PAPER Special Section on Concurrent Systems and Hybrid Systems

Deriving Discrete Behavior of Hybrid Systems under Incomplete Knowledge

Kunihiko HIRAISHI[†a)], ***Member***

SUMMARY We study analysis of hybrid systems under incomplete knowledge. The class of hybrid systems to be considered is assumed to have the form of a rectangular hybrid automaton such that each constant in invariants and guards is given as a parameter. We develop a method based on symbolic computation that computes an approximation of the discrete behavior of the automaton. We also show an implementation on a constraint logic programming language.

key words: *hybrid systems, incomplete knowledge, approximation, constraint logic programming*

1. Introduction

We focus on the situation that concrete values of constants in a hybrid system model are hard to be obtained. Typical examples are found in biological systems. Various kinds of reactions in biological systems, such as metabolic pathways, signal transduction pathways and gene regulatory interaction can be represented by hybrid systems [13]. However, it is not easy to measure precise values of constants that influence dynamics of the system. In addition, the values may vary according to the environment. *Qualitative reasoning* [7] was proposed to deal with such a situation. One of disadvantages of qualitative reasoning is that the constraints are too weak, and as a result it sometimes generates a huge number of states which are not realizable. In this paper, we aim to propose an efficient and relatively accurate method for the analysis of hybrid systems under such incomplete knowledge. In addition, we would like to have an efficient implementation of the proposing method.

For these purposes, we use the following techniques:

- In stead of using concrete values, we treat them as symbols, called parameters, where a parameter is a kind of variables such that it does not change its value through the evolution of the system. Incomplete knowledge is described as constraints on parameters such as a partial order on them. In addition, we deal with parameters and constraints by symbolic computation, i.e., constraints are truly evaluated as it is.
- In general, parametric analysis of hybrid systems is harder than the case that concrete values are given. For example, the emptiness problem (given a parametric hybrid automaton, are there concrete values for the parameters so that the automaton has an accepting run?) is undecidable even if we restrict the target to a primitive class of hybrid systems such as timed automata [3]. Under incomplete knowledge on the model, however, it is sufficient to have an approximation of the behavior. We develop a method that computes an approximation of the discrete behavior of the model. The approximated behavior is described in the form of a finite transition system and the computation always terminates. Moreover, we can control the degree of accuracy in the approximation.
- For the implementation, we use a constraint logic programming language with a linear constraint solver.

Manuscript received April 7, 2004.
Final manuscript received June 16, 2004.

[†]The author is with the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.
a) E-mail: hira@jaist.ac.jp

2. Hybrid Systems

In this paper, we follow the definition of hybrid systems described in [2]. A hybrid system is $H = (V, n, Q_0, F, Inv, R)$, where

- V is a finite set of *locations*, and n is a nonnegative integer called *the dimension* of H. Each state of H is a pair (l, x), where $l \in V$ is *the discrete part* of the state and $x \in \mathbb{R}^n$ is *the continuous part* of the state. Let $Q = V \times \mathbb{R}^n$ be called *the state space*.
- $Q_0 \subseteq Q$ is the set of *initial states*.
- $F : Q \to 2^{\mathbb{R}^n}$ assigns to each state (l, x) a set $F(l, x) \subseteq \mathbb{R}^n$, which constrains the time derivative of the continuous part of the state by $\dot{x} \in F(l, x)$ in location l.
- $Inv : V \to 2^{\mathbb{R}^n}$ assigns to each location l *an invariant set* $Inv(l) \subseteq \mathbb{R}^n$, which constrains the value of the continuous part of the state in location l.
- $R \subseteq V \times V$ is a relation capturing discrete state changes.

We refer to the n individual coordinates of the continuous part $\mathbb{R}^n$ of the state space as real-valued *variables*, and we view the continuous part $x = (x_1, \cdots, x_n)$ of a state as an assignment of values to the variables.

A hybrid systems is usually represented as a finite directed graphs $G = (V, E)$ where

- The set of vertices is the set of locations V. With each location l, let $Init(l) := \{x \in Inv(l) \mid (l, x) \in Q_0\}$.
- $E := \{(l, l') \in V \times V \mid \exists x \in Inv(l), x' \in Inv(l') : ((l, x), (l', x')) \in E\}$. With each edge $e = (l, l') \in E$, we associate *a guard set* $Guard(e) := \{x \in Inv(l) \mid \exists x' \in Inv(l) : ((l, x), (l', x')) \in R\}$, and *a reset map* $Reset(e, x) := \{x' \in Inv(l') \mid ((l, x), (l', x')) \in R\}$.

Each trajectory of H originates at an initial state $(l, x) \in Q_0$. As long as $x \in Inv(l)$, x evolves over time according to the differential inclusions $\dot{x} \in F(l, x)$. When $x \in Guard(e)$ for some $e = (l, l') \in E$, the edge e becomes *enabled* and the state instantaneously jump from (l, x) to (l', x') with $x' \in Reset(e, x)$.

Formally, the behavior of H is described in the form of a transition system $T_H = (Q, \rightarrow, Q_0)$, where the transition relation $\rightarrow = \xrightarrow{d} \cup \xrightarrow{\tau}$ is defined as follows:

- *Discrete transitions*: $(l, x) \xrightarrow{d} (l', x')$ if and only if for $e = (l, l') \in E$: $x \in Guard(e)$ and $x' \in Reset(e, x)$.
- *Continuous transitions*: $(l, x^1) \xrightarrow{\tau} (l', x^2)$ if and only if $l = l'$ and there exits a real $\delta \geq 0$ and a differential curve $x : [0, \delta] \rightarrow \mathbb{R}^n$ with $x(0) = x^1, x(\delta) = x^2$, for all $t \in [0, \delta]$ we have $x(t) \in Inv(l)$ and for all $t \in (0, \delta)$ we have $\dot{x} \in F(l, x(t))$.

A discrete trajectory of H is a finite sequence of locations $l_{(1)} \cdots l_{(m)} \in V^*$ such that for some $x^{in(1)}, x^{out(1)}, \cdots, x^{in(m-1)}, x^{out(m-1)}, x^{in(m)} \in \mathbb{R}^n$: (i) $(l_{(1)}, x^{in(1)}) \in Init(l_{(1)})$, and (ii) $(l_{(j)}, x^{in(j)}) \xrightarrow{\tau} (l_{(j)}, x^{out(j)})$, $(l_{(j)}, x^{out(j)}) \xrightarrow{d} (l_{(j+1)}, x^{in(j+1)})$ $(i = 1, \cdots, m-1)$. Let $L_d(H) \subseteq V^*$ be the set of all discrete trajectories of H.

Example 2.1: Figure 1 shows a hybrid system representing a temperature control system [1]. The system controls a reactor tank by moving two independent control rods. The goal is to maintain the coolant between the temperatures θ_m and θ_M $(\theta_m < \theta_M)$. When the temperature reaches its maximum value θ_M, the tank must be refrigerated with one of the rods. When no rod is activated, the temperature rises according to a differential equation $\dot{\theta} = \theta/k + W$, where k and W are constants determined by the reactor tank. If rod $i \in \{1, 2\}$ is activated, then the temperature decreases according to $\dot{\theta} = \theta/k - R_i$, where R_i is a constant determined by control rod i and the reactor tank. A rod can be moved again only if T time units have elapsed since the end of its previous movement. If the temperature of the coolant cannot decrease because there is no available rod, a complete shutdown is required.

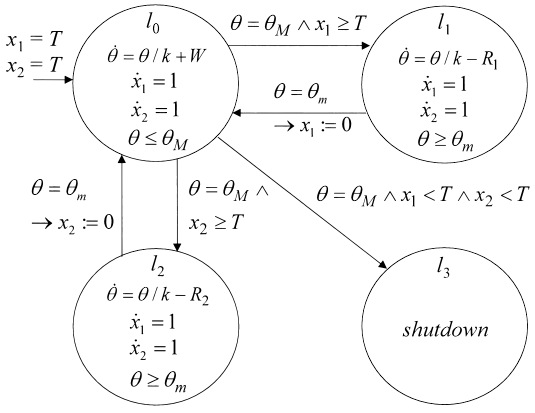


Fig. 1 A temperature control system.

3. Approximation Method

3.1 Parametric Rectangular Automata

The start point of the proposing method is approximation of the target system by a parametric model. Let $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$ be the continuous part of the state. *A rectangular set* is a conjunction of linear inequalities of the form $x_i \approx c$, where $\approx$ is one of $<, \leq, =, \geq >$ and $c \in \mathbb{R}$. A variable x_i is called *a parameter* if the value of x_i does not change in all locations and in all discrete jumps. That is, it is a symbolic constant. *A parametric rectangular set* is a conjunction of linear inequalities of the form $x_i \approx p$, where p is a parameter. For a (parametric) rectangular set B, let B_i be its projection onto the ith coordinate. Thus a rectangular set $B \subseteq \mathbb{R}^n$ is of the form $B = B_1 \times \cdots \times B_n$, where each B_i is a bounded or unbounded interval.

We assume that the target system can be *approximated* in the form of a class of hybrid system, called *a parametric rectangular automaton*, defined as follows:

- For every location l, the set $Init(l)$ and $Inv(l)$ are parametric rectangular sets.
- For every location l, there is a rectangular set B^l such that $F(l, x) = B^l$ for all $x \in \mathbb{R}^n$.
- For every edge e, the set $Guard(e)$ is a parametric rectangular set, and there is a parametric rectangular set B^e and a subset $J^e \subseteq \{1, \cdots, n\}$ such that for all $x \in \mathbb{R}^n$: $Reset(e, x) = \{(x'_1, \cdots, x'_n) \mid$ for all $1 \leq i \leq n$, if $i \in J^e$ then $x'_i \in B^e_i$ else $x'_i = x_i\}$.

In a parametric rectangular automaton, the derivative of each variable stays between fixed bounds, which may be different in different locations. With each discrete jump across an edge e, the value of a variable x_i is either left unchanged (if $i \notin J^e$), or reset nondeterminiscally to a new value within some interval B^e_i determined by parameters (if $j \in J^e$).

The idea behind the approximation is described as follows:

- Discrete jumps happen when values of some variables reach to borders (e.g., θ_m, θ_M and T in Fig. 1). We do not know the concrete values of them, but know only the partial information ($\theta_m < \theta_M$ and $T > 0$).
- For the derivative of each variable, we assume that relative magnitude of it is known as a fixed interval (e.g., $3 \leq \dot{\theta} \leq 5$). We remark that absolute values are not necessary because all borders are given as parameters. For example, $\dot{x}_i = 8, \dot{x}_j = 4$ with guards $x_i \leq b_i, x_j \geq b_j$ are equivalent to $\dot{x}_i = 2, \dot{x}_j = 1$ with guards $x_i \leq b'_i, x_j \geq b'_j$, where $b'_i = b_i/4$ and $b'_j = b_j/4$.

Figure 2 is an approximated model of the hybrid system in Fig. 1.

Now the problem to be considered is described as follows: given a parametric rectangular automaton H, find a set $\widetilde{L}_d(H) \subseteq V^*$ such that $L_d(H) \subseteq \widetilde{L}_d(H)$. In addition, find

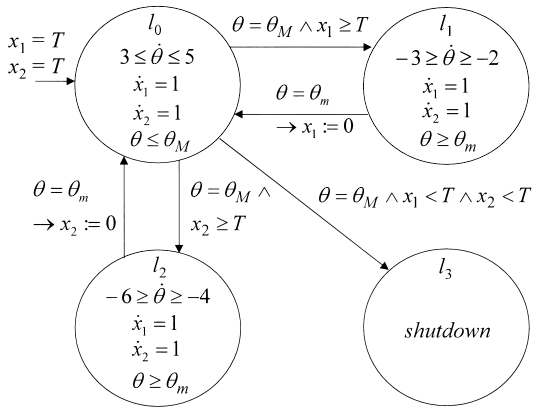


Fig. 2 A parametric rectangular automaton.

conditions on H so that the equality holds.

There are several semantics proposed for hybrid automata, such as timed transition semantics, time-abstract transition semantics, and timed tube semantics [9], [10]. Since the proposing approach checks feasibility of each discrete jump only, we have adopted a semantics in which all continuous transitions are considered silent. This semantics suffices for knowing qualitative behavior of the system.

3.2 Constraint Transition Graphs

The main step of the proposing method is approximation of a given parametric rectangular automaton by a finite transition system, called *a constraint transition graph*.

Each discrete trajectory of a hybrid system gives a constraint on parameters. Consider the hybrid system in Fig. 2. Suppose that $\cdots l_1 l_0 l_1 \cdots \in L_d(H)$. Then the following (in)equalities must be true:

$$\begin{aligned}
&\theta^{in} + 3t \le \theta^{out} \le \theta^{in} + 5t \wedge \\
&x_1^{out} = x_1^{in} + t \ \wedge \ x_2^{out} = x_2^{in} + t \wedge \\
&\theta^{in} = \theta_m \ \wedge \ \theta^{out} = \theta_M \wedge \\
&x_1^{in} = 0 \ \wedge \ x_1^{out} \ge T
\end{aligned}$$

where $\theta^{in}, x_1^{in}, x_2^{in}$ are values of variables θ, x_1, x_2 when the system goes into location l_0, $\theta^{out}, x_1^{out}, x_2^{out}$ are values when the system goes out of l_0, and t is the duration of time elapsed in location l_0.

Now we apply existential quantifier to variables $\theta^{in}, \theta^{out}, x_1^{in}, x_1^{out}, x_2^{in}, x_2^{out}, t$:

$$\begin{aligned}
&\exists \theta^{in}, \theta^{out}, x_1^{in}, x_1^{out}, x_2^{in}, x_2^{out}, t : \\
&\theta^{in} + 3t \le \theta^{out} \le \theta^{in} + 5t \ \wedge \\
&x_1^{out} = x_1^{in} + t \ \wedge \ x_2^{out} = x_2^{in} + t \ \wedge \\
&\theta^{in} = \theta_m \ \wedge \ \theta^{out} = \theta_M \ \wedge \\
&x_1^{in} = 0 \ \wedge \ x_1^{out} \ge T \\
&\quad \equiv \exists \theta^{in}, \theta^{out}, x_1^{in}, x_1^{out}, x_2^{in}, x_2^{out} : \\
&(\theta^{out} - \theta^{in})/5 \le x_1^{out} - x_1^{in} \le (\theta^{out} - \theta^{in})/3 \ \wedge \\
&x_1^{out} - x_1^{in} = x_2^{out} - x_2^{in} \ \wedge \\
&\theta^{in} = \theta_m \ \wedge \ \theta^{out} = \theta_M \ \wedge \\
&x_1^{in} = 0 \ \wedge \ x_1^{out} \ge T \\
&\quad \equiv \exists x_1^{out} : \\
&(\theta_M - \theta_m)/5 \le x_1^{out} \le (\theta_M - \theta_m)/3 \ \wedge \\
&x_1^{out} \ge T \\
&\quad \equiv \theta_M - \theta_m \ge 3T.
\end{aligned}$$

Since all (in)equalities are linear, we can use efficient *quantifier elimination techniques* to perform above calculation [6].

Similarly, we obtain constraint $(\theta_M - \theta_m)/3 + (\theta_M - \theta_m)/4 + (\theta_M - \theta_m)/3 \ge T \equiv 11(\theta_M - \theta_m) \ge 12T$ from trajectory $l_1 l_0 l_2 l_0 l_1$, and constraint $(\theta_M - \theta_m)/5 + (\theta_M - \theta_m)/6 + (\theta_M - \theta_m)/5 < T \equiv 17(\theta_M - \theta_m) < 30T$ from trajectory $l_1 l_0 l_2 l_0 l_3$. On the other hand, trajectory $l_1 l_0 l_2$ gives no constraint, i.e., constraint is always *true*, because x_1 is unconstrained when the system goes out of l_0 and x_2 is unconstrained when the system goes into l_0.

Given a hybrid system H, let $V_H^k \subseteq V^*$ denote the set of all sequence of locations $l_{(1)} l_{(2)} \cdots l_{(j)}$ such that $(l_{(i)}, l_{(i+1)}) \in E$ for $i = 1, \cdots, j-1$ and $j \le k$. That is, each element in V_H^k is a fragment of discrete trajectories of H with a length less than or equal to k. For each $u \in V_H^k$, let $C(u)$ denote the derived constraint. The idea of the proposing approximation is to restrict the consideration on constraints to those derived from V_H^k.

Note that if $u' \in V_H^k$ is a subsequence of $u \in V_H^k$, i.e., there exists $v, w \in V^*$ such that $u = vu'w$, then $C(u) \to C(u')$ holds, where $\to$ is the logical implication. This means that we need only sequences of length k when we intend to extract all constrains derived from V_H^k. However, we need to take care of initial fragments of trajectories, because initial fragments with a length less than k may not be contained in other fragments of length k. For this purpose, we introduce a special symbol ϕ. Suppose $k = 5$ and $l_0 l_1 l_0 l_2 l_0 l_1 \in L_d(H)$. Along this trajectory, we prepare the following sequence of fragments: $\langle \phi\phi\phi\phi\phi \rangle$, $\langle \phi\phi\phi\phi l_0 \rangle$, $\langle \phi\phi\phi l_0 l_1 \rangle$, $\langle \phi\phi l_0 l_1 l_0 \rangle$, $\langle \phi l_0 l_1 l_0 l_2 \rangle$, $\langle l_0 l_1 l_0 l_2 l_0 \rangle$, $\langle l_1 l_0 l_2 l_0 l_1 \rangle$. For a trajectory $l_{(1)} \cdots l_{(m)} \in V^*$ and a positive integer k, let s/k denote the sequence of fragments obtained from s, i.e., $s/k = \langle \phi^k \rangle, \langle \phi^{k-1} l_{(1)} \rangle, \langle \phi^{k-2} l_{(1)} l_{(2)} \rangle, \cdots, \langle l_{(1)} \cdots l_{(k)} \rangle, \cdots,$ $\langle l_{(m-k+1)} \cdots l_{(m)} \rangle$. Let $\widetilde{V}_H^k$ denote the set of all such fragments of trajectories.

Now we define *the constraint transition graph* of degree k as $CTG_H^k = (\widetilde{V}_H^k, \widetilde{E}, \psi, \langle \phi^k \rangle)$, where

- $\widetilde{E} \subseteq \widetilde{V}_H^k \times \widetilde{V}_H^k$ denotes the set of edges defined by $(\langle l_{(1)} \cdots l_{(k)} \rangle, \langle l'_{(1)} \cdots l'_{(k)} \rangle) \in \widetilde{E}$ if and only if $l'_{(i)} = l_{(i+1)} (i = 1, \cdots, k-1)$ and $(l_{(k)}, l'_{(k)}) \in E$.
- $\psi : \widetilde{E} \to C(\widetilde{V}_H^k) \cup \{true\}$ denotes a function that assigns a constraint to each edge and is defined by $\psi((u, u')) = C(u')$ if $C(u') \ne C(u)$, $\psi((u, u')) = true$ otherwise.

For the hybrid system in Fig. 2, we obtain the constraint transition graph of degree 5 shown in Fig. 3, where constraints assigned to edges are: (1) $\theta_M - \theta_m \ge 3T$, (2) $\theta_M - \theta_m < 5T$, (3) $11(\theta_M - \theta_m) \ge 12T$, (4) $17(\theta_M - \theta_m) < 30T$, (5) $\theta_M - \theta_m \ge 3T$, (6) $\theta_M - \theta_m < 5T$, (7) $7(\theta_M - \theta_m) \ge 6T$,

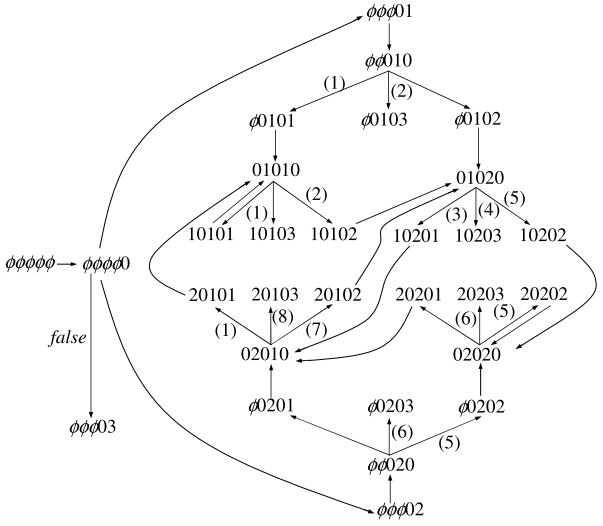


Fig. 3 The constraint transition graph of degree 5.

(8) $11(\theta_M - \theta_m) < 15T$.

3.3 Theoretical Guarantee on Approximation

Let $L_d(CTG_H^k) \subseteq V^*$ denote the set of all discrete trajectories of CTG_H^k, which is defined as follows: $s = l_{(1)} \cdots l_{(m)} \in V^*$ is in $L_d(CTG_H^k)$ if and only if (i) s/k is a path on CTG_H^k, and (ii) $C(s/k) := \wedge_i C(u_i)$ is satisfiable, where $s/k = u_1, \cdots, u_r$,

The following lemma is obvious from the definition because the constraint transition graph evaluates a subset of constraints.

Lemma 3.1: $L_d(H) \subseteq L_d(CTG_H^k)$ for any positive integer k.

Since any constraint evaluated in CTG_H^k is also evaluated in CTG_H^{k+1}, we have

Lemma 3.2: $L_d(CTG_H^{k+1}) \subseteq L_d(CTG_H^k)$ for any positive integer k.

This result implies that the degree k can be used as a control parameter for the accuracy in the approximation.

However, there exits a hybrid system H such that $L_d(H) \neq L_d(CTG_H^k)$ for any positive integer k. One of the reasons is that the system may produce infinitely many constraints. Let us consider the hybrid system in Fig. 2. Since there is a cycle $l_2 l_0 l_2$ in which variable x_1 is not *initialized*, i.e., not reset to a new value, the following constraint is derived from trajectory $l_1 l_0 (l_2 l_0)^k l_3$ where k is any positive integer: $k \cdot ((\theta_M - \theta_m)/5 + (\theta_M - \theta_m)/6) + (\theta_M - \theta_m)/5 < T$.

We consider a class of parametric rectangular automata such that the equality $L_d(H) = L_d(CTG_H^k)$ holds. Unfortunately, such a class is quite limited. One of such classes is a parametric rectangular automata in which all variables are *simultaneously initialized* at some point in the automaton. Note that a guard condition with equality like $x_i = c$ is considered to be with an initialization, i.e., $x_i = c \rightarrow x_i := c$. We define a complexity measure $\xi(H)$ of a parametric rectangular automaton H by the maximal length of fragments of discrete trajectories in which at least one variable is not initialized on each edge. If the length of a fragment s is greater than $\xi(H)$, then all variables are simultaneously initialized at least once in s.

Theorem 3.3: Suppose that H is a parametric rectangular automaton. If $\xi(H) \neq \infty$, then $L_d(H) = L_d(CTG_H^{\xi(H)+2})$.

Proof. Let $k = \xi(H) + 2$. By Lemma 3.2, $L_d(H) \subseteq L_d(CTG_H^k)$ holds. Let $s = l_{(1)} \cdots l_{(m)} \in L_d(CTG_H^k)$. If $m \leq k$, then obviously $s \in L_d(H)$. We consider the case that $m > k$. Then $s \in L_d(H)$ if and only if $C(\langle \phi l_{(1)} \cdots l_{(m)} \rangle) \equiv C(s/k) \equiv true$. Note that $C(\langle \phi l_{(1)} \cdots l_{(m)} \rangle) \rightarrow C(s/k)$ is always true and we need to prove the converse.

Let $x = (x_1, \cdots, x_n)$ be the continuous part of the state. Let $x^{in(j)} = (x_1^{in(j)}, \cdots, x_n^{in(j)})$ and $x^{out(j)} = (x_1^{out(j)}, \cdots, x_n^{out(j)})$ denote the values of x when the system goes into location $l_{(j)}$ and goes out of $l_{(j)}$, respectively. Suppose that the system goes along the discrete trajectory s. Let $f_{(j)}$ be the constraint that must hold when the system goes through location $l_{(j)}$. As we have seen in 3.2, $f_{(j)}$ is a conjunction of (in)equalities over $x^{in(j)}, x^{out(j)}$ and parameters. Note that if a variable x_i is not initialized on edge $(l_{(j)}, l_{(j+1)})$, then $f_{(j+1)}$ has equality $x_i^{out(j)} = x_i^{in(j+1)}$ and therefore $x_i^{out(j)}$ is contained in both $f_{(j)}$ and $f_{(j+1)}$. Otherwise, $x_i^{out(j)}$ is contained in $f_{(j)}$ only.

Then $C(\langle \phi l_{(1)} \cdots l_{(m)} \rangle)$ is of the form $\exists x^{in(j)}, x^{out(j)}$ $(j = 1, \cdots, m) : f_{(1)} \wedge \cdots \wedge f_{(m)}$. Since $m > k = \xi(H)$, there are locations $l_{(r_1)}, \cdots, l_{(r_h)}$ at which all variables are simultaneously initialized before entering them. Therefore, $C(\langle \phi l_{(1)} \cdots l_{(m)} \rangle)$ can be written in the form $F_0 \wedge \cdots \wedge F_h$ such that

$$F_i \equiv \exists x^{in(j)}, x^{out(j)} (j = r_i, r_i + 1, \cdots, r_{i+1} - 1) : f_{(r_i)} \wedge f_{(r_i+1)} \wedge \cdots \wedge f_{(r_{i+1}-1)}$$

where $r_0 = 1$ and $r_{h+1} = m + 1$. Since $r_{i+1} - r_i \leq \xi(H)$ for any $i = 0, \cdots, h$, there exists $u \in s/k$ such that u contains $l_{(r_i-1)} l_{(r_i)} \cdots l_{(r_{i+1}-1)} l_{(r_{i+1})}$, where $l_{(0)} = \phi$ and $l_{(m+1)}$ is empty, as a subsequence and therefore $C(u) \rightarrow F_i$ holds. Hence, we have $C(s/k) \rightarrow C(\langle \phi l_{(1)} \cdots l_{(m)} \rangle)$. ∎

3.4 State Space Representation

To check properties on discrete trajectories in $L_d(CTG_H^k)$, we need a finite representation of it. For this purpose, we define a finite transition system $\widetilde{T}_H^k = (\widetilde{Q}, \stackrel{\tilde{d}}{\rightarrow}, \widetilde{q}_0)$ by

- $\widetilde{Q} = \widetilde{V}_H^k \times 2^{\psi(\widetilde{E})}$, i.e., each state is a pair (u, C) where u is a state of CTG_H^k and C is a subset of all constraints appeared in CTG_H^k.
- $\stackrel{\tilde{d}}{\rightarrow} \subseteq \widetilde{Q} \times \widetilde{Q}$ is defined by $(u, C) \stackrel{\tilde{d}}{\rightarrow} (u', C')$ if and only if (i) $(u, u') \in \widetilde{E}$, (ii) $\psi((u, u')) \wedge (\wedge_{C_i \in C} C_i)$ is satisfiable, and (iii) $C' = C \cup \psi((u, u'))$.
- $\widetilde{q}_0 = (\langle \phi^k \rangle, \emptyset)$.

Then $s = l_{(1)} \cdots l_{(m)} \in L_d(CTG_H^k)$ if and only if there exists a trajectory $\widetilde{q}_0 \stackrel{\tilde{d}}{\rightarrow} (u_1, C_1) \stackrel{\tilde{d}}{\rightarrow} \cdots \stackrel{\tilde{d}}{\rightarrow} (u_m, C_m)$ on $\widetilde{T}_H^k$ such that $s/k = \langle \phi^k \rangle, u_1, \cdots, u_m$.

4. Implementation

The proposed method can be implemented easily on constraint logic programming languages. Constraint logic programming (CLP) is a merger of two declative paradigms: constraint solving and logic programming [6]. The syntax of CLP language is similar to that of logic programming language like Prolog, except that various kind of numerical and/or algebraic constraints can be described in the program. There have been several researches on application of CLP to the analysis of hybrid systems [11], [14].

We here show an implementation on a constraint logic programming language Keyed CLP [12]. Keyed CLP is equipped with a linear constraint solver based on the simplex method, and allows users to write linear equalities and inequalities on the Real field directly in the program. In addition, Keyed CLP has a different type of predicates, called keyed predicates, which has the form Predicate-Name $(Key_1, \cdots, Key_n : Arg_1, \cdots, Arg_m)$. Each predicate in logic programming can be seen as a relational table in a database. A keyed predicate corresponds to a tuple with $Key_1, \cdots, Key_n$ as the key attributes. In a relational table, a tuple is uniquely determined if all the values of the key attributes are specified. This property is preserved in the execution of Keyed CLP, i.e., every predicate which has the same predicate name and the same key values must have the same values of $Arg_1, \cdots, Arg_m$. If the key part is empty, then the values of $Arg_1, \cdots, Arg_m$ are unique on every computation paths. This mechanism allows us to introduce global variables into logic programming.

The following is a source code of Keyed CLP that computes the approximated state space $\widetilde{T}_H^k$ from the constraint transition graph in Fig. 3.

```
/* simulator */
exec(terminal, V):-!.
exec(ID, V):- lookup(stack(ID, V :)),!.
exec(ID, V):- stack(ID, V :),
  link(ID, ID1, N), constraint(N, V, V1),
  write([[ID, V], [ID1, V1]]), nl,
  exec(ID1, V1).

go:- exec(snnnnn, [0,0,0,0,0,0,0,0]), fail.
go.

/* link */
link(snnnnn, snnnn0, 0).
link(snnnn0, snnn01, 0).
link(snnnn0, snnn02, 0).
link(snnn01, snn010, 0).
link(snn010, sn0101, 1).
...
link(s20203, terminal, 0).

/* parameters and stack */
param(: TM, Tm, T):- TM > Tm, T > 0.
stack(_, _ :).

/* constraints */
constraint(0, V, V):-!.
constraint(1, [A, B, C, D, E, F, G, H],
              [1, B, C, D, E, F, G, H]):-
  (A = 1; param(: TM, Tm, T),
          TM - Tm >= 3 * T), !.
constraint(2, [A, B, C, D, E, F, G, H],
              [A, 1, C, D, E, F, G, H]):-
  (B = 1; param(: TM, Tm, T),
          TM - Tm < 5 * T), !.
...
constraint(8, [A, B, C, D, E, F, G, H],
              [A, B, C, D, E, F, G, 1]):-
  (H = 1; param(: TM, Tm, T),
          11 * (TM - Tm) < 15 * T), !.
```

The algorithm explores states in a depth-first manner. Predicate `exec()` is the main part of the program. The graph structure of the constraint transition graph is given by predicates `link()`. Satisfiability of constraints are checked in predicate `constraint()`. It refers a keyed predicate `param(: TM, Tm, T)` with empty key, which is used as global variables. Visited states are stored in a system stack defined by a keyed predicate `stack(ID, V :)`, where `ID` and `V` correspond to elements in $\widetilde{V}_H^k$ and $2^{\psi(\widetilde{E})}$, respectively. If we call keyed predicate `stack(ID, V :)` with values assigned to `ID` and `V`, then it is automatically push onto a system stack, and the status of the stack can be checked by system predicate `lookup()`. In the second line of `exec()`, calling `lookup(stack(ID, V :))` checks if the current state is already in the stack or not. If it is true, then `exec()` succeeds and backtrack occurs by calling `fail` in `go`. Otherwise, the current state is push onto the stack (calling `stack(ID, V :)` in the third line of `exec()`) and successors are explored. Output of the program is the set of all pairs $(\widetilde{q}, \widetilde{q}')$ of states such that $\widetilde{q} \stackrel{\tilde{d}}{\rightarrow} \widetilde{q}'$ and $\widetilde{q}$ is reachable from $\widetilde{q}_0$.

5. Related Work

QSIM [7] is one of typical algorithms for qualitative reasoning. It describes the system by a collection of *qualitative differential equations*. Each variable is a function of time and the value at each time step is represented as a pair $(qmag, qdir)$, where $qmag$ takes a value from a totally ordered set of symbols, called *landmark values*, and $qdir$ is the sign of its derivative. Differential equations are described by constraints on qualitative values. For example, constraint `(d/dt x y)`, which means $dx/dt = y$, is true if and only if $qdir$ of x is equal to $sign(y)$, where $sign(y)$ is determined from $qmag$ of y.

The behavior of QSIM algorithm is outlined as follows: It generates all successors without considering constraints, and eliminate some of them which are inconsistent. Repeat

this process until no new states are obtained. It was shown that the behavior generated by QSIM contains the real behavior. Comparing with QSIM, the proposing method is also an exploration on discrete state space but it takes account numerical constraints represented by linear inequalities. Computational cost is not so high because constraints are restricted to be linear. Moreover, we can control the degree of accuracy in the simulation.

Discrete abstraction of state space is also an approach to make the state space to be a discrete set [2]. It partitions the state space by an equivalence relation called *bisimulation*. If a finite bisimulation is obtained, then we can know discrete behavior from the abstracted transition system. However, there is no guarantee that a finite bisimulation exists.

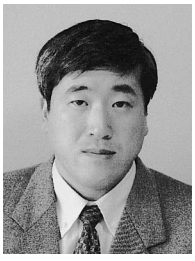
As an approximation method for analyzing linear hybrid systems, a method based on *convex approximation* was proposed [8]. It computes upper approximations containing the convex hull of the reachability set, and the computation always terminates. While the convex approximation directly manipulate the state space, the proposing method works on the parameter space which has a smaller dimension than that of the state space. In addition, the method in [8] focuses on the reachability set only, not on the dynamics of the model such as discrete trajectories.

6. Conclusion

The proposing approach is also applicable to more general class of parametric hybrid systems in which invariants and guards are defined as linear sets instead of rectangular sets. For nonlinear hybrid systems, we need to have a rectangular approximation of the system. Accuracy of the result strongly depends on this step. Therefore, we need to develop a systematic way to do this. It was shown that quantifier elimination technique can be applied to larger classes than rectangular automata, such as hybrid automata with parametric inhomogeneous linear differential systems at each location [4], [5]. We will be able to incorporate this result with the proposed approach.

References

[1] R. Alur, et al., "The algorithmic analysis of hybrid systems," Theoretical Computer Science, vol.138, pp.3–34, 1995.
[2] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas, "Discrete abstraction of hybrid systems," Proc. IEEE, vol.88, no.7, pp.971–984, 2000.
[3] R. Alur and T.A. Henzinger, "Parametric real-time reasoning," Proc. 25th Annual ACM Symposium on Theory and Computing, pp.592–601, 1993.
[4] H. Anai and V. Weispfenning, "Reach set computations using real quantifier elimination," Proc. HSCC2001, LNCS 2034, pp.63–76, 2001
[5] H. Anai and V. Weispfenning, "On reachability problem for linear hybrid systems," Trans. Institute of Systems, Control and Information Engineers, vol.15, no.3, pp.109–116, 2002.
[6] J. Jaffar and M.J. Maher, "Constraint logic programming: A survey," J. Logic Programming, vol.19/20, pp.503–581, 1994.
[7] B. Kuipers, Qualitative Reasoning, The MIT Press, 1994.
[8] N. Halbwachs, Y. Proy, and P. Raymond, "Verification of linear hybrid systems by means of convex approximations," Lecture Notes in Computer Science, vol.818, pp.222–237, 1994.
[9] T.A. Henzinger, "The theory of hybrid automata," Proc. 11th Annual Symposium on Logic in Computer Science (LICS), pp.278–292, 1996.
[10] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?," J. Comput. Syst. Sci., vol.57, pp.94–124, 1998.
[11] T.J. Hickey and D.K. Wittenberg, "Modeling hybrid systems using analytic constraint logic programming," Technical Report, Dep. Computer Science, Brandeis Univ., 2002.
[12] K. Hiraishi, "A constraint logic programming language Keyed CLP and its applications to decision making problems in OR/MS," Decision Support Systems, vol.14, pp.269–281, 1995.
[13] H. Matsuno, S. Fujita, A. Doi, M. Nagasaki, and S. Miyano, "Towards biopathway modeling and simulation," Lecture Notes in Computer Science, vol.2679, pp.3–22, 2003.
[14] L. Urbina, "Analysis of hybrid systems in CLP(R)," Lecture Notes in Computer Science, vol.1118, pp.451–467, 1996.



Kunihiko Hiraishi received from the Tokyo Institute of Technology the B.E. degree in 1983, the M.E. degree in 1985, and D.E. degree in 1990. He is currently a professor at School of Information Science, Japan Advanced Institute of Science and Technology. His research interests include discrete event systems and formal verification. He is a member of the IEEE, IPSJ, and SICE.