# A Note on Abstract Interpretation Strategies for Hybrid Automata[⋆] [⋆⋆]

Thomas A. Henzinger[⋆⋆⋆] and Pei-Hsin Ho

Computer Science Department, Cornell University, Ithaca, NY 14853
(tah|ho)@cs.cornell.edu

**Abstract.** We report on several abstract interpretation strategies that are designed to improve the performance of HyTech, a symbolic model checker for linear hybrid systems. We (1) simultaneously compute the target region from different directions, (2) conservatively approximate the target region by dropping constraints, and (3) iteratively refine the approximation until sufficient precision is obtained. We consider the standard abstract convex-hull operator and a novel abstract extrapolation operator.

## 1 Introduction

Hybrid automata extend finite automata with continuous activities [ACHH93]. Consider a communication protocol $A$ with $k$ propositional variables (values 0 or 1), and a robot $B$ with $k$ propositional control variables and $m$ real-valued status variables (e.g., the position of the robot). Let $d = k + m$. A state of $A$ is a point in $k$-dimensional *boolean* space; a state of $B$, a point in $d$-dimensional *euclidean* space. Each transition of $A$ is a $k$-dimensional boolean transformation; if $B$ is *linear*, then each discrete control transition of $B$ is a $d$-dimensional linear transformation and each continuous activity of $B$ is a $d$-dimensional piecewise-linear motion. This geometric intuition inspires the abstract interpretation strategies we develop.

For the finite-state system $A$, model-checking algorithms compute the state set (*target region*) that satisfy a given temporal requirement specification, by iterative approximation of the target region as a fixpoint. This is done either enumeratively [CES86] or symbolically, by representing state sets (*regions*) as boolean expressions [BCM$^+$92]. Since the state space of the linear hybrid automaton $B$ is infinite, enumerative model checking is no longer possible, and regions must be represented symbolically. A symbolic model-checking algorithm for linear hybrid automata has been developed [HNSY94, ACH$^+$95],

[⋆] This research was supported in part by the National Science Foundation under grant CCR-9200794, by the Air Force Office of Scientific Research under contract F49620-93-1-0056, by the Office of Naval Research under YIP grant N00014-95-1-0520, and by the Defense Advanced Research Projects Agency under grant NAG2-892.

[⋆⋆] This paper appeared in *Hybrid Systems II* (P. Antsaklis, W. Kohn, A. Nerode, S. Sastry, eds.), *Lecture Notes in Computer Science* **999**, Springer-Verlag, 1995, pp. 252–264.

[⋆⋆⋆] Phone: (607) 255-3009. FAX: (607) 255-4428.

and implemented (HyTech) by representing regions as real-valued linear expressions [AHH93]. The linearity condition is the key so that if the target region is a finitely approximable fixpoint, then it can be computed in the theory of the reals with addition, at "tolerable" cost. The most serious limitation of finite-state model checking, the state-explosion problem, reoccurs for the infinite state spaces of linear hybrid automata in three guises. The main theoretical limitation of the method is that termination (i.e., finite approximability) is not guaranteed (problem A); the main practical limitation of the implementation is its cost, which is largely due to two factors. First, continuous activities correspond to quantifier-elimination steps on real-valued linear expressions (problem B); second, if expressions are transformed into disjunctive normal form for further manipulation, then the number of disjuncts grows rapidly (problem C).

Having identified the three problems, we improve the performance of HyTech by using algorithms that manipulate geometric objects rather than expressions, and by adding various abstract interpretation strategies, both exact and conservative, to guide the fixpoint computation of the target region. To address problem A, we approach the target region simultaneously from several directions, so that success is guaranteed even if only one direction terminates (Section 3). To address problem B, we represent regions as unions of convex polyhedra, so that quantifier elimination is replaced by linear operations on polyhedra. To address problem C, we conservatively approximate the union of several convex polyhedra by a single convex polyhedron. We discuss two abstract operators—convex-hull approximation and extrapolation (Section 4). Each approximation can be iteratively refined to achieve the desired precision (Section 5).

We wish to point out that the abstract operators and the iterative approximation process are well-known *abstract interpretation* techniques [CC77, Cou81] applied to a new domain—that of linear hybrid automata. The convex-hull operator [CH78] and the widening operator [CC77, CC92], which is similar to our extrapolation operator, are dynamic abstract operators used for convergence acceleration [CC92]. The idea of two-way iteratively refining approximations is also due to Patrick and Radhia Cousot [CC92], and it has been used for the analysis of real-time systems [WTD94]. Finally, we acknowledge the work of Nicolas Halbwachs, who has applied the convex-hull and widening operations to the analysis of synchronous programs [Hal93] and linear hybrid automata [HRP94]. The current implementation of HyTech makes use of Halbwachs's polyhedron-manipulation library. We suggest a new extrapolation operator instead of Halbwachs's widening operator, which is sometimes too coarse for our purposes (on the other hand, widening, unlike extrapolation, guarantees the convergence of iterative application). We feel that the convex-hull and extrapolation operators are particularly suited for the analysis of the *continuous* and *linear* state spaces, because a polyhedron more naturally represents all of its interior points rather than just the grid of interior integer points, and because the linear regions are closed under both approximation operations.

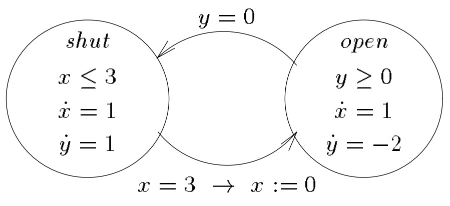


**Fig. 1.** The water-tank automaton

## 2 Linear Hybrid Automata

Linear hybrid automata are formally defined in [HH95]; here we give only an informal review of the definition. A *linear hybrid automaton* $A$ consists of a finite set $X$ of $m$ real-valued variables and a labeled graph $(V, E)$. The edges in $E$ represent discrete transitions and are labeled with guarded commands on $X$; the vertices in $V$ represent continuous activities and are labeled with constraints on the variables in $X$ and their first derivatives. A *state* $(v, s)$ of the hybrid automaton $A$ consists of a control location $v \in V$ (a vertex) and a data state $s \in \mathbb{R}^m$ (a valuation for the variables in $X$). A *data region* is a finite union of convex polyhedra in $\mathbb{R}^m$. A *region* $S = \bigcup_{v \in V}(v, S_v)$ is a collection of data regions $S_v \subseteq \mathbb{R}^m$, one for each control location $v \in V$.

The state $(v, s)$ of the hybrid automaton $A$ can change in two ways: (1) [time step] by a continuous activity that changes the data state $s$ along a smooth function in accordance with the label of the control location $v$; or (2) [transition step] by an instantaneous discrete transition that changes both the control location $v$ and the data state $s$ according to the label of an edge with the source location $v$. The state $\sigma$ can reach the state $\sigma'$ in a *single step*, denoted $\sigma \Rightarrow \sigma'$, if $\sigma$ can evolve into $\sigma'$ by a time step followed by a transition step. The hybrid automaton $A$, then, determines a transition system $(\Sigma, \Rightarrow)$ with the infinite state space $\Sigma = V \times \mathbb{R}^m$ and the transition relation $\Rightarrow \subseteq \Sigma^2$. A *trajectory* of $A$ is a finite path in $(\Sigma, \Rightarrow)$.

The *weakest precondition* of the region $S$, denoted *pre*$(S)$, is the set of all states $\sigma$ such that for some state $\sigma'$ in $S$, $\sigma \Rightarrow \sigma'$. The *strongest postcondition* of $S$, denoted *post*$(S)$, is the set of all states $\sigma$ such that for some state $\sigma'$ in $S$, $\sigma' \Rightarrow \sigma$. Both *pre*$(S)$ and *post*$(S)$ are again regions [AHH93]. In the following, it suffices to assume we are given, in place of the linear hybrid automaton $A$, the location space $V$, the dimension $m$, and algorithms for computing the functions *pre* and *post* on regions.

**Example: water tank**

The linear hybrid automaton of Figure 1 models a water-level controller that opens and shuts the outflow of a water tank. The automaton has two variables, $x$ and $y$, and two locations, *shut* and *open*. The variable $x$ represents a clock of the water-level controller and the variable $y$ represents the water level in

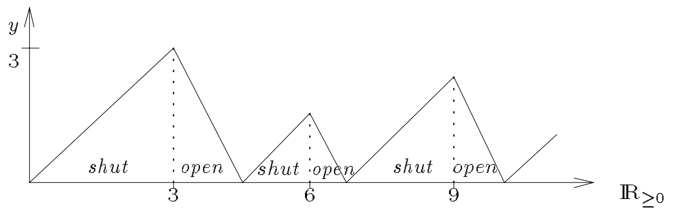


**Fig. 2.** A trace of the water level $y$

the tank. Since the clock $x$ measures time, the first derivative of $x$ is always 1 (i.e., $\dot{x} = 1$). In location *shut*, the outflow of the water tank is shut and the water level increases 1 inch per second ($\dot{y} = 1$); in location *open*, the outflow of the water tank is open and the water level decreases 2 inches per second ($\dot{y} = -2$). The transition from *open* to *shut* (shut the outflow) is taken as soon as the water tank becomes empty: the guard $y = 0$ on the transition ensures that the transition may be taken only when the water level is 0; the constraint $y \geq 0$ on *open* ensures that the transition to *shut* must be taken before the water level becomes negative. The transition from *shut* to *open* (open the outflow) is taken every 3 seconds: the transition is taken whenever $x = 3$, and the transition restarts the clock $x$ at 0. If the automaton is started from the state $(shut, x = y = 0)$, then Figure 2 shows how the water level $y$ changes as a piecewise-linear function of time.

### The symbolic model checker HyTech

In this paper, we study the performance of HyTech, a fully automatic verifier for linear hybrid automata [HH95], on reachability problems. While the original implementation of HyTech represented each data region by boolean combinations of linear inequalities, the current version of HyTech represents each data region by a set of convex polyhedra. In the current version of HyTech, the user interface and the main program are implemented in Mathematica, which calls primitive region-manipulation procedures written in C++. These region-manipulation procedures call on a polyhedron-manipulation library [Hal93].

## 3 Forward versus Backward Reachability Analysis

The *reachability problem* for a hybrid automaton $A$ asks, given an *initial region* $S$ and a *final region* $T$, if there is a trajectory of $A$ that leads from a state in $S$ to a state in $T$. If $T$ represents the set of "unsafe" states specified by a safety requirement, then the safety requirement can be verified by reachability analysis.

There are two possible approaches for solving the reachability problem by symbolic model checking [ACHH93]: we may compute the target region $post^*(S)$ of states that can be reached from the initial region $S$ and check if $post^*(S) \cap T = \emptyset$ (forward reachability analysis), or we may compute the target region $pre^*(T)$ of states from which the final region $T$ can be reached and check if

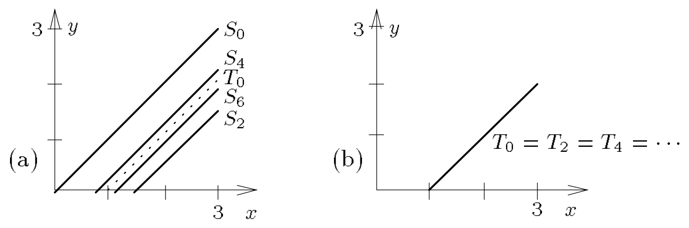


**Fig. 3.** Exact forward and backward analysis

$pre^*(T) \cap S = \emptyset$ (backward reachability analysis). For any given reachability problem, one approach may perform better than the other approach; indeed, it may be that one approach terminates and the other does not.

Recall, for example, the water-tank automaton of Figure 1. We wish to check if the final region $T = (\mathit{shut}, 1 \leq x \leq 3 \wedge x = y + 1)$ can be reached from the initial region $S = (\mathit{shut}, x = y \wedge x \leq 3)$. The symbolic model-checking procedure computes the two regions $post^*(S)$ and $pre^*(T)$ iteratively as the limits of two infinite sequences of regions. Let $S_{i+1} = post(S_i)$ be the region of states that can be reached from $S$ by $i+1$ single steps, and let $T_{i+1} = pre(T_i)$ be the region of states that can reach $T$ by $i+1$ single steps. Then $post^*(S)$ is the limit $\cup_{i \geq 0} S_i$ of the infinite region sequence $\cup_{0 \leq i \leq n} S_n$, for $n \geq 0$, and $pre^*(T)$ is the limit $\cup_{i \geq 0} T_i$ of the infinite region sequence $\cup_{0 \leq i \leq n} T_n$, $n \geq 0$. It is clear that if $post^{n+1}(S) \subseteq \cup_{0 \leq i \leq n} post^i(S)$, then $post^*(S) = \cup_{0 \leq i \leq n} post^i(S)$, and an analogous termination condition holds for the computation of $pre^*(T)$. HyTech checks if $T$ is reachable from $S$ on the fly; that is, if $post^n(S) \cap T \neq \emptyset$ or $pre^n(T) \cap S \neq \emptyset$, then $T$ is reachable from $S$ and the limit computation is aborted. Since for each $i \geq 0$,

$$S_{2i} = (\mathit{shut}, x - y = 1 + \frac{(-1)^{i+1}}{2^i} \wedge x \leq 3)$$

(Figure 3(a)) and

$$S_{2i+1} = (\mathit{open}, 2x + y = 2 + \frac{(-1)^i}{2^i} \wedge y \geq 0),$$

the forward computation of $post^*(S)$ does not terminate within any finite number of iterations. The backward computation, however, converges in a single iteration with the result

$$T_0 = T_2 = (\mathit{shut}, 1 \leq x \leq 3 \wedge x = y + 1)$$

(Figure 3(b)) and

$$T_1 = T_3 = (\mathit{open}, x + 2y = 2).$$

It follows that

$$pre^*(T) = (\mathit{shut}, 1 \leq x \leq 3 \wedge x = y + 1) \cup (\mathit{open}, x + 2y = 2).$$

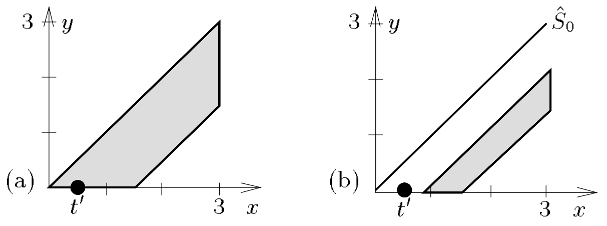


**Fig. 4.** Application of the naive and the refined convex-hull operators

Since $S \cap pre^*(T) = \emptyset$, we conclude that the final region $T$ is not reachable from the initial region $S$. For optimal performance, we have implemented a strategy that dovetails both approaches by computing the alternating sequence $S_0, T_0, S_1, T_1, S_2, \ldots$ of regions.

## 4 Convergence Acceleration through Abstract Operators

Sometimes the exact computation of a target region is prohibitively expensive or does not terminate, independent of the verification strategy (forward vs. backward). In such cases, we approximate the target region and iteratively refine the approximation until sufficient precision is obtained.

We discuss two convergence acceleration operators, convex hull and extrapolation, which can be turned on or off by the user of HyTech. Both operators overapproximate a union of convex polyhedra by a single convex polyhedron and thus reduce the time and space requirements of the verifier; indeed, either operator, or the combination of both operators, may cause the termination of an otherwise infinite computation. If we overapproximate, by $\hat{S}$, the target region $post^*(S)$ of states that are reachable from the initial region $S$ (i.e., $post^*(S) \subseteq \hat{S}$), and $\hat{S}$ contains no final state from $T$, then we can conclude that $T$ is not reachable from $S$; if, on the other hand, $\hat{S}$ does contain a final state, then we cannot reach a valid conclusion and must refine the approximation.

An *abstract operator* $+$ is a binary operator on data regions such that for two data regions $S$ and $S'$, the result $S+S'$ is a convex data region with $S \cup S' \subseteq S+S'$. For the abstract operator $+$, the *approximated postcondition* of the region $S$ is the region $post_+(S) = \cup_v (v, S_v + post(S)_v)$. Then $\cup_{0 \leq i \leq n} post^i(S) = post^n_\cup(S) \subseteq post^n_+(S)$, and we can overapproximate the target region $post^*(S)$ by the limit $\hat{S} = \cup_{i \geq 0} post^i_+(S)$. While we concentrate on the approximate forward analysis in this chapter, analogous observations hold for the approximate backward analysis using *pre*.

### Convex hull

An obviously abstract operator is the *convex-hull operator* $\sqcup$ [Hal93, HRP94], which maps two data regions $S$ and $S'$ to the convex hull of the union $S \cup S'$. Recall once again the water-tank automaton of Figure 1. Now we wish to check if the final state $T' = (\mathit{shut}, x = \frac{1}{2} \wedge y = 0)$ can be reached from the initial region $S = (\mathit{shut}, x = y \wedge x \leq 3)$. Again, the forward computation of the target region $\mathit{post}^*(S)$ does not terminate. This time, however, we can make the forward approach work with the help of the convex-hull operator. Let $\tilde{S}_i = \mathit{post}^i_{\sqcup}(S)$. Then $\tilde{S}_0 = S$,

$$\begin{aligned}
\tilde{S}_1 &= (\mathit{shut}, x - y = 0 \wedge x \leq 3) \cup (\mathit{open}, y + 2x = 3 \wedge y \geq 0),\\
\tilde{S}_2 &= (\mathit{shut}, (x - y = 0 \wedge x \leq 3) \sqcup (x - y = \tfrac{3}{2} \wedge x \leq 3))\\
&\quad \cup (\mathit{open}, y + 2x = 3 \wedge y \geq 0)\\
&= (\mathit{shut}, 0 \leq x - y \leq \tfrac{3}{2} \wedge x \leq 3) \quad (\mathit{Figure}\ 4(a))\\
&\quad \cup (\mathit{open}, y + 2x = 3 \wedge y \geq 0),\\
\tilde{S}_3 &= (\mathit{shut}, 0 \leq x - y \leq \tfrac{3}{2} \wedge x \leq 3) \cup (\mathit{open}, \tfrac{3}{2} \leq y + 2x \leq 3 \wedge y \geq 0),\\
\tilde{S}_4 &= \tilde{S}_3.
\end{aligned}$$

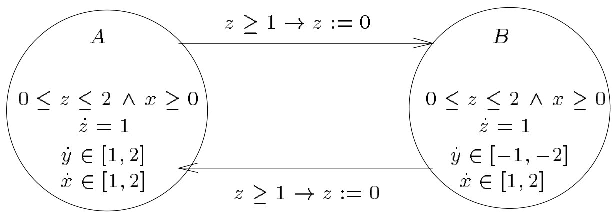
The forward computation terminates with the overapproximation $\tilde{S}_3$ of the target region. Since $\tilde{S}_3$ contains $T'$, however, we cannot conclude that $T'$ is or is not reachable from $S$. Thus we refine our approximation strategy, and apply the convex-hull operator only if the resulting region does not contain any final states. This refinement is sensible for all abstract operators. Formally, we overapproximate the target region $\mathit{post}^*(S)$ by the limit $\hat{S} = \cup_{i \geq 0} \hat{S}_i$, where

$$\hat{S}_{i+1} = \begin{cases} \mathit{post}_{+}(\hat{S}_i) & \text{if } \mathit{post}_{+}(\hat{S}_i) \cap T = \emptyset,\\ \mathit{post}(\hat{S}_i) & \text{otherwise.} \end{cases}$$

For the water-tank example, we obtain $\hat{S}_0 = S$,

$$\begin{aligned}
\hat{S}_1 &= (\mathit{shut}, x - y = 0 \wedge x \leq 3) \cup (\mathit{open}, 2x + y = 3 \wedge y \geq 0),\\
\hat{S}_2 &= (\mathit{shut}, x - y = \tfrac{3}{2} \wedge x \leq 3) \cup (\mathit{open}, 2x + y = 3 \wedge y \geq 0),\\
\hat{S}_3 &= (\mathit{shut}, x - y = \tfrac{3}{2} \wedge x \leq 3)\\
&\quad \cup (\mathit{open}, (2x + y = 3 \wedge y \geq 0) \sqcup (2x + y = \tfrac{3}{2} \wedge y \geq 0))\\
&= (\mathit{shut}, x - y = \tfrac{3}{2} \wedge x \leq 3) \cup (\mathit{open}, \tfrac{3}{2} \leq 2x + y \leq 3 \wedge y \geq 0),\\
\hat{S}_4 &= (\mathit{shut}, (x - y = \tfrac{3}{2} \wedge x \leq 3) \sqcup (\tfrac{3}{4} \leq x - y \leq \tfrac{3}{2} \wedge x \leq 3))\\
&\quad \cup (\mathit{open}, \tfrac{3}{2} \leq 2x + y \leq 3 \wedge y \geq 0)\\
&= (\mathit{shut}, \tfrac{3}{4} \leq x - y \leq \tfrac{3}{2} \wedge x \leq 3) \quad (\mathit{Figure}\ 4(b))\\
&\quad \cup (\mathit{open}, \tfrac{3}{2} \leq 2x + y \leq 3 \wedge y \geq 0),\\
\hat{S}_5 &= \hat{S}_4.
\end{aligned}$$

The forward computaion terminates with the overapproximation $\cup_{0 \leq i \leq 4} \hat{S}_i$ of the target region. Since $\cup_{0 \leq i \leq 4} \hat{S}_i$ does not contain $T'$, we conclude that the final state $T'$ is not reachable from the initial region $S$.



A
B
$z \geq 1 \rightarrow z := 0$
$0 \leq z \leq 2 \wedge x \geq 0$
$\dot{z} = 1$
$\dot{y} \in [1,2]$
$\dot{x} \in [1,2]$
$0 \leq z \leq 2 \wedge x \geq 0$
$\dot{z} = 1$
$\dot{y} \in [-1,-2]$
$\dot{x} \in [1,2]$
$z \geq 1 \rightarrow z := 0$

**Fig. 5.** The robot automaton

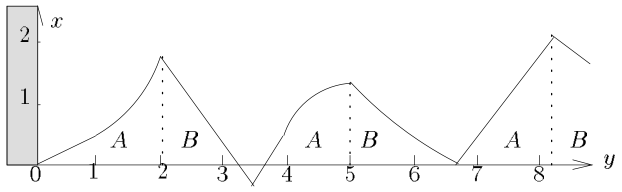


**Fig. 6.** A trajectory of the robot automaton

## Extrapolation

Convex-hull approximation is typically useful for systems with variables whose values are bounded, such as the water-tank automaton. By contrast, if the regions in the sequence $\cup_{0 \leq i \leq n} S_i$, $n \geq 0$, grow without bound, we may want an extrapolation operator that "guesses" an overapproximation of the limit $\cup_{i \geq 0} S_i$. Consider, for example, the hybrid automaton of Figure 5, which models the movement of a robot in the $(x, y)$-plane. The robot can be in one of two modes—heading roughly northeast (location $A$) or heading roughly southeast (location $B$): in mode $A$, the derivatives of the coordinates $x$ and $y$ vary within the closed interval $[1, 2]$; in mode $B$, the derivative of $x$ remains between 1 and 2, while the derivative of $y$ changes its sign and varies between $-1$ and $-2$. The robot changes its mode every 1 to 2 minutes, according to its clock $z$. Moreover, there is a wall at the $x = 0$ line, causing the system invariant $x \geq 0$. Figure 6 shows how the position of the robot in the $(x, y)$-plane may change with time, assuming the robot is started in the initial state $S = (\mathit{shut}, x = y = z = 0)$.

We wish to check if the robot can reach, from $S$, the final position $T = (x = 9 \wedge y = 12)$. At this point, we invite the ambitious reader to prove that $T$ cannot be reached from $S$. Using HyTech, the backward computation of the region $\mathit{pre}^*(T)$ terminates within 74 seconds of CPU time[4] after seven iterations of $\mathit{pre}$, and $S \cap \mathit{pre}^*(T) = \emptyset$. The forward computation of the region $\mathit{post}^*(S)$, by contrast, does not terminate, because the robot keeps zig-zagging to the east and the limit $\cup_{i \geq 0} \mathit{post}^i(S)$ cannot be computed exactly.[5] As convex-hull approx-

[4] All performance figures are given for a SPARC 670MP station.

[5] True, if the target region $T$ is reachable, then it can be reached within a finite number of discrete transitions, and if not, then the invariant $x > 9$ becomes true after a finite number of transitions; extrapolation, however, avoids the ad-hoc "guessing" of

imation does not help with this example, we define an extrapolation operator that approximates the unbounded target region $post^*(S)$; using HyTech, the forward computation with extrapolation, then, terminates within 8 seconds of CPU time.

We first give an intuitive motivation for the extrapolation operator. Suppose that the iterative computation of the target region leads, for a given control state, to the data region (polyhedron) $S$ and, in the subsequent iteration, to the polyhedron $S'$. Suppose, furthermore, that there is a function $f$ such that $f$ maps $S$ to $S'$, mapping extreme points to extreme points. A reasonable guess for the target region, then, would be $f^\infty(S)$.

To formally define an operator $\propto$ that extrapolates $S$ and $f(S)$ to $f^\infty(S)$, we need to review some facts about convex polyhedra [NW88]. By Minkowski's theorem, every nonempty convex polyhedron $R$ in $\mathbb{R}^n$ has a unique (within scalar multiplication) representation by its extreme points $\{x_i \mid i \in I\}$ and extreme rays $\{r_j \mid j \in J\}$ such that

$$S \ = \ \{x \in \mathbb{R}^n \mid x \ = \ \sum_{i \in I} \lambda_i x_i + \sum_{j \in J} \mu_j r_j, \text{ where } \lambda_i, \mu_j \in \mathbb{R}_{\geq 0} \text{ and } \sum_{i \in I} \lambda_i = 1\}.$$

Notice that if the point $x$ and the ray $r$ are in $S$, then so are all points of the form $x + \lambda r$, for $\lambda \in \mathbb{R}_{\geq 0}$. The nonempty convex polyhedron $S$ has *dimension* $k$, denoted $dim(S) = k$, if there are $k+1$ points $x_1, x_2, \ldots, x_{k+1}$ in $S$ such that the $k$ differences $x_2 - x_1, x_3 - x_1, \ldots, x_{k+1} - x_1$ are linearly independent. Notice that if $dim(S) = 0$, then $S$ is a point, and if $dim(S) = 1$, then $S$ is a straight line, a ray, or a line segment. The set $F = \{x \in S \mid ax = b\}$, for two vectors $a$ and $b$, is a *face* of $S$ if every point $x$ in $S$ satisfies the inequality $ax \leq b$. Notice that every face of $S$ is also a convex polyhedron. The face $F$ is a *facet* of $S$ if $dim(F) = dim(S) - 1$. The set $F$ is a 1-dimensional face of $S$ iff $F$ is the intersection of $dim(S) - 1$ facets of $S$ (Criterion (†)).

Now consider two polyhedra $S$ and $S'$. We first compute the convex hull $S \sqcup S'$ of $S \cup S'$. All extreme points and extreme rays of the convex hull $S \sqcup S'$ are extreme points and extreme rays of $S$ or $S'$. Let $\{x_k \mid k \in I\}$ be the extreme points of $S \sqcup S'$ and $S$, let $\{x_l \mid l \in I'\}$ be the remaining extreme points of $S \sqcup S'$, and let $\{r_j \mid j \in J\}$ be the extreme rays of $S \sqcup S'$. Suppose that $k \in I$ and $l \in I'$, and $x_k$ and $x_l$ are the endpoints of a 1-dimensional face of $R \sqcup S'$, which can be checked by Criterion (†). Then it is possible that $f(x_k) = x_l$ for the imaginary function $f$ from $S$ to $S'$. Moreover, if $f$ is applied infinitely many times, then all points of the form $x_k + \lambda(x_l - x_k)$, for $\lambda \in \mathbb{R}_{\geq 0}$, are included in $f^\infty(R)$. We therefore add the ray $x_l - x_k$ into the generators of $S \propto S'$: the *extrapolation operator* $\propto$ maps the two polyhedra $S$ and $S'$ to $S \propto S' \ =$

$$\{x \in \mathbb{R}_n \mid x \ = \ \sum_{i \in I \cup I'} \lambda_i x_i + \sum_{j \in J} \mu_j r_j + \sum_{(k,l) \in K} \mu_{k,l}(x_l - x_k),$$

$$\text{where } \lambda_i, \mu_j, \mu_{k,l} \in \mathbb{R}_{\geq 0} \text{ and } \sum_{i \in I \cup I'} \lambda_i = 1\},$$

suitable invariants.

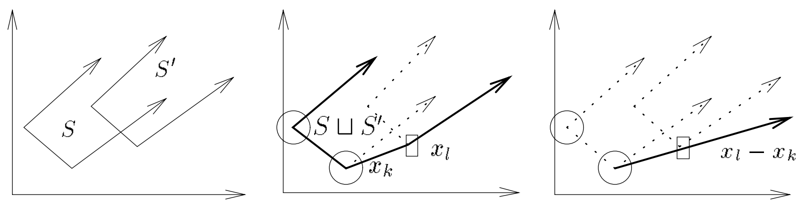


**Fig. 7.** Extrapolation operator

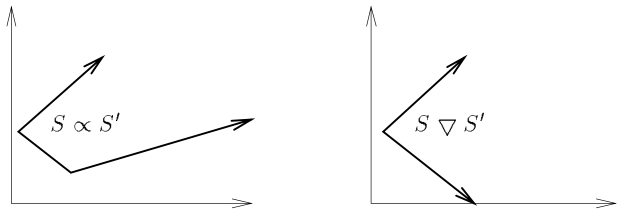


**Fig. 8.** Results of the extrapolation operator and the widening operator

where $(k, l) \in K$ iff $k \in I$ and $l \in I'$ and both $x_k$ and $x_l$ are in the intersection of $dim(R \sqcup S') - 1$ facets of $S \sqcup S'$. Notice that $\propto$ is an abstract operator, that $\propto$ is not symmetric (i.e., $S \propto S'$ and $S' \propto S$ may be different), and that $S \sqcup S' \subseteq S \propto S'$.

Figure 7 shows the application of the extrapolation operator to two convex 2-dimensional polyhedra $S$ and $S'$. The result of the extrapolation $S \propto S'$ is shown as the left figure in Figure 8. As a comparison, the right figure in Figure 8 shows the result of applying the widening operator $\bigtriangledown$ [Hal93, HRP94] to the same regions.

## 5 Two-way Iterative Approximation

Suppose that an approximate forward reachability analysis is inconclusive; that is, the overapproximation $\hat{S}$ of the target region $post^*(S)$ contains some final states from $T$. If the intersection $\hat{S} \cap T$ is a proper subset of $T$, then we have nonetheless obtained new information—namely, that the states in $T - \hat{S}$ are not reachable from $S$—and we may proceed computing backward the new target region $pre^*(T \cap \hat{S})$, or an overapproximation $\hat{T}$ of $pre^*(T \cap \hat{S})$. If $\hat{T} \cap S = \emptyset$, then $T$ is not reachable from $S$; if, on the other hand, $\hat{T}$ does contain some initial states from $S$, then we may continue the two-way iterative approximation process, this time computing forward from $\hat{T} \cap S$.

The two-way iterative approximation strategy is illustrated in Figure 9 and Figure 10, assuming a 2-dimensional state space and convex-hull approximation. The two shaded boxes in the lower left corner of the state space represent the initial region $S$; the two shaded boxes in the upper right corner represent the final region $T$. The derivatives of both variables $x$ and $y$ are 1. While an exact forward or backward analysis would show that $T$ cannot be reached from $S$, both

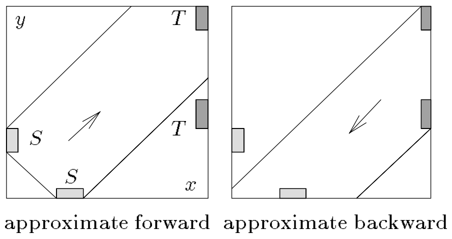


**Fig. 9.** One-way approximative analysis

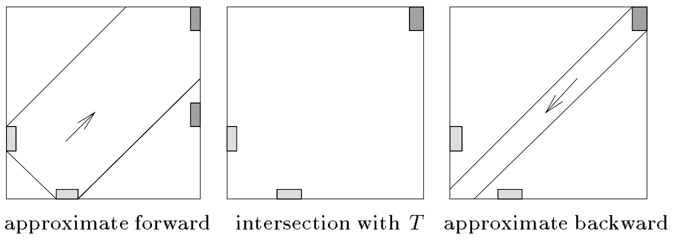


**Fig. 10.** Two-way iterative approximative analysis

approximate forward analysis and approximate backward analysis are inconclusive (Figure 9). An approximate forward analysis followed by an approximate backward analysis is successful (Figure 10).

We now apply the two-way iterative approximation strategy to analyze the bouncing-ball automaton of Figure 11. The variable $x$ represents the horizontal distance of the ball from a reference point, the variable $y$ represents the distance of the ball from the floor, and the variable $z$ represents the "energy" of the ball. Suppose that the ball is dropped at the position $x = 14 \wedge y = 4$ in the direction of the reference point; that is, $S = (\mathit{down}, x = 14 \wedge y = 4 \wedge z = 0)$. While the ball is falling (location *down*), its energy is increasing ($\dot{z} = 1$); when the ball hits the floor ($y = 0$), it loses half of its energy and bounces back up (location *up*); on the way up, the energy decreases ($\dot{z} = -1$) until it becomes 0 and the ball starts to fall again. The trajectory of the ball is shown in Figure 12.

We wish to prove that the ball never reaches the region $T = (x \leq 2 \wedge y = 0)$ of the floor. First notice that the exact forward computation of the target region $\mathit{post}^*(S)$ does not terminate, and convex-hull approximation is of no help. If we use extrapolation, then we obtain the overapproximation $\hat{S}$ of the target region $\mathit{post}^*(S)$, and $\hat{S}$ contains the final states $\hat{S} \cap T = (x = 2 \wedge y = 0 \wedge z = 0)$. Since $\mathit{pre}(\hat{S} \cap T) = \hat{S} \cap T$, a second, backward, pass terminates immediately without reaching the initial state $S$. HyTech requires 12 seconds of CPU time for both passes. (Notice that if we were to begin with a backward pass, attempting to compute the target region $\mathit{pre}^*(T)$, neither the exact computation, nor the approximate computation with convex-hull approximation, nor the approximate computation with extrapolation, would terminate.)

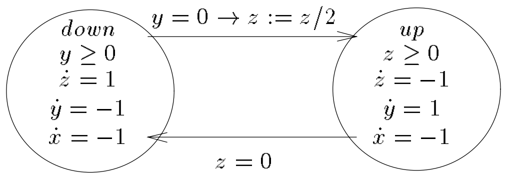


**Fig. 11.** The bouncing-ball automaton

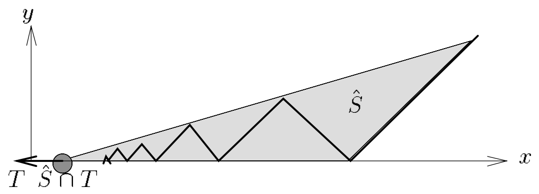


**Fig. 12.** A trajectory of the bouncing-ball automaton

## References

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

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 209–229. Springer-Verlag, 1993.

[AHH93] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. In *Proceedings of the 14th Annual Real-time Systems Symposium*, pages 2–11. IEEE Computer Society Press, 1993.

[BCM$^+$92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: $10^{20}$ states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press.

[CC92] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2/3):103–179, 1992.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the Fifth Annual Symposium on Principles of Programming Languages.* ACM Press, 1978.

[Cou81] P. Cousot. Semantics fundations of program analysis. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, pages 303–342. Prentice-Hall, 1981.

[Hal93] N. Halbwachs. Delay analysis in synchronous programs. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 333–346. Springer-Verlag, 1993.

[HH95] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell Hybrid Technology Tool. This volume, 1995.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[HRP94] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. In B. LeCharlier, editor, *International Symposium on Static Analysis, SAS'94*, Lecture Notes in Computer Science 864, Namur (belgium), September 1994. Springer-Verlag.

[NW88] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[WTD94] Howard Wong-Toi and David L. Dill. Approximations for verifying timing properties. In Teo Rus and Charles Rattray, editors, *Theories and Experiences for Real-Time System Development (Proceedings First AMAST Workshop on Real-Time System Development)*, chapter 7. World Scientific Publishing, 1994.