Approximate decidability and verification of hybrid systems

Jeremy Avigad

Department of Philosophy and
Department of Mathematical Sciences
Carnegie Mellon University

joint work with Sicun Gao and Edmund M. Clarke (and Soonho Kong)

October 2013

Hybrid systems

Hybrid systems combine discrete and analog components:

- Within a state, the system evolves e.g. according to the solution of a differential equation.
- When guards are triggered, the system switches to a new state (may jump / must jump semantics).

Task: show the system cannot reach an unsafe state.

Bounded model checking: the system reaches an unsafe state in at most n steps if

$$\exists \vec{x}_0, \ldots, \vec{x}_n \left(\text{init}(\vec{x}_0) \wedge \bigwedge_{i=0}^{n} \text{trans}(\vec{x}_i, \vec{x}_{i+1}) \wedge \bigvee_{i=0}^{n} \text{unsafe}(\vec{x}_i)\right).$$

Approaches

A symbolic approach: use a decision procedure for real closed fields.

Problems:

- Complexity overwhelms.
- Polynomials may not be expressive enough.
- Undecidability sets in quickly.

How can we integrate numeric approaches?

- Calculations are only approximate.
- We want an exact guarantee.

Approximate decidability

Our solution involves:

- More flexibility: arbitrary computable functions
- A restriction: quantification only over bounded domains
- A compromise: approximate decidability

This provides a general framework for thinking about verification problems.

Outline

- Background and motivation
- Ideas from computable analysis
- δ-decidability
- Complexity
- Implementation

Computable analysis

A real number r is *computable* if there is a computable function $\alpha : \mathbb{N} \to \mathbb{Q}$ such that for every i, $|\alpha(i) - r| < 2^{-i}$.

Call such an α a *name*. So r is computable if it has a computable name.

A function $f : \mathbb{R} \to \mathbb{R}$ is computable if, given a name α for r as input (say, as an oracle), it computes a name for $f(r)$.

Fact: a computable function from $\mathbb{R}$ to $\mathbb{R}$ is continuous.

Computable analysis

Most real numbers arising "in nature" are computable:

$$\pi, e, \gamma, \phi, \ldots$$

Similarly, continuous functions arising in nature are computable:

- polynomials
- trigonometric functions
- exp, log
- absolute value, min, and max
- solutions to ordinary differential equations with Lipschitz-continuous computable functions

Computable analysis

Note that the function

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

is not computable. In other words, we cannot decide $x \geq 0$.

Fix a small δ. We can do the next best thing, that is, decide

- $x \geq 0$
- $x \leq \delta$

Note that there is a "grey area" where either answer is o.k.

(Note also that the procedure cannot be extensional.)

Computable analysis

An important fact:

Proposition

Every computable function f on a compact interval $[a, b]$ has a computable modulus of (uniform) continuity, $M(\varepsilon)$:

$$\forall x, y \in [a, b]\, \forall \varepsilon > 0\, (|x - y| < M(\varepsilon) \to |f(x) - f(y)| < \varepsilon).$$

In fact, one can present a computable function by giving rational approximations on dyadic reals and a modulus of continuity.

δ-decidability

Choose a language with 0, $+$, $-$, $<$, $\leq$, $|\cdot|$, and symbols for *any* computable functions you want.

It is enough to use only $\forall$, $\exists$, $\wedge$, $\vee$, and $\neg$.

We can push negations inwards, and replace

- $s < t$ by $t - s > 0$,
- $s = t$ by $-|s - t| \geq 0$,

and so on.

In short, we can focus on positive combinations of $t > 0$ and $t \geq 0$, and take negation to be an operator on such formulas.

δ-decidability

Let φ be a formula built up from terms $t > 0$, $t \geq 0$ using $\forall$, $\exists$, $\wedge$, and $\vee$. Let $\delta > 0$.

We obtain

- φ^{+} by replacing each $t \geq 0$ by $t > 0$.
- φ^{-} by replacing each $t > 0$ by $t \geq 0$.
- $\varphi^{+\delta}$ by replacing $t \geq 0$ by $t \geq \delta$ and $t > 0$ by $t > \delta$.
- $\varphi^{-\delta}$ by replacing $t \geq 0$ by $t \geq -\delta$ and $t > 0$ by $t > -\delta$.

Call $\varphi^{+\delta}$ the δ-*strengthening* of φ and call $\varphi^{-\delta}$ the δ-*weakening* of φ.

δ-decidability

For $\delta' \geq \delta$ we have

$$\varphi^{+\delta'} \to \varphi^{+\delta} \to \varphi^{+} \to \varphi \to \varphi^{-} \to \varphi^{-\delta} \to \varphi^{-\delta'},$$

and, for example,

$$(\neg\varphi)^{+\delta} = \neg(\varphi^{-\delta}).$$

δ-decidability

Say a formula φ is *bounded* if every quantifier is of the form $\forall x \in [s, t]$ or $\exists x \in [s, t]$.

Theorem

There is an algorithm which, given any bounded formula φ, correctly returns on of the following two answers:

- *φ is true*
- *$\varphi^{+\delta}$ is false.*

Think of the first answer as "the system is safe," and the second as "a small perturbation of the system is unsafe."

Note that there is a grey area where either answer is allowed.

δ-decidability

The idea behind the proof:

- $s \geq 0$ and $t \geq 0$ if and only if $\min(s, t) \geq 0$
- $s \geq 0$ or $t \geq 0$ if and only if $\max(s, t) \geq 0$
- $\forall x \in [a, b]\ t \geq 0$ if and only if $\min_{x \in [a,b]} t \geq 0$
- $\exists x \in [a, b]\ t \geq 0$ if and only if $\max_{x \in [a,b]} t \geq 0$

All of these are computable, and we have already seen that we can decide between

$$t \geq 0 \quad \text{and} \quad t \leq \delta.$$

Related work

Ratschan (2002) considers "robustness" of formulas under perturbations.

Franek, Ratschan, and Zgliczynski (2011) consider a similar notion of "quasi-decidability".

- Algorithm only required to terminate on robust formulas.
- The notion essentially agrees with ours in the Σ_1 case, in the signature they consider.

We add: arbitrary quantifiers, arbitrary computable functions, complexity considerations.

Similar ideas are used in "continuous model theory." We bring in computability and decidability.

Complexity

Ko (1991) considers computational complexity of real-valued functions.

Saying $f : \mathbb{R} \to \mathbb{R}$ is polynomial time computable means one can compute $f(x)$ to within 2^{-n} in time polynomial in n (querying polynomially many bits).

Ko shows that $\max_{x \in [a,b]} f(x) > r$ is a Σ_1^p query, and so on up the polynomial hierarchy.

Solving Lipschitz continuous ODE's, however, can require PSPACE.

Putting it into practice

Criticism: verification is sensitive to formulation (not maintained under equivalence).

This is a feature, not a bug.

Engineer's requirements: find a property that

- ensures safety, and
- comfortably encloses the system.

The second guarantees an answer of "yes."

Implementation

Note that reachability corresponds to the satisfiability of an existential formula, or existential plus "$\forall t$".

In that case, we want either "unsat" (there is no unsafe trace) or that the δ-weakening is "sat."

Ideas:

- Use interval constraint propagation (branching and pruning) to refine approximations as necessary.
- Use SMT search to learn clauses and guide the search in a lazy way.

Implementation

Sicun Gao, Sonhoo Kong, and Ed Clarke have implemented tools *dreal* and *dreach* based on these ideas:

http://dreal.cs.cmu.edu/

The tools use:

- *opensmt* as the SMT engine
- *realpaver* for interval constraint propagation
- *CAPD* for computing interval enclosures of solutions to ODE's

HySAT (Fränzle et al.) and its successor, *iSAT*, also combine numerical methods with an SMT-like framework.

Generalizations and questions

The hard part: finding efficient implementations.

Question: Numerical approximations can degrade quickly. How can we effectively combine numeric and symbolic approaches?

Question: Computable analysis applies to spaces other than the reals, like spaces of functions and spaces of measures. We know how to think of these as limits of approximations, and the same framework applies. Can this be put to good use?