Delta-Complete Reachability Analysis

Sicun(Sean) Gao

(Joint work with Ed Clarke and Jeremy Avigad)

Carnegie Mellon University

Apr 22, 2011

Hybrid Systems

$\mathcal{H} = \langle X, Q, \mathit{Init}, \mathit{Flow}, \mathit{Jump}, \mathit{Inv} \rangle$

- $X \subseteq \mathbb{R}^k$: state space
- Q: a finite set of modes
- $\mathit{Init} \subseteq Q \times X$: initial configurations
- $\mathit{Flow} :\subseteq Q \times X \to TX$: continuous flows
- $\mathit{Jump} :\subseteq Q \times X \to 2^{Q \times X}$: discrete jumps
- $\mathit{Inv} \subseteq Q \times X$: invariants in each mode

Given $\mathit{Unsafe} \subseteq \mathbb{R}^k \times Q$, $[\![\mathcal{H}]\!] \cap \mathit{Unsafe} = \emptyset$?

Example

Example (Transmission Controller)

- $X = \mathbb{R}^3$ (v: Speed, Th: Throttle, Fr: Friction)
- $Q = \{q_1, q_2, q_3\}$ (Gears)
- $Init = (q_1, Th = 0.2 \wedge v = 0)$
- $Inv_{q_1} : 0 \leq v \leq 30, Inv_{q_2} : 25 \leq v \leq 50, Inv_{q_3} : 45 \leq v \leq 70.$
- $Flow_{q_i} : \dfrac{dv}{dt} = c_i(a_i Th - b_i Fr) \wedge \dfrac{dFr}{dt} = e_i v^2.$
- $Jump_{q_1,q_2} : (v \geq 20 \wedge Th > 0.6 \wedge v' = v \wedge Th' = Th)$, etc.

Is $(q_2, Th = 0.1 \wedge v < 30)$ reachable?

Hybrid System Verification is Hard.

- Although there are successful examples, most of the practical systems can not be handled.
- Main Approaches:
  - Geometric Methods
    1. Over-estimate $[\![H]\!]$ up to some time bound t.
    2. Check if $[\![H]\!]^{<t} \cap \mathit{Unsafe} = \emptyset$.
  - Proof-theoretic Methods
    1. Show that $\Phi(\mathcal{H}) \vdash \neg \mathit{Unsafe}$ is derivable syntactically in a sound axiomatic system.

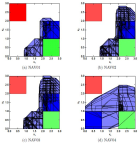
Geometric Methods

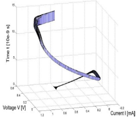
Pros:

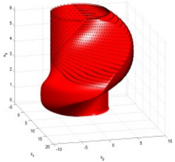
- Computations can be made visible.
- Very helpful for the general understanding of behavior.

Cons:

- High complexity; error control is hard.
- Hard to handle complex dynamics or high dimensions.
- Hard to handle logical operations.







Proof-theoretic Approaches

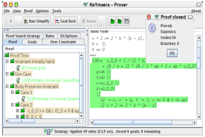
Intro
Encoding
Robustness
Solving
Correctness
End

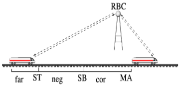
Pros:

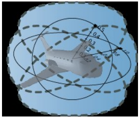
- Highly complex systems/properties.
- Reliable answers.
- No bounds on variables.

Cons:

- Not for debugging.
- Finding invariants needs much human insight.
- Underlying decision procedures can be hard to scale.







Stepping Back

What made traditional model checking scale?

- Encode verification problems into logic formulas.
  - View $\langle X, \rightarrow \rangle$ as a logical structure.
  - Encode properties of interest as a temporal/propositional formula φ.
- Check *satisfiability* of formulas using highly efficient solvers.
  - $\langle X, \rightarrow \rangle \models \varphi$?
  - Use BDD/SAT/SMT solvers to find a model of φ.

Comparison: debugging information, flexible; not visible, bounded

Model-theoretic Methods (Discrete Systems)

Let $\mathcal{M}$ denote the transition system $\langle X, \rightarrow \rangle$.

- Bounded Reachability

  $$\mathcal{M} \models \exists \vec{x}_0, ..., \vec{x}_n (\mathit{Init}(\vec{x}_0) \wedge \bigwedge_{i=0}^{n-1} \mathit{Trans}(\vec{x}_i, \vec{x}_{i+1}) \wedge \mathit{Target}(\vec{x}_n))?$$

- Reachable Set Computation

  $$[\![\exists \vec{x}_0, ..., \vec{x}_{n-1} (\mathit{Init}(\vec{x}_0) \wedge \bigwedge_{i=0}^{n-1} \mathit{Trans}(\vec{x}_i, \vec{x}_{i+1}))]\!]^{\mathcal{M}} = ?$$

- Synthesis Problems

  $$[\![\forall \vec{x}_0, ..., \vec{x}_n (\mathit{Init}(\vec{x}_0) \wedge \bigwedge_{i=0}^{n-1} \mathit{Control}(\vec{x}_i, \vec{x}_{i+1}, \vec{u}_i) \wedge \mathit{Target}(\vec{x}_n))]\!]^{\mathcal{M}} = ?$$

Model-theoretic Methods (Continuous Systems)

Logical encoding is not limited to discrete systems.

- Continuous Dynamics: $\dfrac{d\vec{x}(t)}{dt} = \vec{f}(\vec{x}(t), t)$
  - The solution curve:
    $\alpha : \mathbb{R} \to X,\ \alpha(t) = \alpha(0) + \int_0^t \vec{f}(\alpha(s), s) ds.$
  - Define the predicate
    $[\![Flow_f(\vec{x}_0, \vec{x}, t)]\!]^{\mathcal{M}} = \{(\vec{x}_0, \vec{x}, t) : \alpha(0) = \vec{x}_0, \alpha(t) = \vec{x}\}$
- Reachability
  $\mathcal{M} \models \exists \vec{x}_0, t, \vec{x}\ (Init(\vec{x}_0) \wedge Flow_f(\vec{x}, \vec{x}_0, t) \wedge Target(\vec{x}))$?

Model-theoretic Methods (Hybrid Systems)

Combine the discrete and continuous components[1]:

- $Reach^{0}_{q_0 \to q_0}(\vec{x})$:

  $\exists t_0 \exists \vec{x}_0 \ (Inv_{q_0}(\vec{x}_0) \wedge Inv_{q_0}(\vec{x}) \wedge Flow_{q_0}(\vec{x}, \vec{x}_0, t_0))$

- $Reach^{n+1}_{q_0 \to q}(\vec{x})$:

  $$\exists t_{n+1} \exists \vec{x}_{n+1} \exists \vec{x}'_{n+1} \bigvee_{q' \in Q} [Reach^{n}_{q_0 \to q'}(\vec{x}_{n+1}) \wedge Jump_{q' \to q}(\vec{x}_{n+1}, \vec{x}'_{n+1}) \wedge Flow_q(\vec{x}, \vec{x}'_{n+1}, t_{n+1}) \wedge Inv_q(\vec{x}') \wedge Inv_q(\vec{x}'_{n+1})]$$

$\mathcal{H} \models Reach^{n+1}_{q_0 \to q}(\vec{x}) \wedge Unsafe(\vec{x})$?

[1] Assumption: In each location, the flow stays within the invariant before any jump.

Decision Procedures over Reals

Sadly, in general those first-order formulas over $\mathbb{R}$ can never be decided.

- The arithmetic theory $(\times/+)$ is decidable but highly complex (double-exponential, PSPACE).
  - Available solvers: Usually hard to scale to more than 10 variables.
- Handling nontrivial systems will involve (in the *Flow* predicate) $\exp, \sin/\cos$, *ODEs*, ...
  - Wildly undecidable.

Allowing Errors

On the other hand, large systems of real equalities/inequalities/ODEs are routinely solved numerically.

- They are perfect for simulation, but always regarded inappropriate for verification.
  - (Platzer and Clarke, HSCC 2008)
- Is there a way of using them still?

Allowing Errors

Decide

$$\exists \vec{x}. f(\vec{x}) = 0 \wedge g(\vec{x}) = 0.$$

- Symbolically: We need to consider the global algebraic properties of f and g.
- Numerically: We use iterations that only involve local evaluations of f and g (and their derivatives).
- With error bound δ, we'd "numerically" decide:

$$\exists \vec{x}. |f(\vec{x})| < \delta \wedge |g(\vec{x})| < \delta.$$

Robust Formulas

Consider any formula

$$\varphi := \exists^{I_{\vec{x}}} \vec{x}. \bigvee \left(\bigwedge_i f_i(\vec{x}) = 0 \wedge \bigwedge_j g_j(\vec{x}) \neq 0 \right)$$

Define its δ-perturbed form

$$\varphi^{\delta} := \exists^{I_{\vec{x}}} \vec{x}. \bigvee \left(\bigwedge_i f_i(\vec{x}) < \delta \wedge \bigwedge_j g_j(\vec{x}) \geq \delta \right)$$

We say φ is δ-robust iff

$$\varphi \leftrightarrow \varphi^{\delta}.$$

Robust Formulas (Decidability)

Robust formulas have very nice computational properties.

Definition

$\mathbb{R}_{\mathcal{F}} = \langle \mathbb{R}, \mathcal{F}, < \rangle$ where $\mathcal{F}$ is the set of all real-computable functions. (Type-II computability; exp, sin, ODEs...)

Let φ be a robust and bounded sentence (arbitrary quantification):

Theorem

$\mathbb{R}_{\mathcal{F}} \models \varphi$ *is decidable.*

The proof simulates cylindrical decomposition.

Robust Formulas (Complexity)

In particular, if φ is existentially quantified:

Theorem

If $\mathcal{F}|\varphi$ is real-computable in complexity class $\mathcal{C}$, then deciding φ is in $NP^{\mathcal{C}}$.

This means:

Corollary

Deciding robust bounded existential sentences

1. *in $\mathcal{L}_{+,\times,\exp,\sin}$ is NP-complete.*
2. *in $\mathcal{L}_{LipschitzODE}$ is PSPACE-complete.*

Not Just in Theory

We are developing the practical SMT solver `dReal`.

- DPLL(T) + Interval Constraint Propagation.
  - SAT solver handles Boolean skeleton, ICP handles systems of equations (scalable to 10^3 variables) .
- Currently solvable signature: $+/\times, \exp, \sin$
  - (Gao et al. FMCAD2010)
- In progress: (numerically stable) nonlinear ODEs

Interval Constraint Propagation

- Interval Arithmetic + Constraint Solving

Example

Solve $\{x = y, x^2 = y\}$ for $x \in [1,4]$ and $y \in [1,5]$:
$I^x : [1,4] \to [1,\sqrt{5}] \to [1,\sqrt[4]{5}] \to [1,\sqrt[8]{5}] \to [1,\sqrt[16]{5}] \to \cdots \to [\mathbf{1},\mathbf{1}]$
$I^y : [1,5] \to [1,\sqrt{5}] \to [1,\sqrt[4]{5}] \to [1,\sqrt[8]{5}] \to [1,\sqrt[16]{5}] \to \cdots \to [\mathbf{1},\mathbf{1}]$

- ICP routinely handles thousands of variables and highly nonlinear constraints.

Correctness Guarantee (Formula)

For any existential formula φ (robust or nonrobust), with a tunable error bound δ, we know:

1. Solver says "unsat" $\Rightarrow \varphi$ is δ-robustly unsatisfiable.
   - Unsatisfiable under any perturbation up to δ.
2. Solver says "sat" $\Rightarrow \varphi$ may be unsatisfiable, but φ^{δ} is satisfiable.
   - It means we do know that a syntactically-perturbed version of $\varphi(\vec{x})$ is satisfiable.

This is what we call δ-completeness.

Robust Hybrid Systems

Let $\mathcal{H} = \langle X, Q, Init, Flow, Jump, Inv \rangle$.

Similarly, we can define δ-robust hybrid systems:

- $\mathcal{H}^\delta = \langle X, Q, Init^\delta, Flow^\delta, Jump^\delta, Inv^\delta \rangle$
- $\mathcal{H}$ is δ-robust if

$$\mathcal{H} \sim_{\sigma.bisim} \mathcal{H}^\delta$$

Delta-Complete Bounded Model Checking

When model checking $\mathcal{H}$:

$$\varphi : Reach_{\mathcal{H}}^{\leq n} \text{ is unsat} \Leftrightarrow \mathcal{H} \text{ is safe up to } n$$

1. φ is "unsat" $\Rightarrow$ $\mathcal{H}$ is δ-robustly safe.
   - $\mathcal{H}^{\vec{c}}$ is safe under any δ-perturbation $\vec{c}$.
2. φ is "sat" $\Rightarrow$ $\exists$ δ-perturbation $\vec{c}$, $\mathcal{H}^{\vec{c}}$ is unsafe.
   - The solver returns a solution that shows bug.

This is even better than precise solvers!

Delta-Complete Bounded Model Checking

Pros:

- Highly scalable numerical algorithms and SAT solvers
- Possible to scale to complex dynamics and large dimensions
- No accumulation of numerical errors
- Strong robustness check
- Counterexamples

Cons:

- Bounded variables (can be very loose)
- Bounded unwinding depth
- Computations are not visible
- Debugging, not verifying (yet!)

Conclusion

- Standard model checking techniques (from HW/SW) can be used in realistic continuous/hybrid systems, as long as we have the solver.
- For any solver to scale in this domain, numerical methods have to be exploited.
- Surprisingly, numerical methods will give us stronger results.
- We are developing `dReal` and `dReach`.

"Errors are good (if they work for the verification side)."