Formal Techniques for Analyzing Hybrid Systems

Ashish Tiwari

Techniques for Solving the Model Checking Problem

- Fixpoint over concrete semantics
- Fixpoint over abstract semantics with possibly CEGAR loop
- Certificate-based / Constraint-based

Fixpoint over Concrete Semantics

Temporal operators have a fixpoint semantics

Given a temporal formula ϕ, we find the set $[\phi]$ of all states s.t. all traces starting from this set, satisfy ϕ

Consider the formula $\mathbb{F}\phi$

Assume we have found the set $[\phi]$

Now we need to find $[\mathbb{F}\phi]$

We find $[\mathbb{F}\phi]$ iteratively:

- initially, $\psi = [\phi]$
- add s to ψ if all states s' one-step reachable from s are in ψ
- when ψ changes no more, return ψ

We can now determine if $S \models \mathbb{F}\phi$ by checking if Init $\subseteq [\mathbb{F}\phi]$

Fixpoint over Concrete Semantics

Constructing the set $[\mathbb{G}\phi]$

We iteratively build set of states ψ from where $\mathbb{G}\phi$ fails to hold.

- initially, $\psi = \neg[\phi]$
- add s to ψ if some state s' one-step reachable from s is in ψ
- when ψ changes no more, return $\neg\psi$

Exercise. Describe a fixpoint procedure for the $\mathbb{U}$ operator

Implementing Fixpoint over Concrete Semantics

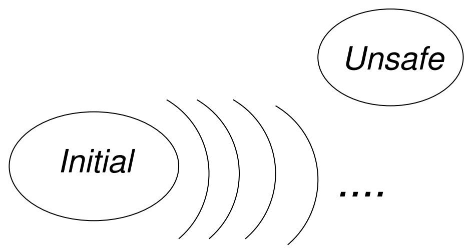
explicit state model checkers: The set of states are represented explicitly

symbolic model checking: The set of states is represented symbolically

In both cases, the exact set, $[\phi]$, is computed

- every state in $[\phi]$ is in the computed representation
- every state in the computed representation is in $[\phi]$

Iterative fixpoint-based Method



- can over-approximate at each step
- bmc
- HyTech, Checkmate, ddt, PhaVer, SpaceEx

Challenge: Computing good quality Pre/Post
symbolic if dynamics easy, and numerical o.w.

Fixpoint over Concrete Semantics

Works for finite-state systems

No guarantee of convergence for infinite-state systems

There can be classes of infinite-state system for which certain representations could be sufficient to capture $[\phi]$

And where fixpoint computation, when performed over this representation, always terminates

Continuous-Time: Fixpoint over Concrete

The dynamics of continuous-time systems are specified using differential equations

The key step in model checking involves computing the one-step successor of a set of states

Given an ODE system, $d\vec{x}/dt = f(x)$, a set Init of initial states, and a time bound T, find (a representation) for the set of states reached from Init in time $[0, T]$ following the given ODE dynamics.

Even if the ODEs are linear, and Init is convex, the set of states reachable in $[0, T]$ may not be convex.

SpaceEx: `spaceex.imag.fr`

Fixpoint over Abstract Semantics

The concrete system may be difficult to analyze

We can then consider abstractions of the system

The goal of abstraction is to get a sufficient, but not necessary, check for $S \models \phi$, which is simpler to decide

The system $S^a = (\mathsf{X}^a, \mathsf{F}^a, I^a)$ is an abstraction of $S^a = (\mathsf{X}, \mathsf{F}, I)$ if there is an abstraction mapping $\alpha : \mathsf{X} \mapsto \mathsf{X}^a$ s.t. whenever $s \to s'$ in S, we have $\alpha(s) \to \alpha(s')$ in S^a.

Exercise. If $PreImage_\alpha(\alpha(\phi)) = \phi$, then
(1) $S^a \models G(\alpha(\phi))$ implies $S \models G(\phi)$
(2) $S^a \models F(\alpha(\phi))$ implies $S \models F(\phi)$

Fixpoint over Abstract Semantics: CEGAR

To solve $S \models \phi$

We can construct S^a, ϕ^a, such that abstraction is lossless on atomic predicates of ϕ

Then, check $S^a \models \phi^a$

If answer is "yes", then we return "yes"

If answer is "no", then we can

- try to determine if the trajectory that falsifies ϕ^a is spurious
- if so, we can try to refine S^a
  (Make it lossless on more predicates)

Abstractions

A canonical way to create an abstraction $(\mathsf{X}^a, \mathsf{F}^a, I^a)$ of $(\mathsf{X}, \mathsf{F}, I)$:

Partition X into subspaces, and each subspace is an abstract state

$\mathsf{X}^a = \{\mathsf{X}_1, \mathsf{X}_2, \ldots\}$ where X is a disjoint union $\bigcup_i \mathsf{X}_i$

Define F^a using the abstract one-step relation:

$\mathsf{X}_i \rightarrow^a \mathsf{X}_j$ if $\exists s \in \mathsf{X}_i : \exists s' \in \mathsf{X}_j : s \rightarrow s'$

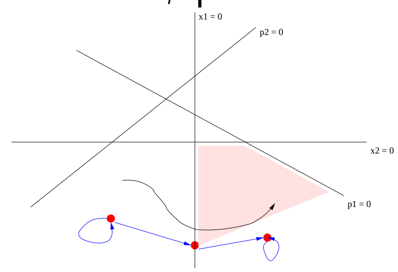
Define I^a: $\mathsf{X}_i \in I^a$ if there is a state $s \in \mathsf{X}_i$ s.t. $s \in I$

Partition s.t. predicates in property are union of abstract spaces

Abstractions

Abstractions can be used on discrete- and continuous-space systems

Consider a system with state space $\Re^2$, partitioned w.r.t signs of x_1, x_2, p_1, p_2:



$\{x_1 = 0, x_2 < 0, p_1 < 0, p2 > 0\} \overset{\#}{\Rightarrow} \{x_1 > 0, x_2 < 0, p_1 < 0, p2 > 0\}$ if

$$\boxed{\exists x_1, x_2 : x_1 = 0 \wedge x_2 < 0 \wedge p_1 < 0 \wedge p2 > 0 \wedge \frac{dx_1}{dt} > 0}$$

Abstraction-based Analysis

Two options:

Construct an abstraction, and then model check it

- HybridSAL approach

Interleave model checking (fixpoint computation) and abstraction computation

- Abstract Interpretation

Flavors of Abstraction

Abstraction: Map concrete system to an abstract system that has no less behaviors

Choices:

- abstract state space: qualitative abstraction
- abstract the dynamics: relational abstraction
- abstract initial set and safe set

Flavor 1: Qualitative Abstraction

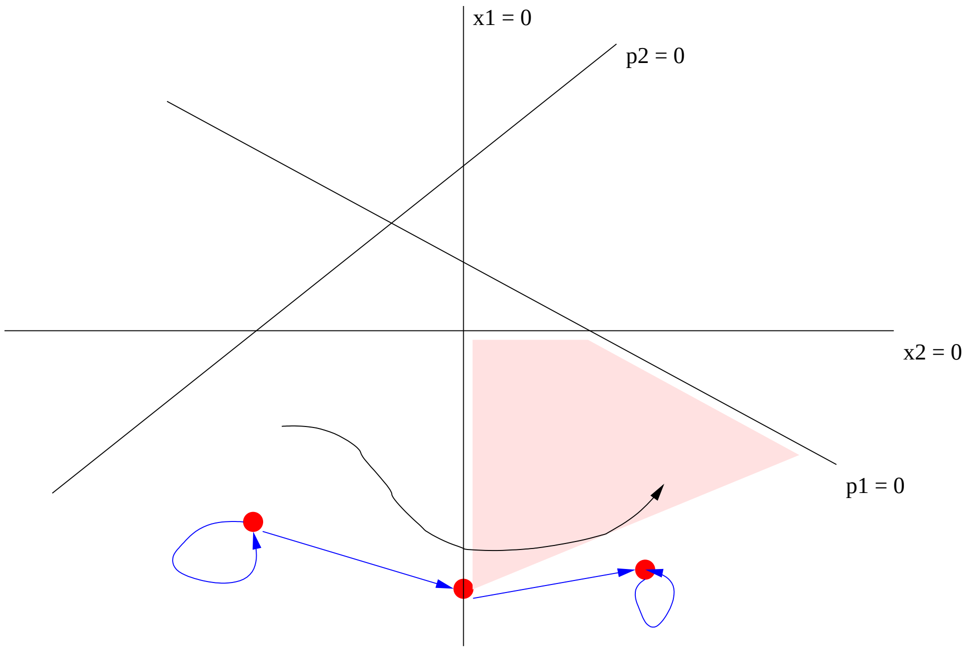
Map concrete state space to abstract state space and lift concrete dynamics to abstract dynamics

Components of qualitative abstractor:

- abstraction mapping:
  value of concrete variables $\mapsto$ value of predicates
- continuous dynamics $\mapsto$ abstract using qualitative reasoning
- discrete dynamics $\mapsto$ abstract using predicate abstraction

Flavor 1: Qualitative Abstraction

Partition concrete state space, e.g. $\Re^2$, w.r.t signs of polynomials x_1, x_2, p_1, and p_2.



There will be an abstract transition from $x_1 = 0 \wedge x_2 < 0 \wedge p_1 < 0 \wedge p_2 > 0$ to $x_1 > 0 \wedge x_2 < 0 \wedge p_1 < 0 \wedge p_2 > 0$ if

$$\boxed{\exists x_1, x_2 : x_1 = 0 \wedge x_2 < 0 \wedge p_1 < 0 \wedge p_2 > 0 \wedge \frac{dx_1}{dt} > 0}$$

Flavor 1: Abstracting Discrete Transitions

Discrete Transition: $(q, \psi(X), q', New(X))$, where

- q, q': modes,
- $\psi(X)$: enabling condition, and
- $New(X)$: assignments to continuous variables.

Abstract Discrete Transition: $((q, \phi_1), (q', \phi_2))$ if the formula

$$\exists X^o, X : \phi_1(X^o) \wedge \psi(X^o) \wedge X = New(X^o) \wedge \phi_2(X)$$

is satisfiable.

Flavor 1: Features of Qualitative Abstraction

Abstract state space $:= 3^P \times \mathbf{Q}$

Correctness The abstractions constructed by the algorithm are sound with respect to the hybrid automata semantics.

Relative Completeness Let ϕ be a QF fmla over X (in $\Re$) that represents the set of reachable states and $P = Poly(\phi)$. Let ψ be the reachable set computed by the algorithm with seed P. If the saturation process terminates, then $\psi = \phi^c$.

Note further that:

- Abstractions can be refined by adding more polynomials,
- Only simple computational steps involved.

Flavor 1: Qualitative Abstraction Example

Thermostat:

$q = \text{off} : \frac{dx}{dt} = -x$

$q = \text{on} : \frac{dx}{dt} = 100 - x$

$g_{12} = x \leq 70$

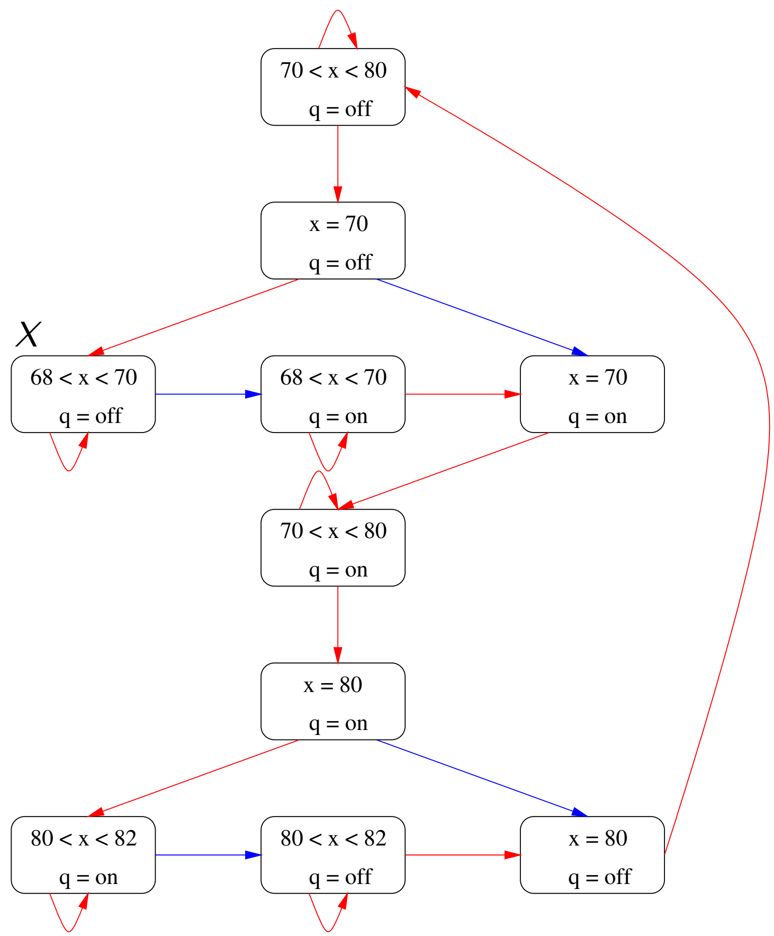
$g_{21} = x \geq 80$

$I(\text{off}) = x > 68$

$I(\text{on}) = x < 82$

partition x by

$0, 68, 70, 80, 82$



Flavor 1: Qualitative Abstraction Issues

- Very coarse
- How to find good predicates ?

Can we improve the quality of abstraction?

Flavor 2: Relational Abstraction

Abstracting the dynamics, not the state space

- creates a discrete infinite-state abstraction
- does not abstract the state-space;
  only the ODE transitions are over-approximated by discrete transitions:
  $\vec{x} \longrightarrow \vec{x}'$ if there is a solution F of the ODE s.t. $F(0) = \vec{x}$ and $F(t) = \vec{x}'$ for some $t \geq 0$
- HybridSAL finds an over-approximation $\longrightarrow$ without finding F
- completely automatic for linear ODEs

Implemented in the HybridSal Relational Abstracter

Flavor 2: Relationalizing Continuous Dynamics

Replace $\frac{d\vec{x}}{dt}$ by a relation that defines how the initial state relates to the final state

$$\begin{aligned} \frac{d\vec{x}}{dt} &= f(\vec{x}) && (1) \\ &\Downarrow \\ R(\vec{x}, \vec{y}) &\text{ if } \vec{y} = F(t), \vec{x} = F(0), \dot{F} = f && (2) \end{aligned}$$

Example:

$$\begin{aligned} \frac{dx}{dt} &= -x && (3) \\ &\Downarrow \\ R(x, y) &\text{ if } (x \leq y < 0) \vee (0 < y \leq x) \vee (x = y = 0) && (4) \end{aligned}$$

Flavor 2: Relational Abstraction Examples

continuous-time continuous-space concrete system	continuous-space discrete-time relational abstraction
$\dot{x} = 1, \dot{y} = 1$	$x' - x = y' - y \quad \wedge \quad y' \geq y$
$\dot{x} = 2, \dot{y} = 3$	$(x' - x)/2 = (y' - y)/3 \quad \wedge \quad y' \geq y$
$\frac{dx}{dt} = -x$	$x \geq x' > 0 \vee x \leq x' < 0 \vee x = x' = 0$
$\frac{dx}{dt} = -x + y$ $\frac{dy}{dt} = -x - y$	$\max(\|x\|, \|y\|) \geq \max(\|x'\|, \|y'\|) \wedge$ $x^2 + y^2 \geq x'^2 + y'^2$
$\frac{d\vec{x}}{dt} = A\vec{x}$	$(c^T\vec{x} \geq c^T\vec{x'} > 0 \ \vee$ $c^T\vec{x} \leq c^T\vec{x'} < 0 \ \vee$ $c^T\vec{x} = c^T\vec{x'} = 0) \quad \wedge \ldots$

Flavor 2: WHY Relational Abstraction

Concept: Analyze hybrid systems by first replacing ODEs by their relational abstraction

Why is this a good idea?

- separation of concerns
  - use knowledge from control/system theory/linear algebra/Lyapunov functions/barriers to construct high-quality relationalizations of ODEs
  - then use verification techniques for infinite-state systems
- accuracy improves as we get closer to decidable classes
  - relationalization is lossless for timed automata, LHAs
  - almost lossless for other decidable classes of CDSs
- good quality abstractions automatically computed for linear ODEs

- generalizes to timed relational abstraction etc.

Flavor 2: Relational Abstraction Challenge

Is it possible to compute relational abstractions?

We do not want to abstract discrete-time transition relations, because model checkers (and static analyzers) can handle them

Is it possible to compute relational abstractions of continuous-time dynamics?

- For linear ODEs, both real and complex left eigenvectors yield high quality relational abstractions
- For nonlinear ODEs, there are generic methods that are not fully automated

Flavor 2: Computing Relational Abstractions

Suppose dynamics are $\frac{d\vec{x}}{dt} = A\vec{x}$

- Compute left eigenvector $\vec{c}^T$ of A

$$\vec{c}^T A = \lambda \vec{c}^T$$

- Note that

$$\frac{d(\vec{c}^T\vec{x})}{dt} = \vec{c}^T\frac{d\vec{x}}{dt} = \vec{c}^T A\vec{x} = \lambda\vec{c}^T\vec{x}$$

- Thus, we can relate the initial value of $c^T\vec{x}$ and its future value $c^T\vec{x}'$ as follows:

$$0 < \vec{c}^T\vec{x}' \leq \vec{c}^T\vec{x} \ \vee \ 0 > \vec{c}^T\vec{x}' \geq \vec{c}^T\vec{x} \ \vee \ 0 = \vec{c}^T\vec{x}' = \vec{c}^T\vec{x}$$

if $\lambda < 0$. And if $\lambda > 0$, then $\vec{x}, \vec{x}'$ swap places.

This idea generalizes to $\frac{d\vec{x}}{dt} = A\vec{x} + \vec{b}$

Flavor 2: Computing Relational Abstractions Example

$$\frac{dx}{dt} = x - 2y$$
$$\frac{dy}{dt} = -2x + y$$

A matrix has 2 real eigenvalues: -1 and 3

Two left eigenvectors: $(1, 1)$ and $(1, -1)$

Relational abstraction:

$$(0 < x' + y' \leq x + y \ \vee \ x + y \leq x' + y' < 0 \ \vee \ x + y = x' + y' = 0) \ \wedge$$
$$(0 > x' - y' \geq x - y \ \vee \ x - y \geq x' - y' > 0 \ \vee \ x - y = x' - y' = 0)$$

Note: left eigenvectors are potential barrier certificates

Computing Relational Abstractions 2

Suppose dynamics are $\frac{d\vec{x}}{dt} = A\vec{x}$

Suppose we have generated relations for all real eigenvalues

Now suppose there is a complex eigenvalue $a + b\imath$

- Find two vectors $\vec{c}^T$ and $\vec{d}^T$ such that
$$\begin{pmatrix} \frac{d\vec{c}^T\vec{x}}{dt} \\ \frac{d\vec{d}^T\vec{x}}{dt} \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} \frac{d\vec{c}^T\vec{x}}{dt} \\ \frac{d\vec{d}^T\vec{x}}{dt} \end{pmatrix}$$
- Thus, the values of $\vec{c}^T\vec{x}$ and $\vec{d}^T\vec{x}$ spiral in (or spiral out) if $a < 0$ (respectively if $a > 0$)
- Hence, we can relate their initial values to their future values
$$(\vec{c}^T\vec{x})^2 + (\vec{d}^T\vec{x})^2 \geq (\vec{c}^T\vec{x}')^2 + (\vec{d}^T\vec{x}')^2$$
if $a < 0$, and the inequalities are reversed if $a > 0$

Computing Relational Abstractions 2: Example

$$\begin{aligned} \frac{dx}{dt} &= -2x + 5y \\ \frac{dy}{dt} &= -2x - 4y \end{aligned}$$

A matrix has 2 complex eigenvalues: $-3 \pm 3\iota$

Two left eigenvectors: $(1+\iota, 2-\iota)$ and its conjugate

Relational abstraction:

$$\begin{aligned} &(0 < (x'-y')^2 + (x'+2y')^2 \leq (x-y)^2 + (x+2y)^2) \;\vee \\ &(x'-y')^2 + (x'+2y')^2 = (x-y)^2 + (x+2y)^2 = 0 \end{aligned}$$

Note: more potential barrier certificates

Flavor 2: Computing Relational Abstractions 3

Suppose dynamics are $\frac{d\vec{x}}{dt} = A\vec{x}$

Now suppose there exists $p(\vec{x})$ and $q(\vec{x})$ s.t.

$$\frac{dp}{dt} = c \qquad \frac{dq}{dt} = d$$

for some constants c, d

- The value of p and q change linearly with time
- A relational abstraction $R_{crate}(\vec{x}, \vec{x'})$ of $\frac{d\vec{x}}{dt} = A\vec{x}$ is:

$$\frac{p' - p}{c} = \frac{q' - q}{d} \geq 0$$

Flavor 2: Computing Relational Abstractions 4

If $R_1(\vec{x}, \vec{x}')$ and $R_2(\vec{x}, \vec{x}')$ are two relational abstractions of the same system, then $R_1(\vec{x}, \vec{x}') \ \wedge \ R_2(\vec{x}, \vec{x}')$ is also a relational abstraction of that system

So, we compute different relational abstractions for the same linear system based on its different (left) eigenvectors

And return the conjunction of those relations as the final relational abstraction

Relational Abstraction of Hybrid Systems

Given a hybrid system, its relational abstraction can be constructed as follows:

- replace continuous dynamics in each mode by its relational abstraction
- keep state space and discrete transitions unchanged

Verify safety property on the relational abstraction

Using infinite bounded model checking and k-induction

Verifying Relational Abstractions

One step of the abstract model can describe

a continuous evolution followed by a discrete transition

$$\vec{x} \rightarrow^{t}_{cont} \vec{y} \rightarrow_{disc} \vec{z}$$

I.e., do not need to consider two contiguous 'continuous' steps

Hence, we can use small depths when performing infinite bounded model checking

Depth 1 is sufficient to verify safety of continuous systems

Flavor 2: Technical Issues

Poor support for nonlinear in SMT solvers (used for Infinite bounded model checking and k-induction)

So, HybridSal provides linear option:

$$x^2 + y^2 \leq x'^2 + y'^2 \quad \mapsto \quad |x| \leq |x'| + |y'| \;\wedge\; |y| \leq |x'| + |y'|$$

Mode invariants not enforced in the relational abstraction

Can create timed RA for sampled data systems, but BMC depth increases

HybridSal provides command-line options that can improve precision

Flavor 3: Aligned Abstraction

Safety verification problem has three components:

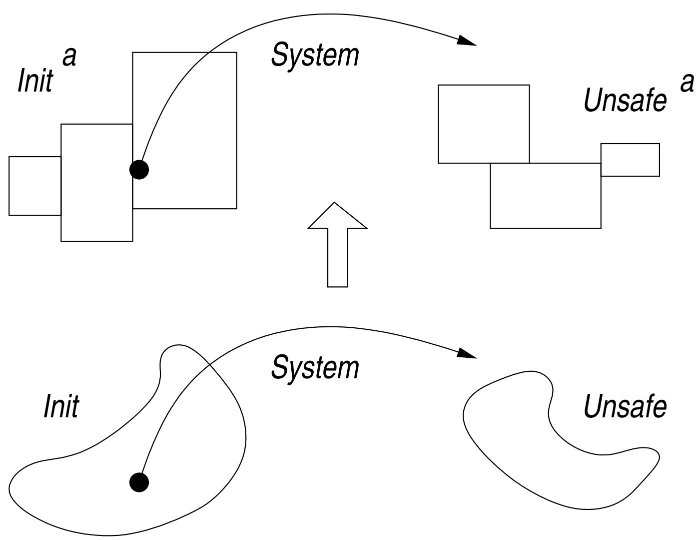
- System, defining state space and dynamics
- Initial states
- Unsafe states

Abstraction-based methods always abstract the system

Can we abstract the initial and unsafe sets ?

Flavor 3: New Abstraction Technique

Replace Init by Init^a where $\text{Init} \subseteq \text{Init}^a$
Replace Unsafe by Unsafe^a where $\text{Unsafe} \subseteq \text{Unsafe}^a$
in a way that the verification problem $(\text{Init}^a, S, \text{Unsafe}^a)$ is easily solved



Flavor 3: Aligned Sets

Consider S: $\frac{dx}{dt} = -x + y - z$, $\frac{dy}{dt} = -x - 3y + z$, $\frac{dz}{dt} = 2$

Consider initial region Init: $x + y \in [2, 4]$, $z = 0$

Consider unsafe region Unsafe: $x + y \geq 1$, $z \geq 2$

The expressions $x + y$ and z are aligned because

$$\begin{aligned} \tfrac{d}{dt}(x+y) &= -x + y - z + (-x - 3y + z) &= -2(x+y) \\ \tfrac{d}{dt}z &= 2 &= 2 \end{aligned}$$

Hence, $z(t) = z(0) + 2t$ and $(x+y)(t) = (x+y)(0)e^{-2t}$

p is aligned if $\dot{p} = \text{constant}$ or $\dot{p} = \lambda p$

Flavor 3: Aligned Safety Verification

If the initial and unsafe sets are specified only using aligned expressions, we call it aligned safety verification problem

The aligned problem is decidable

initial/unsafe set aligned: the expression defining its boundary changes monotonically in a specific way

$\frac{dx}{dt} = -x + y - z, \;\; \frac{dy}{dt} = -x - 3y + z, \;\; \frac{dz}{dt} = 2$
Aligned: Init: $x + y \in [2, 4]$, $z = 0$; Unsafe: $x + y \geq 1$, $z \geq 2$
Not Aligned: Init: $x \in [2, 3], y = 1, z = 0$; Unsafe: $x \geq 1, y \geq 1, z \geq 2$

Flavor 3: Decision Procedure for Aligned Problems

We try to find T: the time when the system reaches an unsafe state
First find all constraints on T
If constraints satisfiable, return unsafe
If constraints unsatisfiable, return safe

Consider the aligned expression z
initially $z = 0$ and in the unsafe region $z \geq 2$
Therefore, $T \geq 1$

Consider the aligned expression $x + y$
initially $x + y \in [2, 4]$ and in the unsafe region $x + y \geq 1$
Therefore, $4e^{-2T} \geq 1$, i.e., $T \leq \ln(4)/2$

The constraint $T \geq 1$ and $T \leq \ln(4)/2$ is unsatisfiable.
Hence, the system is safe

Flavor 3: Correctness For Aligned Instances

Soudness is immediate:
Soundness: If the procedure returns $safe$, then the system is truely safe

Completeness requires a technical condition:
Completeness: If the procedure returns $unsafe$, then the system really is unsafe

Flavor 3: Finding Aligned Directions

Given the system dynamics, can we find the set of aligned directions?

For e.g., how do we find the expressions $(x + y)$ and z given the ODEs

$$\frac{dx}{dt} = -x + y - z, \quad \frac{dy}{dt} = -x - 3y + z, \quad \frac{dz}{dt} = 2$$

We use the eigenstructure of the A matrix

$(x + y) = [1, 1, 0] * [x; y; z]$, and $[1, 1, 0]$ is a left eigenvector of the A matrix corr. to eigenvalue -2

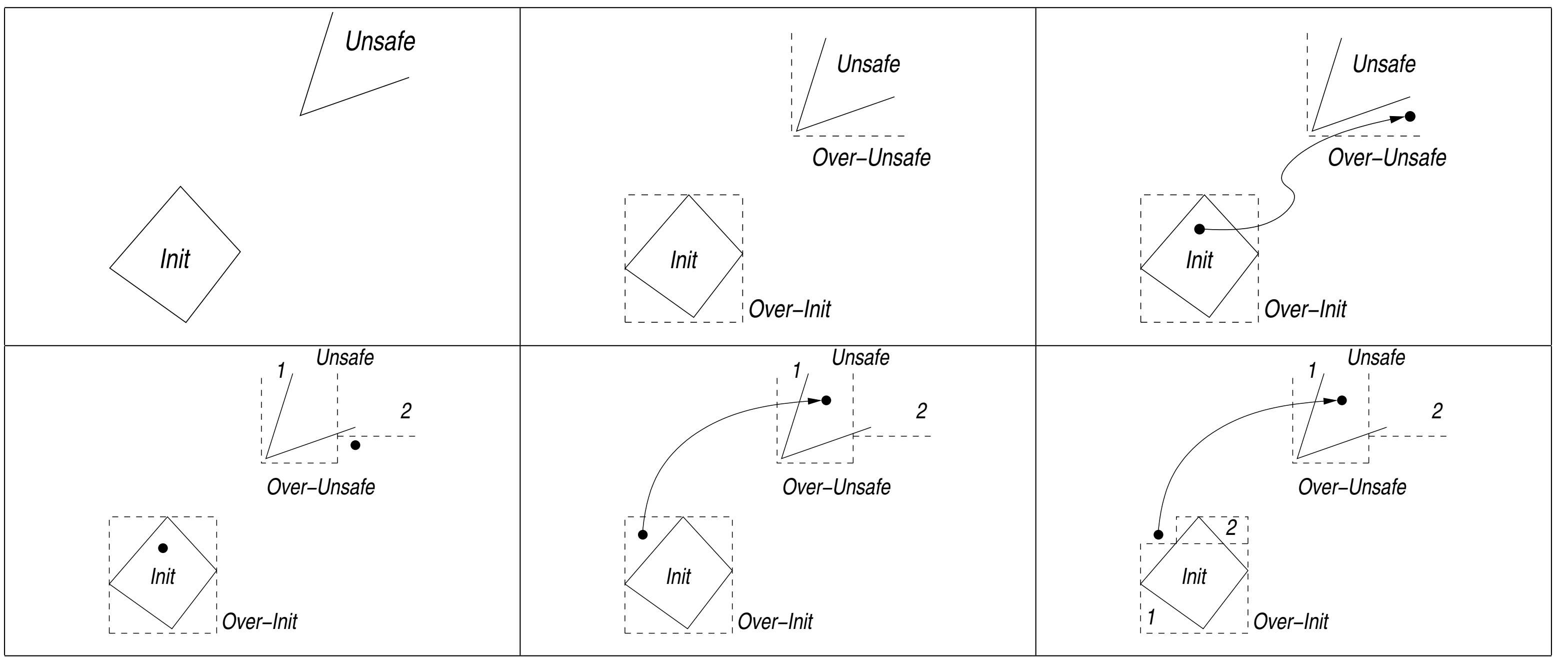
$z = [0, 0, 1] * [x; y; z]$, and $[0, 0, 1]$ is a left eigenvector of the A matrix corr. to eigenvalue 0

Flavor 3: Extending to Unaligned Instances

Counter-Example Guided Abstraction Refinement:

- Find aligned directions
- Abstract to an aligned instance
- Solve the aligned instance
- If safe, then done
- If unsafe, then use the counterexample to refine the aligned abstraction

Flavor 3: CEGAR for Unaligned Safety Verification



Remove regions, not points

The more aligned directions, the better the algorithm performs

Flavor 2: Relational Abstraction Revisited

Can we improve precision of abstraction?

Improving precision of relational abstraction by piecewise linear approximation of exponential and trigonometric functions

Let p be the linear form corr to left eigenvector of A. Let t be the time variable. Let $\lambda > 0$.

$$(\frac{dp}{dt} = \lambda p) \;\Rightarrow\; p' = pe^{\lambda(t'-t)} \;\Rightarrow\; \ln(p') - \ln(p) = \lambda(t' - t)$$

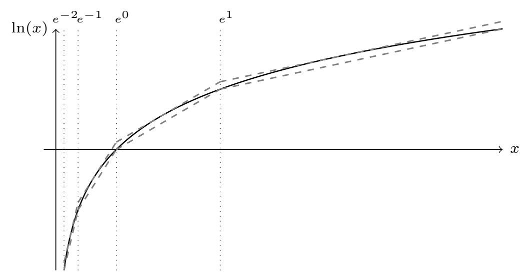
The above "relational abstraction" is nonlinear. We use a piecewise linear lower and upper approx for ln

Flavor 2: Piecewise Linear Approx for ln

Relational abstraction:

$$\ln_{lb}(p') - \ln_{ub}(p) \leq \lambda(t' - t) \leq \ln_{ub}(p') - \ln_{lb}(p)$$

where the lower- and upper-bound approxs are:



Improve precision by increasing number of intervals

Flavor 2: Improving Rel Abs for Complex Case

Recall $(p'^2 + q'^2)^{0.5} = (p^2 + q^2)^{0.5} e^{\lambda(t'-t)}$

Hence we get a relational abstraction:

$$\ln(p'^2 + q'^2)^{0.5} - \ln(p^2 + q^2)^{0.5} = \lambda(t' - t)$$

Again, using the piecewise linear approx. for ln:

$$\begin{aligned}\ln_{ub}(p'^2 + q'^2)^{0.5} - \ln_{lb}(p^2 + q^2)^{0.5} \geq \lambda(t' - t)\\ \ln_{lb}(p'^2 + q'^2)^{0.5} - \ln_{ub}(p^2 + q^2)^{0.5} \leq \lambda(t' - t)\end{aligned}$$

We can additionally also use piecewise linear approximations of the 2-norm function:

$$\max(|x|, |y|) \leq (x^2 + y^2)^{0.5} \leq |x| + |y|$$

This relates amplitude with time

Flavor 2: Relating Phase and Time

Recall:

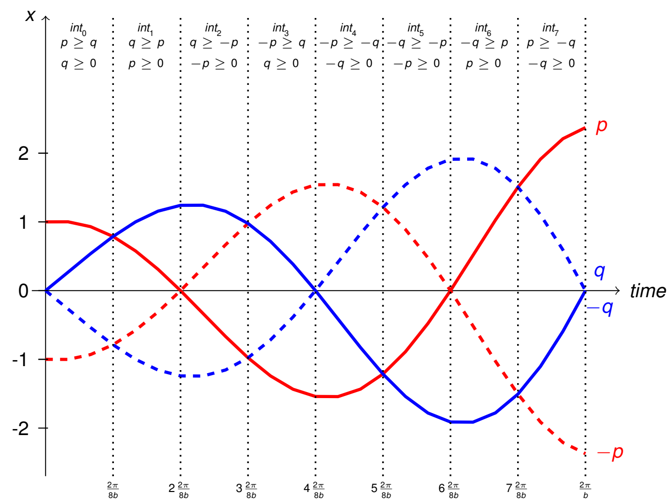
$$
\begin{aligned}
p' &= (p^2 + q^2)^{0.5} e^{a(t'-t)} \cos(b(t'-t) + \tan^{-1}(q/p)) \\
q' &= (p^2 + q^2)^{0.5} e^{a(t'-t)} \sin(b(t'-t) + \tan^{-1}(q/p))
\end{aligned}
$$

If ω denotes the phase, then:

$$b(t'-t) = \omega(p', q') - \omega(p,$$

We need piecewise linear approximations of the ω

Can get bounds based on sign of $p, q, p-q$



Certificate-based Verification

Eliminate iterative fixpoint search

Directly search for proofs

Safety Verification using Inductive Invariants

Consider showing $S \models \mathbb{G}(\mathsf{Safe})$

A discrete-time system always remains inside the set $\mathsf{Safe}(\vec{x})$ of good states if there is an inductive invariant $\mathsf{Inv}(\vec{x})$ such that

$$\begin{aligned} \forall \vec{x} : \mathsf{Init}(\vec{x}) \;&\Rightarrow\; \mathsf{Inv}(\vec{x}) \\ \forall \vec{x}, \vec{x'} : \mathsf{Inv}(\vec{x}) \wedge t(\vec{x}, \vec{x'}) \;&\Rightarrow\; \mathsf{Inv}(\vec{x'}) \\ \forall \vec{x} : \mathsf{Inv}(\vec{x}) \;&\Rightarrow\; \mathsf{Safe}(\vec{x}) \end{aligned}$$

How to find such an Inv?

Safety Verification using Inductive Invariants

Pick a template $T(\vec{a}, \vec{x})$ for the inductive invariant
Generated Constraint:

$$\begin{aligned} \exists \vec{a} : \forall \vec{x}, \vec{x'} : \quad & (\text{Init}(\vec{x}) \Rightarrow T(\vec{a}, \vec{x})) \ \wedge \\ & (T(\vec{a}, \vec{x}) \wedge t(\vec{x}, \vec{x'}) \Rightarrow T(\vec{a}, \vec{x'})) \ \wedge \\ & (T(\vec{a}, \vec{x}) \Rightarrow \text{Safe}(\vec{x})) \end{aligned}$$

Safety Verification: Continuous-Time

A continuous-time system $\dot{\vec{x}} = f(\vec{x})$ always remains inside the set $\mathsf{Safe}(\vec{x})$ of good states if
there is an inductive invariant $T(\vec{a}, \vec{x})$ such that

$$\begin{aligned} \exists \vec{a} : \forall \vec{x} : \quad & (\mathsf{Init}(\vec{x}) \Rightarrow T(\vec{a}, \vec{x})) \ \wedge \\ & (\vec{x} \in \partial T(\vec{a}, \vec{x}) \Rightarrow f(\vec{x}) \in \mathbf{T}\, T(\vec{a}, \vec{x})) \ \wedge \\ & (T(\vec{a}, \vec{x}) \Rightarrow \mathsf{Safe}(\vec{x})) \end{aligned}$$

The middle condition can be formulated for polynomial systems as: $p \geq 0$ is inductive if

$$\forall (\vec{x}) : p(\vec{x}) = 0 \Rightarrow \vec{\nabla} p(\vec{x}) \cdot f(\vec{x}) \geq 0$$

Soundness and Completeness Issues

Sound, but incomplete, rule for safety verification of polynomial CDS S with dynamics $dX/dt = f(X)$ and initial states Init:

$$\begin{array}{ll} (A1) & \text{Init} \Rightarrow p \geq 0 \\ (A2) & p = 0 \Rightarrow L_f(p) \geq 0 \\ (A3) & p \geq 0 \Rightarrow \text{Safe} \\ (A4) & p = 0 \Rightarrow \vec{\nabla} p \neq 0 \\ \hline & \texttt{Reach}(S) \subseteq \text{Safe} \end{array}$$

Relatively complete

Inductiveness Using Lie Derivative

Let $p := x_1^2 + x_2^2 - 0.5$

The set $p \leq 0$ is inductive if

$$
\begin{aligned}
p = 0 \;\; \Rightarrow \;\; & \frac{dp}{dt} < 0 \\
& \vee \frac{dp}{dt} = 0 \wedge \frac{d^2p}{dt^2} < 0 \\
& \vee \frac{dp}{dt} = \frac{d^2p}{dt^2} = 0 \wedge \frac{d^3p}{dt^3} < 0 \\
& \cdots
\end{aligned}
$$

where $\frac{dp}{dt} := \vec{\nabla} p \cdot f$ is Lie derivative of p wrt f.

Several sound checks, but no complete check in general

For special cases, finite complete checks exist

Details

Constraint-based approach for analysis of hybrid systems

Key idea: Bounded search for certificate of a specific form

Constraint-Based Verification:

1. Fix a form (template) for the certificate
   Progress function, $ax^2 + by^2$, for reachability
   Invariant set, $ax^2 + by^2 \geq 0$, for safety
2. Once the form is fixed, existence of a certificate reduces to existence of template variables $a, b, \ldots$:
3. Overall formula takes the form:

$$\exists a, b, \ldots : \forall x, y, \ldots : \cdots$$

4. We solve the $\exists\forall$ formula to find values for $a, b, \ldots$

Certificate-based Verification

Key Observation: Verification = searching for right witness

Property	Witness
Stability	Lyapunov function
Safety	Inductive Invariant
Liveness	Ranking function
Controllability	Controlled Invariant

Certificate-Based Verification

Certificate-based verification reduces the verification problem to an $\exists\forall$ formula.

$$M \models \phi$$
$$\Uparrow$$
$$\exists \Phi : ((M \models \Phi) \;\wedge\; (\Phi \Rightarrow \phi))$$
$$\Uparrow$$
$$\exists \Phi : \forall \vec{x} : \text{ quantifier-free FO formula}$$
$$\Uparrow$$
$$\exists \vec{a} : \forall \vec{x} : \text{ quantifier-free FO formula}$$

The last step performed by choosing a template for Φ

Example: Certificate-Based Safety

Example: $\frac{dx_1}{dt} = -x_1 - x_2 \qquad \frac{dx_2}{dt} = x_1 - x_2$

Problem: If $x_1 \leq 0.5$ and $x_2 \leq 0.5$ initially, prove $G(x_2 \leq 1)$

Let us find a certificate of the form $p \leq 0$ where $p := ax_1^2 + bx_2^2 + c$

We need to solve

$$\begin{aligned}\exists a, b, c : \forall x_1, x_2 : \quad & (p = 0 \Rightarrow \frac{dp}{dt} < 0) \ \wedge \\ & (x_1 \leq 0.5 \wedge x_2 \leq 0.5 \Rightarrow p \leq 0) \ \wedge \\ & (p \leq 0 \Rightarrow x_2 \leq 1)\end{aligned}$$

We get $p := x_1^2 + x_2^2 - 0.5$. Proved.

Certification-based Verification Without $\exists\forall$

A Lyapunov function is a certificate for stability

We can discover Lyapunov functions by solving $\exists\forall$ formulas

But even without solving $\exists\forall$ formulas, we can determine stability of linear systems

Can we find useful invariants without solving $\exists\forall$ formulas ?

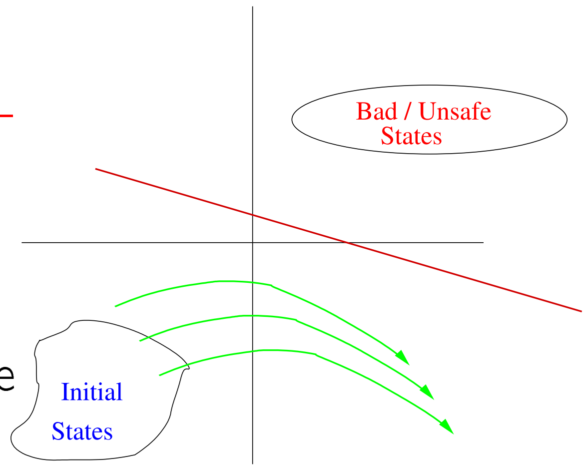
Inductive Sets of Linear Systems

Consider $\frac{d\vec{x}}{dt} = A\vec{x}$

If $\vec{c}$ is a left eigenvector of A, then

$$\vec{c}^{\,T} A = \lambda \vec{c}^{\,T}$$

Let $p := \vec{c}^{\,T}\vec{x}$, we have



$$\frac{dp}{dt} = \frac{d\vec{c}^{\,T}\vec{x}}{dt} = \vec{c}^{\,T}\frac{d\vec{x}}{dt} = \vec{c}^{\,T}A\vec{x} = \lambda\vec{c}^{\,T}\vec{x} = \lambda p$$

Hence, $p \geq 0$ and $p \leq 0$ are inductive sets

The surface $p = 0$ is called a barrier certificate

Inductive sets for linear systems can be obtained by analyzing A

Example: Certificate-based Verification w/o $\exists\forall$

Example. Consider a cruise control:

$$\begin{aligned}\dot{v} &= a \\ \dot{a} &= -4v + 3v_f - 3a + gap \\ g\dot{a}p &= -v + v_f\end{aligned}$$

where v, a is the velocity and acceleration of this car, v_f is the velocity of car in front, and gap is the distance between the two cars.

Prove that the cars will not crash when ACC mode is initiated in given set of states.

Solution: Use inductive invariant corr to the negative real eigenvalue of A.

Example: Certificate-Based Safety

$$\text{Example:} \qquad \frac{dx_1}{dt} = x_2 \qquad \frac{dx_2}{dt} = -x_1$$

Problem: If $x_1 = 1$ and $x_2 = 0$ initially, prove $G(x_1 \leq 1)$

Let us find a certificate of the form $p \leq 0$ where $p := ax_1^2 + bx_2^2 + c$

We need to solve

$$\begin{aligned} \exists a, b, c : \forall x_1, x_2 : \quad & (p = 0 \Rightarrow \frac{dp}{dt} \leq 0) \ \wedge \\ & (x_1 = 1 \wedge x_2 = 0 \Rightarrow p \leq 0) \ \wedge \\ & (p \leq 0 \Rightarrow x_1 \leq 1) \end{aligned}$$

We get $p := x_1^2 + x_2^2 - 1$. Proved.

Example of Certificate-based Verification

Consider the system:

$$
\begin{aligned}
\frac{dx_1}{dt} &= -x_1 - x_2 \\
\frac{dx_2}{dt} &= x_1 - x_2 + x_d
\end{aligned}
$$

Initially: $x_1 = 0, x_2 = 1$

Property: $|x_1| \leq 1$ always

Guess

- Template for witness $W := ax_1^2 + bx_2^2 + c$
- Template for assumption $A := |x_d| < d$

Example Continued

Verification Condition: $\exists a, b, c, d : \forall x_1, x_2, x_d :$

$$\begin{aligned} x_1 = 0 \wedge x_2 = 1 \;&\Rightarrow\; W \leq 0 \\ A \wedge W = 0 \;&\Rightarrow\; \frac{dW}{dt} < 0 \\ W \leq 0 \;&\Rightarrow\; |x_1| \leq 1 \end{aligned}$$

Ask contraint solver for satisfiability of above formula

Solver says: $a = 1, b = 1, c = -1, d = 1$

$$\begin{aligned} x_1 = 0 \wedge x_2 = 1 \;&\Rightarrow\; x_1^2 + x_2^2 - 1 \leq 0 \\ |x_d| < 1 \wedge x_1^2 + x_2^2 - 1 = 0 \;&\Rightarrow\; 2x_1(-x_1 - x_2) + 2x_2(x_1 - x_2 + x_d) < 0 \\ x_1^2 + x_2^2 - 1 \leq 0 \;&\Rightarrow\; |x_1| \leq 1 \end{aligned}$$

This proves that $|x_1| \leq 1$ always.

Notes

- Constraint-based approach can also be used for synthesis
  - E.g. synthesizing the guards for when the robot should switch from one mode to another
- HybridSAL currently does not support:
  - Probabilistic Extension
  - Composition
  - Constraint-based approach