential composition of reachability analyses are realized with PHAVer [4], Version 0.38

system	region	$\mathcal{H}_2^{\#}$		$\mathsf{DTR}(\mathcal{H}_2^{\#}, dtb_{\mathcal{H}_1^{\#}})$	
		result	time	result	time
three water tanks	$x_3 \geq 5$	stable	5.03	stable	4.74
three water tanks	$x_3 \geq 6$	stable	897.82	stable	17.73
three water tanks	$x_3 \geq 7$	stable	13518.26	stable	34.53
three water tanks	$x_3 \geq 8$	–	timeout	stable	57.87
three water tanks	$x_3 \geq 9$	–	timeout	stable	83.13
three water tanks	$x_3 \geq 10$	–	timeout	stable	109.79
four water tanks	$x_4 \geq 5$	stable	18.66	stable	6.84
four water tanks	$x_4 \geq 6$	stable	5052.40	stable	38.38
four water tanks	$x_4 \geq 7$	–	timeout	stable	81.28
four water tanks	$x_4 \geq 8$	–	timeout	stable	160.30
four water tanks	$x_4 \geq 9$	–	timeout	stable	248.09
four water tanks	$x_4 \geq 10$	–	timeout	stable	343.15
two engines heater	$19 \leq x_r \leq 25$	???	14.44	stable	16.11
two engines heater	$20 \leq x_r \leq 25$	???	14.46	stable	17.01
two engines heater	$19 \leq x_r \leq 24$	???	25.04	stable	35.97
two engines heater	$20 \leq x_r \leq 24$	???	24.97	stable	37.39

conduct preliminary experiments with the abstraction-based verification method for region stability of hybrid systems [11,12,13]. This method seems an appealing first target because its preprocessing step doubles the number of continuous variables; i.e., variable elimination is the last resort for relatively small parameters in our scalable benchmark systems. As a consequence, a number of previously unsolved instances of the stability verification problem for classical hybrid system benchmarks were available as test cases. Our method proved to be effective on these tests.

A hybrid system is stable with respect to a given region φ if for every run, there exists a point of time such that from then on, the states on the run are always in the region φ (it may go in and out arbitrarily often before this point of time). The verification method transforms the hybrid system into another one where each continuous variable is duplicated. It is the hybrid system on that a reachability analysis is performed, e.g., by PHAVer [4]. There is an additional step on the output of the reachability analysis which does not matter here; the obstacle to scalability lies in the reachability analysis on the system with the doubled number of variables.

For our case study, we took two classical scalable benchmarks, the water tank system with a parametrized number of tanks and the temperature controller with a parametrized number of internal engines. The instances for three and four tanks and three internal engines were previously unsolved. Using sequential composition of reachability analyses with automatically inferred dwell time bounds for the abstraction of these hybrid systems by variable elimination, the verification method could solve these instances. We refer to Table 1 for the execution times (in sec, on a Pentium 2,7 GHz, with Linux Debian 2.1.18).

In order to obtain more benchmarks for comparing executions, we have varied the regions in the specification of the stability property. For sufficiently large regions, the abstraction by variable elimination was not coarse; i.e., the property could be checked. These regions are, however, not interesting as correctness specifications. The stability property for the water tank system expresses that the final tank (represented by x_3 resp. x_4) must stabilize with a minimum fill-up quantity. The last region is tight; i.e., the stability property does not hot hold for $x_3 \geq 11$ or $x_4 \geq 11$. For the two engines heater, the stability property expresses that the room temperature (represented by x_r) must stabilize within a "comfort zone". The last region is tight; i.e., the system does not stabilize for $x_r \geq 21$ or $x_r \leq 23$.

$\mathcal{H}_1^{\#}$ is the relaxation of $\mathcal{H}$ described in Section 5. The choice of the variable to be eliminated for the abstraction of $\mathcal{H}$ (yielding relaxation $\mathcal{H}_2^{\#}$) is determined by specification of the correctness property; i.e., we eliminate all variables that are not used for specifying the region (of 'stable states'). For the interesting regions, the abstraction $\mathcal{H}_2^{\#}$ is too coarse; the verification tool returns the answer

Table 2. Checking of a safety property: an automatic highway with arbiter example. $\mathcal{H}_1^{\#}$ is a relaxation of $\mathcal{H}$ obtained by ignoring the synchronization between the parallel components of the hybrid system $\mathcal{H}$. The relaxation $\mathcal{H}_2^{\#}$ is obtained by ignoring the values of the continuous variables of $\mathcal{H}$. Execution times are in seconds on a Pentium 2.7 GHz, with Linux Debian 2.1.18. Both, plain reachability and sequential composition of reachability analyses are realized with PHAVer [4], Version 0.38.

Number of cars	$\mathcal{H}$	$\mathsf{DTR}(\mathcal{H}_2^{\#}, dtb_{\mathcal{H}_1^{\#}})$
8	2.06	4.29
9	2.93	4.92
10	4.76	5.57
11	32.05	6.78
12	58.87	7.81
13	171.64	7.94
14	829.56	9.39
15	5904.12	10.65
16	timeout	12.48
17	timeout	13.20
18	timeout	14.83
19	timeout	15.47
20	timeout	16.93

"stability unknown", while the verification method applied to $\mathsf{DTR}(\mathcal{H}_2^{\#}, dtb_{\mathcal{H}_1^{\#}})$ terminates with the answer "system stable" (see also Table 1, where "???" stands for "stability unknown").

7.2 Case Study: Safety Verification

We take the model of a central arbiter for an automated highway from [8]. We will check the safety property that no cars collide.

In this example the model is represented as a linear hybrid automaton. $\mathcal{H}_1^{\#}$ is obtained by ignoring the synchronization between the parallel components of the hybrid system $\mathcal{H}$. The relaxation $\mathcal{H}_2^{\#}$ is obtained by ignoring the values of the continuous variables of $\mathcal{H}$. The detailed results are presented in Table 2. We can see that using sequential composition of reachability analyses we can get the results much faster then just using plain reachability analysis with PHAVer.

8 Related Work and Conclusion

Related Work. Dwell time is a natural concept that appears in different uses [2,5,7,9,10,14,16]. Jumping ahead and summarizing the detailed discussion to follow, it seems that our work is the first to investigate the possibility of inferring dwell time bounds via a (black-box) reachability analysis applied to an (in general coarse) auxiliary abstraction, and using them for improving the precision of the reachability analysis of another (in some sense, complementary) abstraction of the hybrid system by a refinement (the refinement being a transformation of hybrid systems).

In [2], the dimension of the model is reduced by projecting out continuous variables and replacing differential equations by differential inclusions and, in case the resulting abstraction is too coarse, adding (manually inferred) information about staying time. The idea of using differential inclusions may be interesting in our setting as well.

In [5], clock variables replace variables with non-linear dynamics, which is possible only if the value of the replaced 'non-linear' variable can be inferred from the value of the replacing clock variable.

In [9,16], the abstract discrete model based on *rectangular cells* is refined by adding (manually inferred) information about the time between the entry and the exit of a rectangular cell. In [14], a hybrid system is transformed into a timed system by replacing each continuous variable with a clock variable. The motivation for such an approach is the possibility to use a model checker for timed automata. It may be interesting to combine the idea of splitting a location according to rectangular cells with our approach, in order to obtain more expressive dwell time constraints.

In [8], a coarse *relaxation abstraction* (which can be done by eliminating continuous variables) is iteratively refined (in a counterexample-guided fashion) by adding back continuous variables. This refinement is orthogonal to, and can potentially be combined with dwell time refinement.

The notion of *average dwell time* in [7,10] refers to a run, in contrast with our notion of dwell time which refers to an individual location. A check of the validity of a given average dwell time (but not its automatic inference) is proposed in [10]). It is not clear how one would relate the two notions. Average dwell time is used to bound the frequency of discrete transitions, which is a condition in a local proof rule for asymptotic stability.

Conclusion and Future Work. In this paper, we have presented new methods both for inferring and for exploiting dwell time bounds. As a consequence, we have obtained an approach where one can incorporate the result of a first, auxiliary reachability analysis into a second reachability analysis in order to refine its abstraction and thus improve its precision. Our approach is generic in that it uses reachability analysis as a black-box method. We have presented initial experiments that indicate its practical potential for checking safety and stability. The experiments used one particular tool, but in principle, the approach can be piggy-packed on other reachability analysis tools (each with its own abstraction method, and with different scopes of applicability), and it will be interesting to explore those options. Also, an interesting perspective is to investigate the Russian doll technique, i.e., the nesting of a sequence of reachability analyses for deriving a sequence of successively stronger dwell time bounds.

References

1. Alur, R., Grosu, R., Hur, Y., Kumar, V., Lee, I.: Modular specification of hybrid systems in CHARON. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 6–19. Springer, Heidelberg (2000)
2. Asarin, E., Dang, T.: Abstraction by projection and application to multi-affine systems. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 32–47. Springer, Heidelberg (2004)
3. Dang, T.: *d/dt*, `http://www-verimag.imag.fr/~tdang/ddt.html`
4. Frehse, G.: PHAVer: Algorithmic verification of hybrid systems past HyTech. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 258–273. Springer, Heidelberg (2005)
5. Henzinger, T.A., Ho, P.-H.: Algorithmic analysis of nonlinear hybrid systems. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 225–238. Springer, Heidelberg (1995)
6. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: HyTech: A model checker for hybrid systems. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 460–463. Springer, Heidelberg (1997)
7. Hespanha, J.P., Morse, A.S.: Stability of switched systems with average dwell-time. In: Decision and Control (1999)
8. Jha, S.K., Krogh, B.H., Weimer, J.E., Clarke, E.M.: Reachability for linear hybrid automata using iterative relaxation abstraction. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 287–300. Springer, Heidelberg (2007)
9. Maler, O., Batt, G.: Approximating continuous systems by timed automata. In: Fisher, J. (ed.) FMSB 2008. LNCS (LNBI), vol. 5054, pp. 77–89. Springer, Heidelberg (2008)

10. Mitra, S., Liberzon, D., Lynch, N.A.: Verifying average dwell time of hybrid systems. ACM Trans. Embedded Comput. Syst. 8(1) (2008)
11. Podelski, A., Wagner, S.: Model checking of hybrid systems: From reachability towards stability. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 507–521. Springer, Heidelberg (2006)
12. Podelski, A., Wagner, S.: Region stability proofs for hybrid systems. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 320–335. Springer, Heidelberg (2007)
13. Podelski, A., Wagner, S.: A sound and complete proof rule for region stability of hybrid systems. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 750–753. Springer, Heidelberg (2007)
14. Puri, A., Varaiya, P.: Verification of hybrid systems using abstractions. In: Hybrid Systems II, pp. 359–369. Springer, Heidelberg (1995)
15. Silva, B.I., Richeson, K., Krogh, B.H., Chutinan, A.: Modeling and verification of hybrid dynamical system using CheckMate. In: ADPM (2000)
16. Stursberg, O., Kowalewski, S., Engell, S.: On the generation of timed discrete approximations for continuous systems. Mathematical and Computer Modeling of Dynamical Systems 6(1), 51–70 (2000)
17. Torrisi, F.D., Bemporad, A.: Hysdel — a tool for generating computational hybrid models for analysis and synthesis problems. IEEE Transactions on Control Systems Technology 12 (2004)