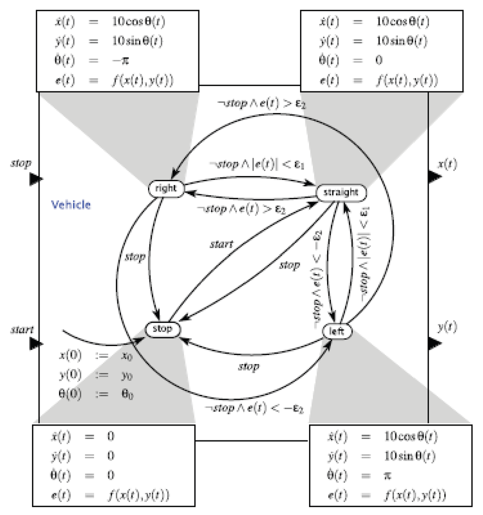
Delta-Complete Methods for Nonlinear Hybrid Systems

Carnegie Mellon University

Sicun Gao, Jeremy Avigad, and Edmund M. Clarke

Introduction

Hybrid systems combine discrete and continuous behaviors.

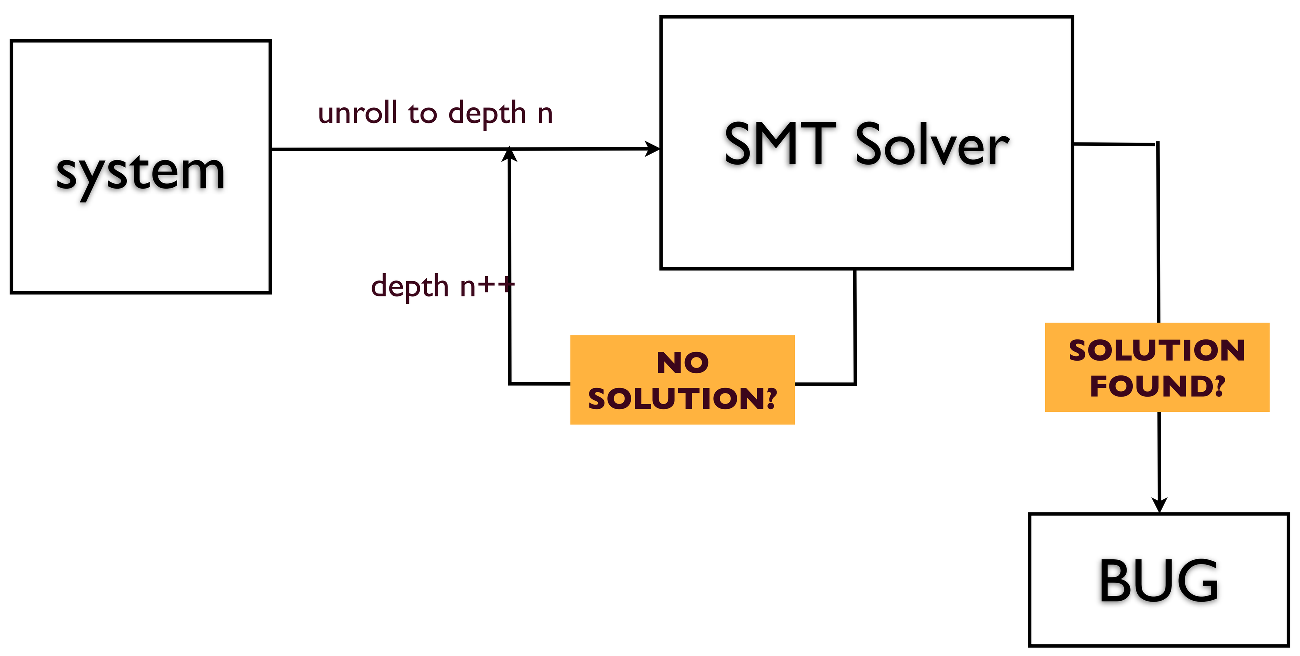


- In realistic hybrid systems, the continuous behaviors are specified by nonlinear real functions and differential equations.
- Current verification tools have great difficulty in handling these functions.

The Call for Efficient SMT Solvers

- Modern verification tools, especially Bounded Model Checkers, rely on efficient solvers for deciding satisfiability of logic formulas. For instance, highly scalable SAT solvers are the key for industrial-strength hardware model checking tools.

- Solving verification problems for nonlinear hybrid systems requires the use of Satisfiability Modulo Theories (SMT) solvers over real numbers.



- However, nonlinear SMT problems over reals are in general undecidable (with transcendental functions), and existing tools that handle basic nonlinear formulas (polynomials) already have very high complexity.

Theory

- On the other hand, nonlinear systems of real equalities and inequalities are routinely solved by highly scalable numerical algorithms.

- However, numerical algorithms always introduce errors that can render formal verification results invalid. [Platzer and Clarke, HSCC07]

- We developed a way of taking numerical errors into account in SMT solvers, and make the use of numerical solvers legitimate.

Type-II Computability

- Type-II Computable Functions: A real-valued function *f(x)* is Type-II computable if given any *a* in its domain, and any positive error bound e, there is an algorithm for computing *y* such that *|f(a)-y|<e*.

- Type-II computability formally describes the functions that are "numerically solvable". Polynomials, exp, continuous ODEs are all Type-II computable.

Perturbations and Robustness

- We define numerical perturbations on SMT formulas over $\mathbb{R}_{\mathcal{F}} = \langle \mathbb{R}, 0, 1, \mathcal{F}, < \rangle$

  - Consider any formula $\varphi := \bigwedge_i (\bigvee_j f_{ij}(\vec{x}) = 0)$.
    - Inequalities are turned into interval bounds on slack variables.
  - A δ-perturbation on φ is a constant vector $\vec{c}$ satisfying $||\vec{c}||_\infty < \delta$, and a δ-perturbed φ is:

$$\varphi^{\vec{c}} := \bigwedge_i (\bigvee_j |f_{ij}(\vec{x})| = c_{ij})$$

  - We say satisfiability of φ is δ-robust (over some bounded $\vec{I}$), if:

    For any δ-perturbation $\vec{c}$, $\exists^{\vec{I}}\vec{x}.\varphi \leftrightarrow \exists^{\vec{I}}\vec{x}.\varphi^{\vec{c}}$.

- Robust Formulas have nice computability and complexity properties.
  - Theorem:

    Satisfiability of robust bounded SMT problems over $\mathbb{R}_{\mathcal{F}}$ is decidable.

    - $\mathcal{F} = \{+, \times, \exp, \sin\}$: **NP**-complete.
    - $\mathcal{F} = \{\text{Lipschitz-continuous ODEs}\}$: **PSPACE**-complete.

Practice

- We can build an SMT solver with the following correctness guarantee, which we call **delta-completeness**:

  - If φ is decided as "unsat", then it is indeed unsatisfiable.
  - If φ is decided as "sat", then:

    Under some δ-perturbation $\vec{c}$, $\varphi^{\vec{c}}$ is satisfiable.

- We use Interval Constraint Propagation (ICP) algorithms to handle real constraints and Interval-based ODE solvers to handle differential equations.

Unrolling Hybrid Systems

- Based on the SMT solver, we can build a Bounded Model Checker for hybrid systems. Formulas are generated in the following way:

  - Continuous Dynamics: $\frac{d\vec{x}(t)}{dt} = \vec{f}(\vec{x}(t), t)$
    - The solution curve: $\alpha : \mathbb{R} \to X,\ \alpha(t) = \alpha(0) + \int_0^t \vec{f}(\alpha(s), s)ds.$
    - Define the predicate (probably no analytic forms) $[\![\mathsf{Flow}_f(\vec{x}_0, t, \vec{x})]\!]^{\mathcal{M}} = \{(\vec{x}_0, t, \vec{x}) : \alpha(0) = \vec{x}_0, \alpha(t) = \vec{x}\}$
  - "$\vec{x}$ is reachable after after 0 discrete jumps":

    $\mathsf{Reach}^0(\vec{x}) := \exists \vec{x}_0, t.\ [\mathsf{Init}(\vec{x}_0) \wedge \mathsf{Flow}(\vec{x}_0, t, \vec{x})]$
  - Inductively, "$\vec{x}$ is reachable after $k+1$ discrete jumps":

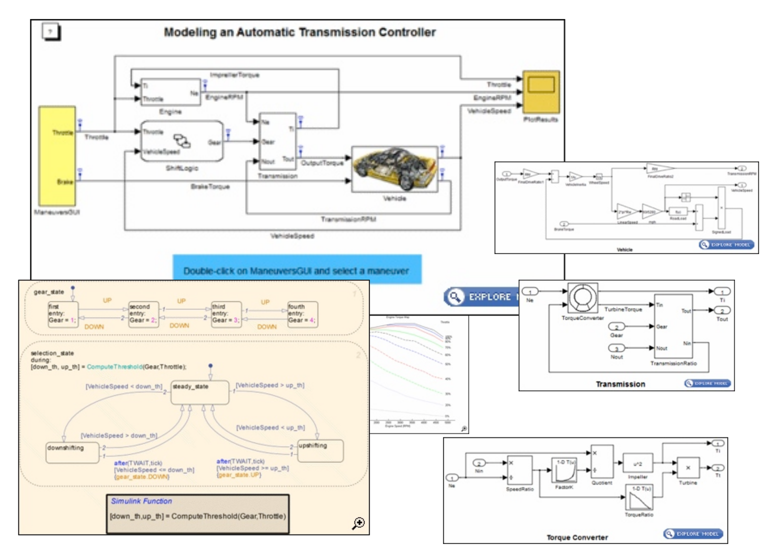
    $\mathsf{Reach}^{k+1}(\vec{x}) := \exists \vec{x}_k, \vec{x}'_k, t.\ [\mathsf{Reach}^k(\vec{x}_k) \wedge \mathsf{Jump}(\vec{x}_k, \vec{x}'_k) \wedge \mathsf{Flow}(\vec{x}'_k, t, \vec{x})]$

- Then using delta-complete solvers, we provide the following correctness guarantee:

- If *Reach* is unsatisfiable, then the system is definitely safe up to n.
- If *Reach* is satisfiable, then there exists a numerical perturbation smaller than the (user-specified) error bound *delta*, such that the system has a bug under that perturbation.

- Note that in this way, we are able to report possible robustness problems in the system! Exact solvers, however, can not provide such support.

Example: Simulink Model of Transmission



Sample Property: *t<200 and Gear = 4 and 70<Speed<80?*

Result: Reachable with the following trace on critical variables:

Speed:
- @0: [1, 1]
- @1: [8.352040521901328629, 8.367597154719149437]
- @2: [29.844748106305949875, 29.856108857390363909]
- @3: [37.106093249135007284, 37.120495559140735509]
- @4: [72.981987126961513468, 72.986794613310777891]

Time:
- @0: [0, 0]
- @1:]6.774259382757060521 4, 6.838379701690715073 4]
- @2: [6.848706083077422235 3, 6.912826402011076787 3]
- @3: [14.752161300744385031, 14.821536317527019833]
- @4: [14.846439470921954 04, 14.929659692431425 1]

Ti:
- @0: [10, 10]
- @1: [63.892425818913153535, 63.947118870255138745]
- @2:]64.321812807985025984, 64.375021420064470 59]
- @3: [3432.148263291640432 7, 3432.225563811623487 7]

Gear:
- @1, @2, @3, @4

Number of Variables: 349 Number of Clauses: 793 Solving Time: 4.5s

Functions Involved: nonlinear polynomials, linear differential equations