

Hierarchical Reasoning and Model Generation for the Verification of Parametric Hybrid Systems

Viorica Sofronie-Stokkermans

University of Koblenz-Landau and Max-Planck-Institut für Informatik, Saarbrücken
sofronie@uni-koblenz.de

Abstract. In this paper we study possibilities of using hierarchical reasoning, quantifier elimination and model generation for the verification of parametric hybrid systems, where the parameters can be constants or functions. Our goal is to automatically provide guarantees that such systems satisfy certain safety or invariance conditions. We first analyze the possibility of automatically generating such guarantees in the form of constraints on parameters, then show that we can also synthesise so-called criticality functions, typically used for proving stability and/or safety of hybrid systems. We illustrate our methods on several examples.

1 Introduction

In this paper we study possibilities of using hierarchical reasoning, quantifier elimination and model generation for the analysis and verification of increasingly more general classes of parametric hybrid systems (where the parameters can be either constants or functions, possibly modeling parametric “updates” of the system and/or environment). Our goal is to give methods which allow to automatically provide guarantees that such systems satisfy certain safety or invariance conditions. Our main contributions can be described as follows:

- We refine the methods for generating constraints on parameters which we gave in [15,3,4] and apply them to increasingly more complex hybrid systems (including non-linear hybrid systems or families of hybrid systems).
- We study possibilities of automatically generating criticality functions (used to characterize regions where safety properties can be guaranteed).

There exist approaches to the verification of parametric hybrid automata (e.g. by Henzinger et al. [1], Frehse [6], Wang [17], Tiwari et al. [7] and Cimatti et al. [2], Platzer et al. [10,11]) possibly used for synthesis [16]. However, in most cases only situations in which the parameters are constants were considered. In [15] and [3,4] we made first steps towards the verification of hybrid systems with a more complex form of parametricity, but with a focus on linear hybrid automata. We here extend these results to parametric hybrid automata, and also explore possibilities of generating criticality functions in order to give guarantees for safety in such systems. The main advantage of our method for generating

criticality functions compared with existing approaches based on Lyapunov function computation (cf. e.g. [12]) is that we restrict to very simple properties of the continuous variables (such as monotonicity properties) and thus avoid the complexity introduced by the complicated dynamics.

Structure. Section 2 introduces the main definitions and results on local theory extensions and hybrid automata used in the paper. In Section 3 we present a general result on generating constraints on parameters under which certain formulae are invariant (which we apply for the verification of parametric linear hybrid automata, then extend to parametric hybrid automata and systems thereof). In Section 4 we present a method for generating criticality functions.

2 Preliminaries

We present the main definitions and results on local theory extensions and hybrid automata used in the paper. For details on local theory extensions we refer to [13,8,9]; for details on hybrid automata we refer for instance to [1].

Local Theory Extensions. Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (S, \Sigma_0, \text{Pred})$, where S is a set of sorts, Σ_0 a set of function symbols, and Pred a set of predicate symbols. We consider extensions $\mathcal{T}_1$ of $\mathcal{T}_0$ with new function symbols in a set Σ_1 whose properties are axiomatized with a set $\mathcal{K}$ of clauses¹ whose free variables are considered to be universally quantified (Notation: $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$, where $\mathcal{T}_1$ has signature $\Pi = (S, \Sigma_0 \cup \Sigma_1, \text{Pred})$ - also written $\Pi = \Pi_0 \cup \Sigma_1$). An extension $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is *local* [13] if it satisfies condition (Loc):²

(Loc) For every finite set G of ground clauses $\mathcal{T}_1 \cup G \models \perp$ iff $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \perp$ where $\mathcal{K}[G]$ is the set of instances of $\mathcal{K}$ where all terms starting with an extension function are in the set $\text{st}(\mathcal{K}, G)$ of all ground terms occurring in $\mathcal{K}$ or G .

A similar *extended locality condition* (ELoc) [9] can be defined for theory extensions in which $\mathcal{K}$ consists of augmented clauses ((ELoc) states that $\mathcal{T}_1 \cup \Gamma \models \perp$ iff $\mathcal{T}_0 \cup \mathcal{K}[\Gamma] \cup \Gamma \models \perp$ holds for all Σ_0 -*extended ground clauses* $\Gamma = \Gamma_0 \cup G$, where Γ_0 is a Π_0^c -sentence and G is a finite set of ground Π^c -clauses). In [8] we introduced Ψ -(extended) locality conditions (Loc ^{Ψ}) and (ELoc ^{Ψ}), where Ψ is a closure operator on ground terms, in which the instances of $\mathcal{K}$ to be considered without loss of completeness are $\mathcal{K}[\Psi_{\mathcal{K}}(G)]$ instead of $\mathcal{K}[G]$, where $\Psi_{\mathcal{K}}(G) = \Psi(\text{st}(\mathcal{K}, G))$.

Hierarchical reasoning. Consider a Ψ -local theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$. Condition (Loc ^{Ψ}) requires that, for every set G of ground Π^c clauses, $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp$ iff $\mathcal{T}_0 \cup \mathcal{K}[\Psi_{\mathcal{K}}(G)] \cup G \models \perp$. In all clauses in $\mathcal{K}[\Psi_{\mathcal{K}}(G)] \cup G$ the function symbols in Σ_1 have as arguments only ground terms, so $\mathcal{K}[\Psi_{\mathcal{K}}(G)] \cup G$ can be flattened

¹ We can also consider sets $\mathcal{K}$ of *augmented clauses*, i.e. of axioms of the form $(\Phi(x_1, \dots, x_n) \vee C(x_1, \dots, x_n))$, where $\Phi(x_1, \dots, x_n)$ is an *arbitrary first-order formula* in the base signature Π_0 and $C(x_1, \dots, x_n)$ is a *clause* containing Σ_1 -functions.

² In what follows, when we refer to *sets* G of ground clauses we assume that they are in the signature $\Pi^c = (S, \Sigma \cup \mathcal{C}, \text{Pred})$, where $\mathcal{C}$ is a set of new constants.

and purified³. The set of clauses thus obtained has the form $\mathcal{K}_0 \cup G_0 \cup \text{Def}$, where Def consists of ground unit clauses of the form $f(c_1, \dots, c_n) = c$, where $f \in \Sigma_1$, $c, c_1, \dots, c_n$ are constants, and $\mathcal{K}_0$ and G_0 do not contain Σ_1 -function symbols.

Theorem 1 ([13,8]). *Let $\mathcal{K}$ be a set of clauses. Assume that $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is a Ψ -local theory extension. For any set G of ground clauses, let $\mathcal{K}_0 \cup G_0 \cup \text{Def}$ be obtained from $\mathcal{K}[\Psi_{\mathcal{K}}(G)] \cup G$ by flattening and purification, as explained above. Then the following are equivalent to $\mathcal{T}_1 \cup G \models \perp$:*

(1) $\mathcal{T}_0 \cup \mathcal{K}[\Psi_{\mathcal{K}}(G)] \cup G \models \perp$.

(2) $\mathcal{T}_0 \cup \mathcal{K}_0 \cup G_0 \cup \text{Con}_0 \models \perp$, where $\text{Con}_0 = \{ \bigwedge_{i=1}^n c_i \approx d_i \rightarrow c \approx d \mid \begin{smallmatrix} f(c_1, \dots, c_n) \approx c \in \text{Def} \\ f(d_1, \dots, d_n) \approx d \in \text{Def} \end{smallmatrix} \}$.

As a consequence we obtain the following decidability transfer result.

Theorem 2 ([13,8]). *If the theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ satisfies condition (Loc^{Ψ}) then satisfiability of ground clauses G w.r.t. $\mathcal{T}_1$ is decidable provided $\mathcal{K}[\Psi_{\mathcal{K}}(G)]$ is finite and $\mathcal{K}_0 \cup G_0 \cup \text{Con}_0$ belongs to a decidable fragment of $\mathcal{T}_0$.*

Similar results hold for extended Ψ -local extensions (then $\mathcal{K}, \mathcal{K}_0$ and G_0 may contain arbitrary Π_0 -sentences and G is a set of Σ_0 -extended ground clauses).

The (Ψ) -locality of an extension can be recognized by proving embeddability of partial into total models assuming that the extension clauses are flat and linear [13,8,9]. If we can guarantee that the support of the total model which we obtain is the same as the support of the partial model we start with, then condition (ELoc^{Ψ}) is guaranteed. The locality proof also explains how to construct models of satisfiable ground (extended) clauses starting from models of their instances.

We present some theory extensions which are Ψ -local. In what follows we will denote by $\mathbb{R}$ both the set of real numbers and the theory of real numbers (more precisely the theory of real closed fields) and by $LI(\mathbb{R})$ linear arithmetic over $\mathbb{R}$.

Monotonicity, Boundedness for Monotone Functions. Any extension of a theory for which $\leq$ is a partial order with functions satisfying⁴ $\text{Mon}^{\sigma}(f)$ and/or $\text{Bound}^t(f)$ is local (cf. e.g. [8]). The extensions satisfy condition (ELoc) if e.g. in $\mathcal{T}_0$ all finite and empty infima (or suprema) exist.

$$\text{Mon}^{\sigma}(f) \quad \bigwedge_{i \in I} x_i \leq_i^{\sigma_i} y_i \wedge \bigwedge_{i \notin I} x_i \approx y_i \rightarrow f(x_1, \dots, x_n) \leq f(y_1, \dots, y_n)$$

$$\text{Bound}^{s,t}(f) \quad \forall x_1, \dots, x_n (s(x_1, \dots, x_n) \leq f(x_1, \dots, x_n) \leq t(x_1, \dots, x_n))$$

where s, t are new functions or $s(x_1, \dots, x_n)$ and $t(x_1, \dots, x_n)$ are Π_0 -terms with variables among $x_1, \dots, x_n$ and: (i) s, t have the same monotonicity as f in any model and (ii) $\forall x_1, \dots, x_n \ s(x_1, \dots, x_n) \leq t(x_1, \dots, x_n)$.

Convexity/Concavity [14]. Let $\mathcal{T}_0$ be $\mathbb{R}$, the theory of real numbers (or the many-sorted combination of the theories of real numbers (sort *real*) and integers

³ The function symbols in Σ_1 are separated from the other symbols by introducing, in a bottom-up manner, new constants c_t for subterms $t = f(c_1, \dots, c_n)$ where $f \in \Sigma_1$ and c_i are constants, together with definitions $c_t = f(c_1, \dots, c_n)$ (C is a set of constants containing all constants introduced by flattening and purification).

⁴ For $i \in I$, $\sigma_i \in \{-, +\}$, and for $i \notin I$, $\sigma_i = 0$; $\leq^+ = \leq, \leq^- = \geq$.

(sort int)). Let f be a unary function symbol with arity $\text{real} \rightarrow \text{real}$ (or resp. $\text{int} \rightarrow \text{real}$). Then $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \text{Conv}_f$ and $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \text{Conc}_f$ satisfy condition (ELoc) where:

$$\text{Conv}(f) \quad \forall x, y, z \left(x < z \leq y \rightarrow \frac{f(z) - f(x)}{z - x} \leq \frac{f(y) - f(x)}{y - x} \right).$$

and the concavity condition Conc_f is the convexity condition for $-f$.

Linear Combinations of Functions. Let $f_1, \dots, f_n, f, g$ be unary function symbols. The extension $\mathbb{R} \subseteq \mathbb{R} \cup \text{BS}$ satisfies condition (ELoc), where BS contains

conjunctions of axioms of type $\forall t (a \leq \sum_{i=1}^n a_i f_i(t) \leq b), \forall t (g(t) \leq \sum_{i=1}^n a_i f_i(t) \leq f(t))$,

or $\forall t, t' (t < t' \rightarrow a \leq \sum_{i=1}^n a_i \frac{f_i(t') - f_i(t)}{t' - t} \leq b)$, where $a, b \in \mathbb{R}$ and (i) g is convex

and f concave and (ii) either g and f satisfy the condition $\forall t (g(t) \leq f(t))$ or correspond to Π_0 -terms and $\models_{\tau_0} \forall t g(t) \leq f(t)$. Using arguments similar to those used in [14] it can be proved that if we additionally require the functions to be continuous locality is still preserved.

If I is an interval of the form $(-\infty, a], [a, b]$ or $[a, \infty)$ then we can define versions of monotonicity/boundedness, convexity/concavity and boundedness axioms for linear combinations of functions and of their slopes relative to the interval I (then conditions (i) and (ii) for f and g are relative to the interval I).

Hybrid Automata. A hybrid automaton (abbreviated HA in what follows) [1] is a tuple $S = (X, Q, \text{flow}, \text{Inv}, \text{Init}, E, \text{guard}, \text{jump})$ consisting of:

- (1) A finite set $X = \{x_1, \dots, x_n\}$ of real valued variables and a finite set Q of control modes, that together define the state space of the system;
- (2) A family $\{\text{flow}_q \mid q \in Q\}$ of predicates over the variables in $X \cup \dot{X}$ (where $\dot{X} = \{\dot{x}_1, \dots, \dot{x}_n\}$, where $\dot{x}_i$ is the derivative of x_i) specifying the continuous dynamics in each control mode; a family $\{\text{Inv}_q \mid q \in Q\}$ of predicates over the variables in X defining the invariant conditions for each control mode; and a family $\{\text{Init}_q \mid q \in Q\}$ of predicates over the variables in X , defining the initial states for each control mode.
- (3) A finite multiset E with elements in $Q \times Q$ (the control switches). Every $(q, q') \in E$ is a directed edge between q (source mode) and q' (target mode); a family of guards $\{\text{guard}_e \mid e \in E\}$ (predicates over X); and a family of jump conditions $\{\text{jump}_e \mid e \in E\}$ (predicates over $X \cup X'$, where $X' = \{x'_1, \dots, x'_n\}$ is a copy of X consisting of “primed” variables).

A *state* of S is a pair (q, a) consisting of a control mode $q \in Q$ and a vector $a = (a_1, \dots, a_n)$ that represents a value $a_i \in \mathbb{R}$ for each variable $x_i \in X$. A state (q, a) is *admissible* if Inv_q is true when each x_i is replaced by a_i . There are two types of *state change*: (i) A *jump* is an instantaneous transition that changes the control location and the values of data variables according to the jump conditions; (ii) In a *flow*, the state can change due to the evolution in a given control mode over an interval of time: the values of the data variables change continuously according to the flow rules of the current control location; all intermediate states are admissible. A *run* of S is a finite sequence $s_0 s_1 \dots s_k$

of admissible states s_j such that (i) the first state s_0 is an initial state of S (the values of the variables satisfy Init_q for some $q \in Q$), (ii) each pair (s_j, s_{j+1}) is either a jump of S or the endpoints of a flow of S .

Notation. In what follows we use the following notation. If $x_1, \dots, x_n \in X$ we denote the sequence $x_1, \dots, x_n$ with $\overline{x}$, the sequence $\dot{x}_1, \dots, \dot{x}_n$ with $\overline{\dot{x}}$, and the sequence of values $x_1(t), \dots, x_n(t)$ of these variables at a time t with $\overline{x}(t)$.

Verification Problems. We are interested in invariant checking or bounded model checking *under given constraints on parameters*, and in *deriving constraints between parameters* which guarantee that a certain safety property is an invariant of the system or holds for paths of bounded length. We studied this problem in [15] and [3,4] but only for very simple types of linear hybrid systems. In this paper we continue this direction of research in two different ways:

Deductive Verification and Synthesis. We identify possibilities for generating conditions on parameters under which a certain formula (i) is an invariant or (ii) holds for paths of bounded length. For simplicity we will here focus on (i); (ii) can be solved in a similar way.

Criticality Functions. We give methods for generating *criticality functions* which ensure that a hybrid automaton satisfies certain safety conditions.

Example 1 (Running example). *We consider a temperature controller, modeled as a hybrid automaton with two modes: a heating mode (in which the environment of the object is heated) and a normal mode (heating is switched off). The control variable is x (the temperature of the object). We assume that the system has two parameters (which can be functional or not).*

- *The temperature of the heated environment (due to the heater): a constant h or (if it changes over time) a unary function h (input and output sort: real).*
- *Perturbation of the temperature of the environment due to external causes (e.g. external temperature), modeled using a constant f or (if it changes in time) a unary function f (input and output sort: real).*

Invariants and flows in the two modes are described below ($k > 0$ is a constant which depends only on the surface of the object which is being heated):

Mode 1 (Heating): Invariant: $T_a \leq x(t) \leq T_b$; Flow: $\frac{dx}{dt} = -k(x - (h + f))$,

Mode 2 (Normal): Invariant: $T_c \leq x(t) \leq T_a$; Flow: $\frac{dx}{dt} = -k(x - f)$.

Control Switches. *We have two control switches:*

e_{12} : *switch from Mode 1 to Mode 2 (if the temperature of the object becomes too high heating is switched off): guard $_{e_{12}}$: $x \geq T_b$; jump $_{e_{12}}$: $(x' = x)$.*

e_{21} : *switch from Mode 2 to Mode 1 (if the temperature of the object becomes too low heating is switched on): guard $_{e_{21}}$: $x \leq T_c$; jump $_{e_{21}}$: $(x' = x)$.*

Let $\text{Safe} = T_m \leq x(t) \leq T_M$ (a safety condition for the heater). Our goals are:

- (1) *check that Safe is an invariant (or that it holds on all runs of bounded length),*
- (2) *generate constraints which guarantee that Safe is an invariant,*
- (3) *generate a criticality function which would provide a guarantee for safety.*

In Sect. 3 we give methods for solving (1) and (2) for increasingly larger classes of parametric hybrid automata. We then solve (3) by using methods for model generation in local theory extensions (Sect. 4).

3 Deductive Verification and Synthesis

We present a general result which will be used for the verification of parametric linear hybrid automata (Sect. 3.1) and, in extended form also for more general classes of parametric hybrid automata, possibly interconnected (Sect. 3.2–3.3).

Let $\mathcal{T}_0$ be a Π_0 -theory and let Σ_P be a set of parameters (function and constant symbols). Let Σ be a signature such that $\Sigma \cap (\Sigma_0 \cup \Sigma_P) = \emptyset$, containing functions which can change their values during flows and jumps. Let $\Sigma' = \{f' \mid f \in \Sigma\}$ be a disjoint copy of Σ representing the new functions after the updates. Let $\mathcal{KF}$ be a set of clauses over $\Pi_0 \cup \Sigma \cup \Sigma_P$ expressing the properties of the functions in $\Sigma \cup \Sigma_P$, and let $\text{Update}(\bar{x}, \bar{x}', \bar{f}, \bar{f}')$ (containing for every x also x' and for every $f \in \Sigma$ also $f' \in \Sigma'$) be a set of clauses expressing the way the variables and the functions in Σ are changed during updates.

Let Ψ be a set of clauses in the signature $\Pi_0 \cup \Sigma_P \cup \Sigma$ which can contain (implicitly universally quantified) variables as arguments of the functions in $\Sigma_P \cup \Sigma$.

Example 2. In Example 1, $\mathcal{T}_0$ is the theory of real numbers. The parameters in Σ_P are $T_m, T_M, T_a, T_b, T_c, T_d$ (constants) and h, f (functions). The set of functions which change their values is $\Sigma = \{x\}$; $\Sigma' = \{x'\}$. An example of set of axioms describing the properties of parameters is: $\mathcal{KF} = \{T_m \leq T_M, T_a \leq T_c \leq T_b \leq T_d, k \geq 0, \forall t(0 \leq h(t))\}$. We have two types of updates: updates due to flows (in which x is the value of the variable x at the beginning of the flow and x' the value at the end of the flow) and updates due to jumps (described by the jump conditions). More complex examples are given in Sect. 3.1–3.3.

Theorem 3 ([15]). Let Γ_0 be a set of constraints on the parameters. Assume that $\Gamma_0, \Psi, \mathcal{KF}$ and Update are sets of clauses which define the following chain of theory extensions satisfying condition ELoc : $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \Gamma_0 \subseteq \mathcal{T}_0 \cup \Gamma_0 \cup \Psi \subseteq \mathcal{T}_0 \cup \Gamma_0 \cup \Psi \cup \mathcal{KF} \subseteq \mathcal{T}_0 \cup \Gamma_0 \cup \Psi \cup \mathcal{KF} \cup \text{Update}$. Assume that either (i) ground satisfiability in $\mathcal{T}_0$ is decidable and all variables in these theory extensions occur below extension functions, or (ii) satisfiability of $\exists \forall$ formulae is decidable in $\mathcal{T}_0$. Then we can decide whether Ψ is an invariant (using repeatedly Theorem 1).

If $\mathcal{T}_0$ has quantifier elimination then we can construct a constraint on the parameters Γ which guarantees that Ψ is an invariant:

Theorem 4. Assume that the theory $\mathcal{T}_0$ has quantifier elimination. We can construct a universally quantified formula $\forall \bar{x} \Gamma(\bar{x})$ (containing also some of the parameters) such that for every structure $\mathcal{A}$ with signature $\Pi_0 \cup \Sigma \cup \Sigma' \cup \Sigma_P$ which is a model of $\mathcal{T}_0$, if $\mathcal{A} \models \forall \bar{x} \Gamma(\bar{x})$ then Ψ is an invariant w.r.t. interpretation $\mathcal{A}$.

Proof: We use instantiation, renaming (as in Thm. 1) and quantifier elimination in $\mathcal{T}_0$ to construct a formula Γ such that if Ψ is not invariant w.r.t. $\mathcal{A}$, i.e.

$\mathcal{A} \models \mathcal{T}_0 \cup \mathcal{KF} \cup \exists \bar{x}, \bar{x}' (\Psi(\bar{x}, \bar{f}) \wedge \text{Update}(\bar{x}, \bar{x}', \bar{f}, \bar{f}') \wedge \neg \Psi(\bar{x}', \bar{f}'))$ then $\mathcal{A}$ is a model of $\exists x \neg \Gamma(x)$, so, if $\mathcal{A} \models \forall \bar{y} \Gamma(\bar{y})$ then Ψ is an invariant w.r.t. $\mathcal{A}$. $\square$

In what follows we present applications of this result to increasingly more complex classes of hybrid automata.

3.1 Parametric Linear Hybrid Automata

A hybrid automaton S is a linear hybrid automaton (LHA) if it satisfies the following two requirements [1]:

1. *Linearity*: For every control mode $q \in Q$, the flow condition flow_q , the invariant condition Inv_q , and the initial condition Init_q are convex linear predicates⁵. For every control switch $e = (q, q') \in E$, the jump condition jump_e and the guard guard_e are convex linear predicates. In addition, as in [3,4], we assume that the flow conditions flow_q are conjunctions of *non-strict* linear inequalities.
2. *Flow independence*: For every control mode $q \in Q$, the flow condition flow_q is a predicate over the variables in $\dot{X}$ only (and does not contain any variables from X). This requirement ensures that the possible flows are independent from the values of the variables, and only depend on the control mode.

We also consider *parametric* linear hybrid automata (PLHA), defined as linear hybrid automata for which a set $\Sigma_P = P_c \cup P_f$ of parameters is specified (consisting of parametric constants P_c and parametric functions P_f) with the difference that for every control mode $q \in Q$ and every mode switch e :

- (1) the linear constraints in the invariant conditions Inv_q , initial conditions Init_q , and guard conditions guard_e are of the form: $g \leq \sum_{i=1}^n a_i x_i \leq f$,
- (2) the inequalities in the flow conditions flow_q are of the form: $\sum_{i=1}^n b_i \dot{x}_i \leq b$,
- (3) the linear constraints in jump_e are of the form $\sum_{i=1}^n b_i x_i + c_i x'_i \leq d$,

(possibly relative to an interval I) where the coefficients a_i, b_i, c_i and the bounds b, d are either numerical constants or parametric constants in P_c ; and g and f are (i) constants or parametric constants in P_c , or (ii) parameteric functions in P_f satisfying the convexity (for g) resp. concavity condition (for f), or concrete functions with these convexity/concavity properties such that $\forall t (g(t) \leq f(t))$. The flow independence conditions hold as in the case of linear hybrid automata.

Note: In the definition of PLHA we allow a general form of parametricity, in which the bounds in state invariants, guards and jump conditions can be expressed using functions with certain properties. Such parametric descriptions of bounds are useful for instance in situations in which we want to verify systems which have non-linear behavior and use a parametric approximation for them.

Example 3. Consider the hybrid automaton S presented in Example 1. If in the heating mode the invariant is $T_a \leq x(t) \leq T_b$ and the flow is $\frac{dx}{dt} = -k(x - (h + f))$, where f, h are constants, then we can approximate the flow by the linear flow:

$$-k(T_b - (h + f)) \leq \dot{x} \leq -k(T_a - (h + f)). \quad (1)$$

⁵ An atomic linear predicate is a strict or non-strict linear inequality. A convex linear predicate is a finite conjunction of linear inequalities.

We can obtain similar bounds for $\dot{x}$ also for mode 2. Thus, we can approximate S using a linear hybrid automaton S' . If we can guarantee safety in S' , then safety is preserved for all possible runs which satisfy the flow conditions of S' , in particular also for all runs of S , so S is safe.

We give methods of deciding whether a formula Ψ is an invariant and of deriving conditions that guarantee that a PLHA S has a given safety property. To use Thm. 4, we analyze the possible updates in PLHA by jumps and flows.

Jumps. A jump update can be expressed by the linear inequality

$$\text{Jump}_e(\bar{x}, \bar{x}') = \text{guard}_e(\bar{x}) \wedge \text{jump}_e(\bar{x}, \bar{x}').$$

Flows. Assume that $\text{flow}_q(t) = \bigwedge_{j=1}^{n_q} (\sum_{i=1}^n c_{ij}^q \dot{x}_i(t) \leq_j c_j^q)$. We alternatively axiomatize flows in mode q in the time interval $[t_0, t_1]$ (where $0 \leq t_0 \leq t_1$) as follows:

$$\text{Flow}_q(t_0, t_1) = \forall t(t_0 \leq t \leq t_1 \rightarrow \text{Inv}_q(\bar{x}(t))) \wedge \forall t, t'(t_0 \leq t < t' \leq t_1 \rightarrow \underline{\text{flow}}_q(t, t'))$$

$$\text{where: } \underline{\text{flow}}_q(t, t') = \bigwedge_{j=1}^{n_q} \left(\sum_{i=1}^n c_{ij}^q (x_i(t') - x_i(t)) \leq_j c_j^q (t' - t) \right).$$

In [3,4] we showed that for LHA no precision is lost with this axiomatization. We can, in fact simplify the axiomatization of flows further by suitably instantiating the universal quantifiers in Flow_q and obtain:

Theorem 5 ([4]). *The following are equivalent for any LHA:*

- (1) Ψ is an invariant of the automaton.
- (2) For every $q \in Q$ and $e = (q, q') \in E$, the formulae $F_{\text{init}}(q)$, $F'_{\text{flow}}(q)$ and $F_{\text{jump}}(e)$ are unsatisfiable, where:

$$\begin{array}{ll} F_{\text{init}}(q) & \text{Init}_q(\bar{x}(t_0)) \wedge \neg \Psi(\bar{x}(t_0)) \\ F'_{\text{flow}}(q) & \text{Inv}_q(\bar{x}(t_0)) \wedge \Psi(\bar{x}(t_0)) \wedge \underline{\text{flow}}_q(t_0, t) \wedge \text{Inv}_q(\bar{x}(t)) \wedge \neg \Psi(\bar{x}(t)) \wedge t \geq t_0 \\ F_{\text{jump}}(e) & \Psi(\bar{x}(t)) \wedge \text{Jump}_e(\bar{x}(t), \bar{x}'(0)) \wedge \text{Inv}_{q'}(\bar{x}'(0)) \wedge \neg \Psi(\bar{x}'(0)). \end{array}$$

Let S be a parametric LHA with parameters $\Sigma_P = P_c \cup P_f$. Assume that the properties of the parameters are expressed as $\Gamma_0 \wedge \Gamma_f$, where Γ_0 is a conjunction of linear inequalities representing the relationships between parameters in P_c and Γ_f is a set of (universally quantified) clauses expressing the properties of the functional parameters (in P_f) – containing the convexity/concavity conditions for the bounding functional parameters. Let Ψ be a property expressed as convex linear predicate over X , possibly containing parameters (constants as coefficients; either constants or functions as bounds in the linear inequalities). In [4] we showed that checking whether Ψ is an invariant is decidable in PTIME if bounds are not functional and in EXPTIME otherwise.⁶

Since the theory of reals allows quantifier elimination, the following result is a direct consequence of Thm. 4.

⁶ The proof in [4] is direct; this result can also be proved using the fact that extensions with axioms expressing boundedness conditions on linear combinations of functions and of their slopes define local theories and therefore (i) updates by flows as well as updates by jumps define local theory extensions for any extension of the reals; (ii) any convex predicate Ψ defines a local extension of the theory of reals.

Theorem 6. Let S be a PLHA and Ψ be a property expressed as a convex linear predicate over X , possibly containing parameters. We can effectively derive a set Γ of (universally quantified) constraints on the parameters such that whenever Γ holds in an interpretation $\mathcal{A}$, Ψ is an invariant w.r.t. $\mathcal{A}$.

This method for constraint synthesis can in particular be used for:

1. *Invariant generation:* Let S be a fixed (non-parametric) LHA. We consider invariant “templates” Ψ expressed by linear inequalities with parametric bounds and coefficients and determine constraints on these parameters which ensure that Ψ is an invariant. By finding values of the parameters satisfying these constraints we can generate concrete invariants.

2. *Generation of control conditions:* Assume that Ψ is fixed (non-parametric), but that the mode invariants, the flow conditions, the guards, and the jumps are represented parametrically (as conjunctions of a bounded number of linear inequalities). We can determine constraints on the parameters which ensure that Ψ is an invariant. By finding values of the parameters satisfying these constraints we can determine control conditions which guarantee that Ψ is invariant.

Example 4. Consider a variant of the HA in Example 1 in which T_a, T_b are functional parameters, Inv_1 is $\forall t(T_a(t) \leq x(t) \leq T_b(t))$, and the flow in mode 1 is described by: $-k(x_b - g) \leq \dot{x} \leq -k(x_a - g)$ (for simplicity we abbreviated $h + f$ by g). Let $\Psi = (T_m \leq x(t) \leq T_M)$. Assume that the properties of the parameters are axiomatized by $\mathcal{KF}: \{\forall t(x_a \leq T_a(t)), \forall t(T_b(t) \leq x_b), x_a < x_b, T_m < T_M\}$. We derive a condition Γ which guarantees that Ψ is preserved under flows in mode 1 using Thm. 4 as follows. Consider the formula:

$$\begin{aligned} \exists t_0, t_1 \quad & (T_m \leq x(t_0) \leq T_M) \wedge (T_a(t_0) \leq x(t_0) \leq T_b(t_0)) \wedge \\ & (x(t_1) \leq x(t_0) - k(t_1 - t_0)(x_a - g) \wedge x(t_0) - k(t_1 - t_0)(x_b - g) \leq x(t_1)) \\ & (T_a(t_1) \leq x(t_1) \leq T_b(t_1)) \wedge (x(t_1) < T_m \vee T_M < x(t_1)). \end{aligned}$$

After purification we obtain:

$$\begin{aligned} \exists t_0, t_1, T_{a0}, T_{b0}, x_0, x_1 \quad & (T_m \leq x_0 \leq T_M) \wedge (T_{a0} \leq x_0 \leq T_{b0}) \wedge \\ & (x_1 \leq x_0 - k(t_1 - t_0)(x_a - g) \wedge x_0 - k(t_1 - t_0)(x_b - g) \leq x_1) \wedge \\ & (T_{a1} \leq x_1 \leq T_{b1}) \wedge (x_1 < T_m \vee T_M < x_1). \end{aligned}$$

After eliminating the quantifiers $\exists x_0, x_1$ (assuming $\mathcal{KF}$ holds and $T_m \leq T_M$) and then replacing back T_{a0}, T_{a1} and T_{b0}, T_{b1} we obtain:

$$\begin{aligned} \exists t_0, t_1 \quad & (T_a(t_0) \leq T_M \wedge T_m \leq T_b(t_0)) \wedge \\ & T_m \leq T_b(t_1) + k(t_1 - t_0)(x_b - g) \wedge T_a(t_0) \leq T_b(t_1) + k(t_1 - t_0)(x_b - g) \wedge \\ & T_a(t_1) + k(t_1 - t_0)(x_a - g) \leq T_M \wedge T_a(t_1) + k(t_1 - t_0)(x_a - g) \leq T_b(t_0) \wedge \\ & [((k(t_1 - t_0)(x_b - g) > 0) \wedge T_a(t_1) < T_m \wedge T_a(t_0) < T_m + k(t_1 - t_0)(x_b - g)) \vee \\ & ((k(t_1 - t_0)(x_a - g) < 0) \wedge T_b(t_1) > T_M \wedge T_M + k(t_1 - t_0)(x_a - g) < T_b(t_0)))]). \end{aligned}$$

The condition Γ_1 which ensures that Ψ is an invariant under flows in mode 1 is:

$$\begin{aligned} \forall t_0, t_1 \quad & ((T_a(t_0) \leq T_M \wedge T_m \leq T_b(t_0)) \wedge \\ & T_m \leq T_b(t_1) + k(t_1 - t_0)(x_b - g) \wedge T_a(t_0) \leq T_b(t_1) + k(t_1 - t_0)(x_b - g) \wedge \\ & T_a(t_1) + k(t_1 - t_0)(x_a - g) \leq T_M \wedge T_a(t_1) + k(t_1 - t_0)(x_a - g) \leq T_b(t_0) \\ \rightarrow & [((k(t_1 - t_0)(x_b - g) \leq 0) \vee T_a(t_1) \geq T_m \vee T_a(t_0) \geq T_m + k(t_1 - t_0)(x_b - g)) \wedge \\ & ((k(t_1 - t_0)(x_a - g) \geq 0) \vee T_b(t_1) \leq T_M \vee T_M + k(t_1 - t_0)(x_a - g) \geq T_b(t_0)))]). \end{aligned}$$

Invariant generation. Assume that the control of the LHA (i.e. the functions T_a, T_b and their bounds x_a, x_b) is fixed, e.g. $k = 1$, T_a is the constant function $x_a = 15$ and T_b is the constant function $x_b = 20$, $h = 25$ and $f = 10$. We can use the constraint in Γ_1 to determine for which values of the constants T_m and T_M Ψ is an invariant under flows in mode 1 (we can check that it is, e.g., for $T_m = 15$ and $T_M = 20$). The generation of control conditions for guaranteeing that a (non-parametric) Ψ is invariant is similar.

3.2 Parametric Hybrid Automata

We now extend the methods developed above for parametric LHA to more general HA. For the sake of simplicity, we only consider (parametric) hybrid automata S with one continuous variable x . (The case when we have several variables is similar, but the presentation is more complicated.) Let $\Sigma = \{x\}$ and $\Sigma' = \{x'\}$. Assume that mode invariants, initial states, guards and jump conditions are expressed as sets of clauses in an extension of the theory of real numbers with additional functions in a set $\Sigma_1 = \Sigma \cup \Sigma_P$, where Σ_P is a set of parameter names (both functions and constants). We study the problem of deriving constraints on parameters which guarantee that a certain formula is an invariant. We therefore analyze the possible updates by jumps and flows.

Jumps. Assume that the guards and the jump conditions are given by formulae in a certain extension of the theory of real numbers. A jump update can be expressed by the formula: $\text{Jump}_e(x, x') = \text{guard}_e(x) \wedge \text{jump}_e(x, x')$.

Flows. In the case of parametric hybrid automata, the flows are described by differential equations. As we restrict to the case of one continuous variable x , we assume that in mode q the flow is described by: $\frac{dx}{dt}(t) = f_q(x(t))$. Thus:

$$\text{Flow}_q(t_0, t_1) = \forall t(t_0 \leq t \leq t_1 \rightarrow \text{Inv}_q(x(t))) \wedge \forall t(t_0 \leq t \leq t_1 \rightarrow \frac{dx}{dt}(t) = f_q(x(t))).$$

Let Ψ be a set of clauses in the signature $\Pi_0 \cup \Sigma_P \cup \Sigma$ which can contain (implicitly universally quantified) variables as arguments of the functions in $\Sigma_P \cup \Sigma$.

- Theorem 7.** (1) *For every jump e we can construct a universally quantified formula $\forall \bar{x} \Gamma_e(\bar{x})$ (containing also some of the parameters) such that for every structure $\mathcal{A}$ with signature $\Pi_0 \cup \Sigma \cup \Sigma' \cup \Sigma_P$ if $\mathcal{A}$ is a model of $\mathcal{T}_0$ and of Γ_e then Ψ is an invariant under the jump e (in interpretation $\mathcal{A}$).*
- (2) *For every flow in a mode q we can construct a universally quantified formula $\forall \bar{x} \Gamma_q(\bar{x})$ (containing also some of the parameters) such that for every structure $\mathcal{A}$ with signature $\Pi_0 \cup \Sigma \cup \Sigma' \cup \Sigma_P$ if $\mathcal{A}$ is a model of $\mathcal{T}_0$ and of Γ_q then Ψ is an invariant under flows in q (in interpretation $\mathcal{A}$).*

Proof: (1) follows from Thm. 4 for the case of jump updates. (2) Assume that $\mathcal{A}$ is a model in which Ψ is not invariant under flows in mode q . Then there exist time points $t_0 < t_1$ such that (i) $\mathcal{A} \models \forall t(t_0 \leq t \leq t_1 \rightarrow \text{Inv}_q(x(t)))$ and (ii) the interpretation in $\mathcal{A}$ of the function $x, x_A : \mathbb{R} \rightarrow \mathbb{R}$ is differentiable and has the property that $\forall t(t_0 \leq t \leq t_1 \rightarrow \frac{dx}{dt}(t) = f_q(x(t)))$. Then, by the mean value theorem:

$$\mathcal{A} \models \Psi(t_0) \wedge \forall t(t_0 \leq t \leq t_1 \rightarrow \text{Inv}_q(x(t))) \wedge \forall t, t'(t_0 \leq t < t' \leq t_1 \rightarrow \exists c(t \leq c \leq t' \wedge \frac{x(t') - x(t)}{t' - t} = f_q(x(c))) \wedge \neg \Psi(x(t_1)).$$

Therefore, $\mathcal{A}$ is a model of any set of instances of the formula above, in particular of those instances in which the universally quantified variables are instantiated with the constants $\{t_0, t_1\}$ (we can take more or fewer instances, depending on how strong we want the condition Γ_q to be). For every choice of instances for the pair of variables t, t' in the flow description above we need to replace the existentially quantified variable c with a new constant. We can now continue as in the proof of Thm. 4 and obtain the formula Γ_q . $\square$

Example 5. Consider a variant of the HA in Example 1 in which f and h are unary functions, and the invariants and flow in the two modes are described by:

Mode 1 (Heating): Invariant: $\text{Inv}_1(x(t)) := x(t) \leq T_M$

Flow: $\frac{dx}{dt}(t) = -k(x(t) - (h(t) + f(t)))$

Mode 2 (Normal): Invariant: $\text{Inv}_1(x(t)) := T_m \leq x(t)$

Flow: $\frac{dx}{dt}(t) = -k(x(t) - f(t))$

Let $\Psi(t) = T_m \leq x(t) \leq T_M$. We derive constraints which guarantee that this Ψ is invariant in mode 2. We proceed as in the proof of Theorem 7: If $\mathcal{A}$ is an interpretation in which Ψ is not invariant then there exist $t_0, t_1 \in \mathbb{R}$ such that:

- (a) $t_0 < t_1 \wedge \forall t(t_0 \leq t \leq t_1 \rightarrow T_m \leq x(t))$
- (b) $\forall t', t''(t_0 \leq t' < t'' \leq t_1 \rightarrow \exists y(t_0 \leq y \leq t' \wedge \frac{x(t') - x(t)}{t' - t} = -k(x(y) - f(y)))$
- (c) $(T_m \leq x(t_0) \wedge x(t_0) \leq T_M) \wedge (x(t_1) > T_M \vee x(t_1) < T_m)$

This also holds if we instantiate t with t_0 , and t_1 ; t' with t_0 ; and t'' with t_1 , so:

$$\mathcal{A} \models (T_m \leq x(t_0)) \wedge (T_m \leq x(t_1)) \wedge \exists y(t_0 \leq y \leq t_1 \wedge \frac{x(t_1) - x(t_0)}{t_1 - t_0} = -k(x(y) - f(y))) \wedge (T_m \leq x(t_0) \wedge x(t_0) \leq T_M) \wedge (x(t_1) > T_M \vee x(t_1) < T_m)$$

We know that there exists a value for y , say $a \in \mathbb{R}$ for which the formula holds. We introduce a new constant c ; let $\mathcal{A}^c$ be the expansion of $\mathcal{A}$ in which the interpretation of c is $c_A := a$. Then:

$$\mathcal{A}^c \models (T_m \leq x(t_0)) \wedge (T_m \leq x(t_1)) \wedge \frac{x(t_1) - x(t_0)}{t_1 - t_0} = -k(x(c) - f(c)) \wedge (T_m \leq x(t_0) \wedge x(t_0) \leq T_M) \wedge (x(t_1) > T_M \vee x(t_1) < T_m)$$

We purify the formula introducing the abbreviations: $d_0 = x(t_0)$, $d_1 = x(t_1)$ and $d = x(c)$ and $d_f = f(c)$. We obtain the following set of constraints:

$$\exists c, d, d', t_0, t_1, c_0, c_1 [T_m \leq c_0 \wedge T_m \leq c_1 \wedge \frac{c_1 - c_0}{t_1 - t_0} = -k(d - d_f) \wedge (T_m \leq c_0 \wedge c_0 \leq T_M) \wedge (c_1 > T_M \vee c_1 < T_m)]$$

We eliminate the variables t_0, t_1, c_0 , and c_1 (we used REDLOG [5]) under the assumption that $t_0 < t_1$ and $t_0 \leq c \leq t_1$ and obtain the equivalent formula:

$$\exists c, d, d' (-k(d - d_f) > 0 \wedge T_M > T_m)$$

We now replace again the constants d and d_f with the terms they represent and obtain: $\exists c (-k(x(c) - f(c)) > 0 \wedge T_M > T_m)$. We can conclude that if $T_M > T_m$ then Ψ is invariant under all flows in mode 2 if $\forall c \ k*(f(c) - x(c)) \leq 0$ (i.e. if the system does not heat in this state because of the external temperature). Of course this is only a sufficient condition.

3.3 Interconnected Families of Hybrid Automata

We can also consider systems of interconnected parametric hybrid automata $\{S_1, \dots, S_n\}$ with a parametric number of components under the assumptions:

- (1) The invariants, guards, jump and flow conditions of $S_1, \dots, S_n$ can all be expressed similarly (and can be written globally, using indices);
- (2) The relationships between the hybrid automata are uniform (and can again be expressed globally using indices);
- (3) The topology of the system can be represented using data structures (e.g. arrays, lists, trees).

A general formalization of such situations is out of the scope of this paper. We here present the ideas on an example.

Example 6. Consider a family of n water tanks with a uniform description, modeled by hybrid automata $S_1, \dots, S_n$. Assume that every S_i has one continuous variable L_i (representing the water level in S_i), and that the input and output in mode q are described by parameters in_i and out_i . Every S_i has one mode in which the water level evolves according to rule $\dot{L}_i = \text{in}_i - \text{out}_i$. We write $L(i, t), \text{in}(i)$ and $\text{out}(i)$ instead of $L_i(t), \text{in}_i$ and resp. out_i .

Assume that the water tanks are interconnected in such a way that the input of system S_{i+1} is the output of system S_i . A global constraint describing the communication of the systems is therefore:

$$\forall i(2 \leq i \leq n-1 \rightarrow (\text{in}(i) = \text{out}(i-1)) \quad \wedge \quad \text{in}(1) = \text{in}.$$

An example of a “global” update describing the evolution of the systems S_i during a flow in interval $[t_0, t_1]$:

$$\forall i(L(i, t_1) = L(i, t_0) + (\text{in}(i) - \text{out}(i))(t_1 - t_0)).$$

Let $\Psi(t) = \forall i(L(i, t) \leq L_{\text{overflow}})$. Assume that $\forall i(\text{in}(i) \geq 0 \wedge \text{out}(i) \geq 0)$. We generate a formula which guarantees that Ψ is an invariant as in the proof of Thm. 4. We start with the following formula (for simplicity of presentation we already replaced $\text{in}(i)$ with $\text{out}(i-1)$):

$$\begin{aligned} \exists t_0, t_1 \quad & t_0 < t_1 \wedge (\forall i(L(i, t_0) \leq L_{\text{overflow}}) \wedge \exists j(L(j, t_1) > L_{\text{overflow}})) \wedge \\ & \forall i((i = 1 \wedge L(1, t_1) = L(1, t_0) + (\text{in} - \text{out}(1))(t_1 - t_0)) \vee \\ & (i > 1 \wedge L(i, t_1) = L(i, t_0) + (\text{out}(i-1) - \text{out}(1))(t_1 - t_0))). \end{aligned}$$

We skolemize (replacing j with the constant i_0) and instantiate all universally quantified variables i in the formula with i_0 . After replacing $T(i_0, t_j)$ with c_j , $\text{out}(i_0 - 1)$ with d_1 , and $\text{out}(i_0)$ with d_2 we obtain:

$$\begin{aligned} \exists t_0, t_1 \exists c_0, c_1 [& t_0 < t_1 \wedge (c_0 \leq L_{\text{overflow}}) \wedge c_1 > L_{\text{overflow}} \wedge \\ & ((i_0 = 1 \wedge c_1 = c_0 + (\text{in} - d_2)(t_1 - t_0)) \vee (i_0 > 1 \wedge c_1 = c_0 + (d_1 - d_2)(t_1 - t_0)))]. \end{aligned}$$

We eliminate c_1 and c_0 using quantifier elimination and obtain:

$$\exists t_0, t_1 [t_0 < t_1 \wedge ((i_0 = 1 \wedge -(\text{in} - d_2)(t_1 - t_0) < 0) \vee (i_0 > 1 \wedge -(d_1 - d_2)(t_1 - t_0) < 0))].$$

This is equivalent (after eliminating also t_0, t_1) with:

$$((i_0 = 1 \wedge (\text{in} - d_2) > 0) \vee (i_0 > 1 \wedge (d_1 - d_2) > 0)).$$

We replace d_1, d_2 back and reintroduce the existential quantifier for i_0 :

$$\exists i_0((i_0 = 1 \wedge (\text{in} - \text{out}(i_0)) > 0) \vee (i_0 > 1 \wedge (\text{out}(i_0 - 1) - \text{out}(i_0)) > 0)).$$

The negation is $\forall i((i=1 \rightarrow (\text{in}-\text{out}(i_0)) \leq 0) \wedge (i>1 \rightarrow (\text{out}(i-1)-\text{out}(i)) \leq 0))$. This condition guarantees that Ψ is an invariant for the family of systems.

Similar results can be obtained if updates are caused by changes in topology (insertion or deletion of water tanks in the system).

4 Generating Criticality Functions

We study possibilities of proving safety properties for hybrid automata by generating so-called criticality functions, i.e. functions which measure the “distance” (in a certain sense) from the current state and a safe state and are guaranteed to decrease during the evolution of the system. If such criticality functions can be constructed then the system can be proved to be safe. For keeping the notation and the proofs simple, we consider hybrid automata with only one continuous variable x . All definitions below are formulated for this special case.

Definition 8. Let S be a hybrid automaton with a finite set Q of modes s.t.: (i) the dynamics in mode q is given by $\frac{dx}{dt}(t) = f_q(x(t))$, and (ii) the invariant of mode q is Inv_q . A family of criticality functions for this system is a family of maps $c_q : \mathbb{R} \rightarrow \mathbb{R}$, $q \in Q$ with the following properties:

- (1) $\forall x, c_q(x) \geq 0$;
- (2) for all $q \in Q$, whenever $\text{Inv}_q(x)$ holds, we have $\frac{dc_q}{dx} \frac{dx}{dt} \leq 0$;
- (3) for all mode switches $e = (q_1, q_2) \in E$ and all x, x' with $\text{guard}_e(x) \wedge \text{jump}_e(x, x')$ we have $c_{q_2}(x') \leq c_{q_1}(x)$.

Let $\text{Safe}(x)$ be a safety property expressed by a formula with free variable x , containing x but not $\dot{x}$. Then the family $\{c_q\}_{q \in Q}$ is a family of criticality functions relative to Safe if there exists $k > 0$ such that for all $q \in Q$:

- (4) For all $x \in \mathbb{R}$, $(\text{Inv}_q(x) \rightarrow (\text{Safe}(x) \leftrightarrow c_q(x) \leq k))$.

By (2) and (3), c_q will never increase throughout any run of the system. By (4), it is impossible for a run beginning in a state satisfying Safe to end in a state which does not satisfy Safe , as this would require an increase of $c_q(x)$.

Techniques used for computing Lyapunov functions can also be used for computing criticality functions (linear matrix inequalities, convex feasibility problem). In [12], in order to compute criticality functions the hybrid system is decomposed as follows: strongly connected components/cycles are detected; predicates at border nodes are used in order to obtain smaller optimization problems.

Our goal is to exploit simpler properties of the functions describing the evolution of state vectors at each mode to compute a family of criticality functions. We will focus on monotonicity properties.

Assumption 1. In what follows we assume that for every mode $q \in Q$ we have: (1) Inv_q defines a convex set and (2) if the flow condition in q is $\frac{dx}{dt}(t) = f_q(x(t))$ then the function f_q is continuous and the set $\{x \in \mathbb{R} \mid f_q(x) = 0\}$ is finite.

Remark 9. Let S be a hybrid system. Assume that the flow in mode q is given by $\frac{dx}{dt}(t) = f_q(x(t))$. Let $I \subseteq \{x \mid \text{Inv}_q(x)\}$. If $f_q(x(t)) \geq 0$ for all $t \in I$ then x is increasing on I and if $f_q(x(t)) \leq 0$ for all $t \in I$ then x is decreasing on I .

Lemma 10. *Let S be a HA satisfying Assumption 1, with real valued variable x and set of modes Q such that for every $q \in Q$ the flow in mode q is given by $\frac{dx}{dt}(t) = f_q(x(t))$. We can construct a HA S' with the same variable x and with set of modes Q' with the property that: (1) in every mode $q \in Q'$ x is monotone (either decreasing or increasing), and (2) for every run in S from state s to state s' there exists a run in S' from s to s' and vice versa.*

Proof: We construct S' as follows: For every mode q we analyze the sign changes of the function f_q . Let $x_1, y_1, \dots, x_n, y_n \in \mathbb{R}$ with $x_1 < y_1 \dots x_n < y_n$ be all the points at which f_q changes its sign. (For instance we can have $f_q(x) \leq 0$ for all $x \in (-\infty, x_1]$, $f_q(x) \geq 0$ for all $x \in [x_i, y_i]$; $f_q(x) \leq 0$ for all $x \in [y_i, x_{i+1}]$; and $f_q(x) \leq 0$ for all $x \in [y_n, \infty)$.) Let $\text{Pos}_{f_q}(x) = \{x \in \mathbb{R} \mid f_q(x) \geq 0\}$ and $\text{Neg}_{f_q}(x) = \{x \in \mathbb{R} \mid f_q(x) \leq 0\}$. Clearly, Pos_{f_q} is a union of intervals in a set I_{pos}^q and Neg_{f_q} is a union of intervals in a set I_{neg}^q . We split mode q accordingly in new modes $\{q_{i,I} \mid I \in I_{\text{pos}}^q\}$ and $\{q_{d,J} \mid J \in I_{\text{neg}}^q\}$ with invariants (in S'): $\text{Inv}_{q_{u,I}} = \text{Inv}_q \wedge C_I$ (C_I is the constraint characterizing interval I). For every $u \in \{i, d\}$ and every I , the flow in mode $q_{u,I}$ (in S') is the same as the flow in q in S . We have the same jumps from $q_{u,I}$ to $q'_{v,J}$ as between q and q' (provided the guards are compatible with C_I). In addition we can define in a natural way back and forth jumps between modes $q_{i,I}$ and $q_{d,J}$ for adjacent intervals I, J . It is easy to check that there exists a run in S from state s to state s' iff there exists a run in S' from s to s' . $\square$

In the modes $q_{i,I}$, x is increasing as a function of time, so every criticality function is decreasing (as a function of x). Similarly every criticality function is increasing (as a function of x) in the modes $q_{d,J}$ where x is decreasing.

Theorem 11. *Let S be a HA satisfying Assumption 1, and let S' be the HA obtained from S as in Lemma 10. Let $\text{Safe}(x)$ be a set of clauses with free variable x describing a safety condition and let $\mathcal{K}$ be the conjunction of the following axioms (where q ranges over the set Q' of nodes in S'):*

- (1) $\forall x(c_q(x) \geq 0)$,
- (2) axioms expressing that c_q is increasing in all modes of S' in which x is decreasing and decreasing in all modes of S' in which x is increasing,
- (3) conditions ensuring that for every control switch (q_1, q_2) , $c_{q_1}(x) \geq c_{q_2}(x')$,
- (4) the condition that $\forall x(\text{Inv}_q(x) \rightarrow (c_q(x) \leq k \leftrightarrow \text{Safe}(x)))$.

We can decide whether $\mathcal{K}$ is satisfiable. Every model of $\mathcal{K}$ yields a family of criticality functions for S' and Safe and thus also guarantees safety of S .

Proof: It is easy to check that $\mathcal{K}$ defines a local theory extension, so it is satisfiable iff the set of instances in which the variables are instantiated with constants occurring in $\mathcal{K}$ is satisfiable. If $\mathcal{K}$ is satisfiable we can use methods for model generation for obtaining a family of functions with the desired properties. $\square$

Example 7. *Consider a variant of the hybrid automaton from Example 1 where $\text{Inv}_1 = x \leq T_M$ and $\text{Inv}_2 = x \geq T_m$ and the functions describing the flow are $f_1(x) = -k(x - (h + f))$ for mode 1 and $f_2(x) = -k(x - f)$ for mode 2. Clearly,*

if $x \geq (h+f)$ then $f_1(x) \leq 0$ and if $x \leq (h+f)$ then $f_2(x) \geq 0$. We can therefore split mode 1 into two modes with the following invariants:

Mode 1_d : Invariant: $x(t) \leq T_M \wedge x(t) \geq (h+f)$

Mode 1_i : Invariant: $x(t) \leq T_M \wedge x(t) \leq (h+f)$

In mode 1_d the temperature x decreases; in mode 1_i it increases. Similarly, we can split mode 2 into two modes: 2_d (invariant: $T_m \leq x(t) \wedge x(t) \geq f$), in which the temperature x decreases and 2_i (invariant: $T_m \leq x(t) \wedge x(t) \leq f$), in which the temperature increases.

We want to find criticality functions $c_{1_d}, c_{1_i}, c_{2_d}, c_{2_i}$ satisfying the conditions in Definition 8. These conditions can be reformulated, using instead of the flow conditions in every mode only the monotonicity of the function x .

Positivity:

$\forall x, c_q(x) \geq 0$ for $q \in \{1_d, 1_i, 2_d, 2_i\}$

Flow:

$\forall x, x' \quad [(h+f \leq x \leq T_M \wedge h+f \leq x' \leq T_M \wedge x \leq x') \rightarrow c_{1_d}(x) \leq c_{1_d}(x')]$

$\forall x, x' \quad [(x \leq T_M \wedge x \leq h+f \wedge x' \leq T_M \wedge x' \leq h+f \wedge x \leq x') \rightarrow c_{1_i}(x) \leq c_{1_i}(x')]$

$\forall x, x' \quad [(T_m \leq x \wedge f \leq x \wedge T_m \leq x' \wedge f \leq x' \wedge x \leq x') \rightarrow c_{2_d}(x) \leq c_{2_d}(x')]$

$\forall x, x' \quad [(T_m \leq x \leq f \wedge T_m \leq x' \leq f \wedge x \leq x') \rightarrow c_{2_i}(x) \leq c_{2_i}(x')]$

Jumps:

$c_{1_d}(T_m) \leq c_{2_d}(T_m) \quad c_{1_i}(T_m) \leq c_{2_d}(T_m) \quad c_{1_d}(T_m) \leq c_{2_i}(T_m) \quad c_{1_i}(T_m) \leq c_{2_i}(T_m)$

$c_{2_d}(T_m) \leq c_{1_d}(T_m) \quad c_{2_i}(T_m) \leq c_{1_d}(T_m) \quad c_{2_d}(T_m) \leq c_{1_i}(T_m) \quad c_{2_i}(T_m) \leq c_{1_i}(T_m)$

$c_{1_d}(h+f) \leq c_{1_i}(h+f) \quad c_{1_i}(h+f) \leq c_{1_d}(h+f) \quad c_{2_d}(f) \leq c_{2_i}(f) \quad c_{2_i}(f) \leq c_{2_d}(f)$

$\text{Inv}_q(x) \rightarrow ((T_{\min} \leq x \wedge x \leq T_{\max}) \leftrightarrow c_q(x) \leq k)$, for $q \in \{1_d, 1_i, 2_d, 2_i\}$.

All conditions above form a set $\mathcal{K}$ of clauses which define a local theory extension. The set of clauses is satisfiable iff the set of instances in which the variables are instantiated with constants occurring in $\mathcal{K}$ is satisfiable. These constants are $T_m, f, h+f, T_M$. We use methods for constructing total models from partial ones in local theory extensions and obtain a family of functions $c_{1_d}, c_{1_i}, c_{2_d}, c_{2_i} : \mathbb{R} \rightarrow \mathbb{R}$ satisfying $\mathcal{K}$. Assume for example that the set of instances has a model in which $T_m < f \leq h+f < T_M$. Let $b_1 = h+f$ and $b_2 = f$. We can easily construct a total model for $\mathcal{K}$ by constructing for $q = 1, 2$, $c_{q_d} = c_{q_i} := c_q$, where the function c_q is strictly decreasing in the interval $(-\infty, b_q]$ and strictly increasing in the interval $[b_q, +\infty)$; $c_q(T_m) = c_q(T_M) = k$ and $c_q(b_q) = c_{q_d}(b_i) = c_{q_i}(b_q)$.

5 Conclusions

In this paper we refined methods for generating constraints on parameters which we proposed in [15,3,4] and applied then to increasingly more complex hybrid automata (parametric linear hybrid automata, parametric hybrid automata, interconnected families of hybrid automata). We then showed that we can use possibilities of automated model generation in local theory extensions for generating criticality functions in hybrid automata. We would like to continue the work analyzing possibilities of modeling systems of similar, interconnected parametric hybrid automata with a parametric number of components with a general

topology defined by using data structures, and of decomposing complex hybrid automata in order to simplify the verification tasks.

Acknowledgments. We thank Werner Damm for suggesting that simpler properties of continuous variables could be used for building criticality functions. We thank the reviewers for their helpful suggestions.

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.

References

1. Alur, R., Henzinger, T.A., Ho, P.H.: Automatic Symbolic Verification of Embedded Systems. *IEEE Trans. Software Eng.* 22(3), 181–201 (1996)
2. Cimatti, A., Roveri, M., Tonetta, S.: Requirements Validation for Hybrid Systems. In: Bouajjani, A., Maler, O. (eds.) *CAV 2009*. LNCS, vol. 5643, pp. 188–203. Springer, Heidelberg (2009)
3. Damm, W., Ihlemann, C., Sofronie-Stokkermans, V.: Decidability and complexity for the verification of reasonable linear hybrid automata. In: *Proceedings of HSCC 2011*, pp. 73–82. ACM (2011)
4. Damm, W., Ihlemann, C., Sofronie-Stokkermans, V.: PTIME Parametric Verification of Safety Properties for Reasonable Linear Hybrid Automata. *Mathematics in Computer Science* 5(4), 469–497 (2011)
5. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin* 31(2), 2–9 (1997)
6. Frehse, G., Jha, S.K., Krogh, B.H.: A counterexample-guided approach to parameter synthesis for linear hybrid automata. In: Egerstedt, M., Mishra, B. (eds.) *HSCC 2008*. LNCS, vol. 4981, pp. 187–200. Springer, Heidelberg (2008)
7. Gulwani, S., Tiwari, A.: Constraint-based approach for analysis of hybrid systems. In: Gupta, A., Malik, S. (eds.) *CAV 2008*. LNCS, vol. 5123, pp. 190–203. Springer, Heidelberg (2008)
8. Ihlemann, C., Jacobs, S., Sofronie-Stokkermans, V.: On Local Reasoning in Verification. In: Ramakrishnan, C.R., Rehof, J. (eds.) *TACAS 2008*. LNCS, vol. 4963, pp. 265–281. Springer, Heidelberg (2008)
9. Ihlemann, C., Sofronie-Stokkermans, V.: On hierarchical reasoning in combinations of theories. In: Giesl, J., Hähnle, R. (eds.) *IJCAR 2010*. LNCS, vol. 6173, pp. 30–45. Springer, Heidelberg (2010)
10. Platzer, A., Quesel, J.-D.: Logical verification and systematic parametric analysis in train control. In: Egerstedt, M., Mishra, B. (eds.) *HSCC 2008*. LNCS, vol. 4981, pp. 646–649. Springer, Heidelberg (2008)
11. Platzer, A., Quesel, J.-D.: European train control system: A case study in formal verification. In: Breitman, K., Cavalcanti, A. (eds.) *ICFEM 2009*. LNCS, vol. 5885, pp. 246–265. Springer, Heidelberg (2009)
12. Oehlerking, J., Theel, O.: Decompositional Construction of Lyapunov Functions for Hybrid Systems. In: Majumdar, R., Tabuada, P. (eds.) *HSCC 2009*. LNCS, vol. 5469, pp. 276–290. Springer, Heidelberg (2009)
13. Sofronie-Stokkermans, V.: Hierarchic reasoning in local theory extensions. In: Nieuwenhuis, R. (ed.) *CADE 2005*. LNCS (LNAI), vol. 3632, pp. 219–234. Springer, Heidelberg (2005)

14. Sofronie-Stokkermans, V.: Efficient hierarchical reasoning about functions over numerical domains. In: Dengel, A.R., Berns, K., Breuel, T.M., Bomarius, F., Roth-Berghofer, T.R. (eds.) KI 2008. LNCS (LNAI), vol. 5243, pp. 135–143. Springer, Heidelberg (2008)
15. Sofronie-Stokkermans, V.: Hierarchical reasoning for the verification of parametric systems. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 171–187. Springer, Heidelberg (2010)
16. Taly, A., Gulwani, S., Tiwari, A.: Synthesizing switching logic using constraint solving. STTT 13(6), 519–535 (2011)
17. Wang, F.: Symbolic Parametric Safety Analysis of Linear Hybrid Systems with BDD-Like Data-Structures. IEEE Trans. Software Eng. 31(1), 38–51 (2005)