Analysis of Infinite State and Hybrid Systems With SAL

John Rushby

Computer Science Laboratory

SRI International

Menlo Park, California, USA

Introduction

- None of this is my work
  - So it'll be a high-level overview
- Infinite bounded model checking
  - Decision procedures and SAT in ICS
  - k-induction
- Hybrid abstraction

Background: Bounded Model Checking

- A useful form of model checking for finite systems is bounded model checking (BMC)
- Is there a counterexample to this property of length k?
- Try $k = 1, 2, \ldots 100 \ldots$ until you find a bug or run out of resources or patience
- Same method generates structural test cases
  - Counterexample to "there's no execution that takes this path"
- We'll see later that it can also be used for verification

Bounded Model Checking (ctd.)

- Given a system specified by initiality predicate I and transition relation T on states S, there is a counterexample of length k to invariant P if there is a sequence of states $s_0, \ldots, s_k$ such that

$$I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \neg P(s_k)$$

- If finite state, then I and T can be encoded a Boolean functions (i.e., circuits) and we then have a propositional satisfiability (SAT) problem
- Needs less tinkering than BDD-based symbolic model checking, can sometimes handle bigger systems, find deeper bugs
- Now widely used in hardware verification
  - Though they generally use several methods in cascade

Infinite BMC

- Suppose T is not a circuit, but software, or a high-level specification
- It'll be defined over reals, integers, arrays, datatypes, with function symbols, constants, equalities, inequalities etc.
- So we need to solve the BMC satisfiability problem

$$I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \neg P(s_k)$$

  over these theories
- Typical example
  - T has 1,770 variables, formula is 4,000 lines of text
  - Want to do BMC to depth 40
  - Big formulas! Infinite state!
- First step: solve conjunctions of expressions over these theories

Little Engines of Proof (LEP)

- In contrast to one size fits all uniform proof procedures (e.g., resolution), LEP focuses on efficient solutions to important cases, and making them work together
- In the early lifecycle we have cts quantities (real numbers and their derivatives), integers, other infinite and rich domains
- Later in the lifecycle, we have bounded integers, bitvectors, abstract data types
- Several of these theories are decidable, such as
  - Real closed fields
  - Integer linear arithmetic
  - Equality with uninterpreted functions
  - Fixed-width bitvectors
- Challenge is: decide their combination and to do it efficiently

Decision Procedures

- Tell whether a formula is inconsistent, satisfiable, or valid
- Or whether one formula is a consequence of others
  - E.g., does $4 \times x = 2$ follow from $x \leq y$, $x \leq 1 - y$, and $2 \times x \geq 1$ when the variables range over the reals?

  Can use heuristics for speed, but must always terminate and give the correct answer
- Most interesting formulas involve several theories
  - E.g., does

$$f(cons(4 \times car(x) - 2 \times f(cdr(x)), y)) = f(cons(6 \times cdr(x), y))$$

  follow from $2 \times car(x) - 3 \times cdr(x) = f(cdr(x))$?

  Requires the theories of uninterpreted functions, linear arithmetic, and lists simultaneously

Deciding Combinations Of Theories

- We want methods for deciding combinations of theories that are modular (combine individual decision procedures), integrated (share state for efficiency), and sound
- Need to make some compromises
  - The combination of quantified integer linear arithmetic with equality over uninterpreted functions is undecidable

  But the ground (unquantified) combination is decidable
- Our method (Shostak) works for theories that are canonizable and solvable
  - Most theories of practical concern
  - Others can be integrated using the slower method of Nelson-Oppen
  - Or by a new insight that relaxes solvability

Shostak's Method

- Yields a modular, integrated, sound decision procedure for the combined theories
  - Invented at SRI more than 20 years ago
  - Developed continuously since then
  - First correct treatment published in 2002
  - Correctness has been formally verified in PVS
  - Previous/other treatments are incomplete, nonterminating, don't work properly for more than two theories
- Combination of canonizers is a canonizer for the combination
  - Independently useful—e.g., for compiler optimizations
  - Assert path predicates leading to two expressions; expressions are equal if they canonize to identical forms

Deciding Combinations Of Theories

Including Propositional Calculus

- So far, can tell whether one formula follows from several others—satisfiability for a conjunction of literals
- What if we have richer propositional structure
  - E.g., $x < y \wedge (f(x) = y \vee 2 * g(y) < \epsilon) \vee \ldots$ for 1000s of terms
- Should exploit search strategies of modern SAT solvers
- So replace the terms by propositional variables
- Get a solution from a SAT solver (if none, we are done)
- Restore the interpretation of variables and send the conjunction to the core decision procedure
- If satisfiable, we are done
- If not, ask SAT solver for a new assignment—but isn't it expensive to keep doing this?

Deciding Combinations Of Theories Including Propositional Calculus (ctd.)

- Yes, so first, do a little bit of work to find fragments that explain the unsatisfiability, and send these back to the SAT solver as additional constraints (i.e., lemmas)
- Iterate to termination
- We call this "lemmas on demand" or "lazy theorem proving"
- Example, given integer x: $(x < 3 \wedge 2x \geq 5) \vee x = 4$
  - Becomes $(p \wedge q) \vee r$
  - SAT solver suggests $p = T, q = T, r = ?$
  - Ask decision procedure about $x < 3 \wedge 2x \geq 5$, it says No!
  - Add lemma $\neg(p \wedge q)$ to SAT problem
  - SAT solver then suggests $r = T$
  - Interpret as $x = 4$ and we are done
- It works really well

ICS: Integrated Canonizer/Solver

- ICS is our implementation of everything just described
  - And some things not described: proof objects, rich API

  ICS decides the combination of unquantified integer and real linear arithmetic, bitvectors, equality with uninterpreted functions, arrays, tuples, coproducts, recursive datatypes (e.g., lists and trees), and propositional calculus
  - Linear arithmetic solver uses a fast new method
- Its SAT solver is specially engineered for this application
  - Large gains over loose combination with commodity SAT solver

  Benchmarking confirms ICS is competitive as a SAT solver, orders of magnitude faster than other decision procedures
- Accessed as a C library, can be called from virtually any language, also has an interactive ascii front end

ICS (continued)

- Developed under RedHat Linux, but ported to Solaris, MAC OS X, and to Cygwin (for Windows)
- Discharges tens of thousand ESC-type problems per second
- Can be used instead of legacy decision procedures in PVS
- Used in SAL (see later)
- Free for noncommercial purposes under license to SRI
- Visit **`ics.csl.sri.com`** or **`ICanSolve.com`**
- Plans include integer completeness, nonlinear arithmetic, quantifier elimination, definition expansion
  - And more builtin glue logic
- Anything previously done with a SAT solver (e.g., diagnosis, planning, controller synthesis) can be done better with ICS

SAL: Symbolic Analysis Laboratory

- SAL is our system for analyzing state machines
- Civilized (intermediate) language, similar to PVS
  - Parameterized modules, subtypes etc.
  - Specialized to transition systems
  - Both guarded commands and SMV-like assignments
  - Synchronous and asynchronous composition
  - Orthogonal assertion languages (currently LTL and CTL)
- State-of-the-art SMC and BMC model checkers for LTL
  - SMC uses CUDD, BMC can use several SAT solvers
- Unique infinite bounded model checker for LTL
  - Can use several decision procedures
- Unique witness model checker (WMC) for CTL
- Pretty good explicit state model checker
- Scriptable (Scheme) interface over powerful API

Extending (Infinite and Finite) BMC to Verification

- In BMC, we should require that $s_0, \ldots, s_k$ are distinct
  - Otherwise there's a shorter counterexample
- And we should not allow any but s_0 to satisfy I
  - Otherwise there's a shorter counterexample
- If there's no path of length k satisfying these two constraints, and no counterexample has been found of length less than k, then we have verified P
  - By finding its finite diameter
- Seldom works in practice

Alternative: Automated k-Induction

- Ordinary inductive invariance (for P):

  Basis: $I(s_0) \supset P(s_0)$

  Step: $P(r_1) \wedge T(r_1, r_2) \supset P(r_2)$

- Extend to induction of depth k:

  Basis: No counterexample of length k or less

  Step: $P(r_1) \wedge T(r_1, r_2) \wedge P(r_2) \wedge \cdots \wedge P(r_{k-1}) \wedge T(r_{k-1}, r_k) \supset P(r_k)$

  These are close relatives of the BMC formulas

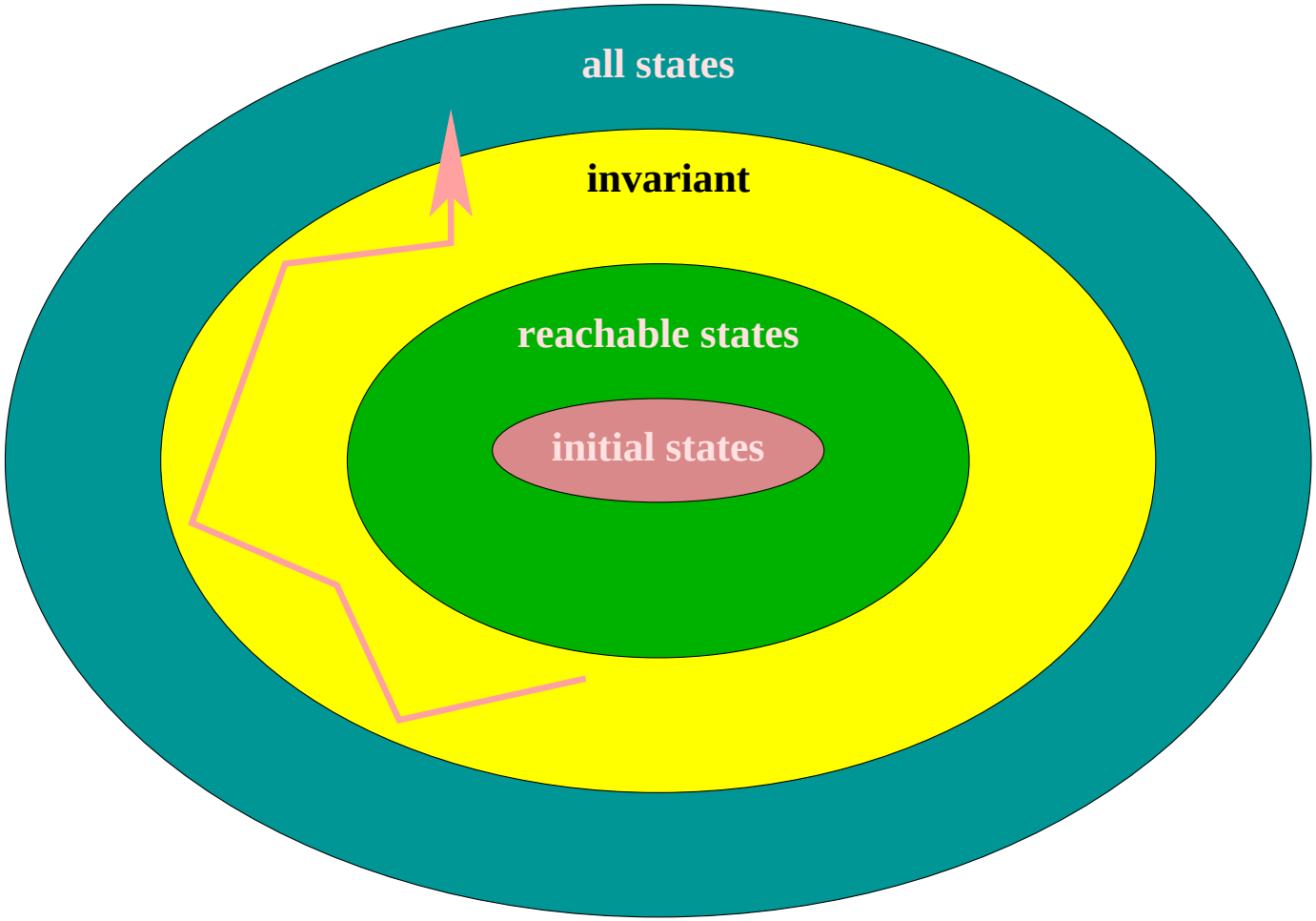
- Induction for $k = 2, 3, 4 \ldots$ may succeed where $k = 1$ does not

- Avoid loops and degenerate cases in the antecedent paths as in BMC

- Method does work in practice

k-Induction is Powerful

Violations get harder as k grows



SAL Infinite Bounded Model Checker

- Shall I do a demo? Lamport's Bakery Algorithm

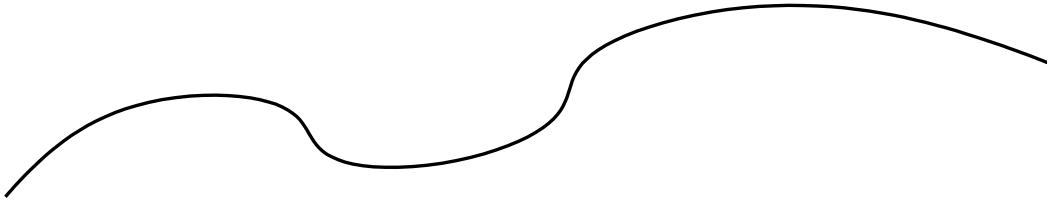
SAL Infinite Bounded Model Checker: Examples

- Can be used to encode timed automata, and k-induction is complete
- Can often find more natural direct encodings for real time
  - Fischer's real time mutual exclusion algorithm (n-process for explicit n upto about 18)
  - TTA startup (in progress)
- Authors: Leonardo de Moura, Harald Rueß, Maria Sorea, Shankar

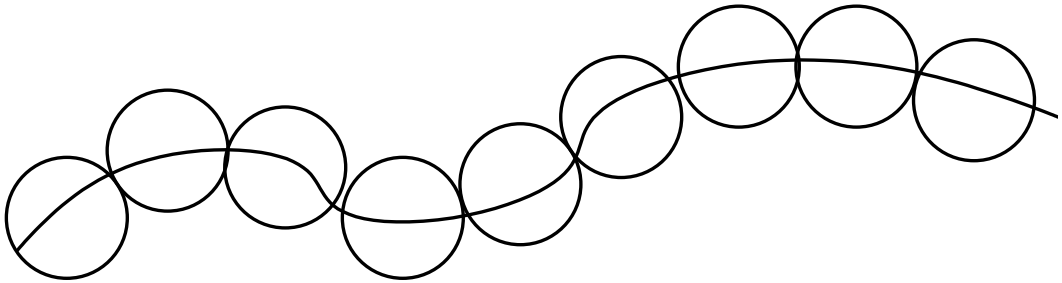
Aside: Integrating BMC With Informal Methods

- With big problems, may be unable to take k far enough for BMC to get to interesting states
- So, instead, start from states found during random simulation
- Can be seen as a way to amplify the power of simulation
- Or to extend its reach

Amplifying The Power Of Simulation



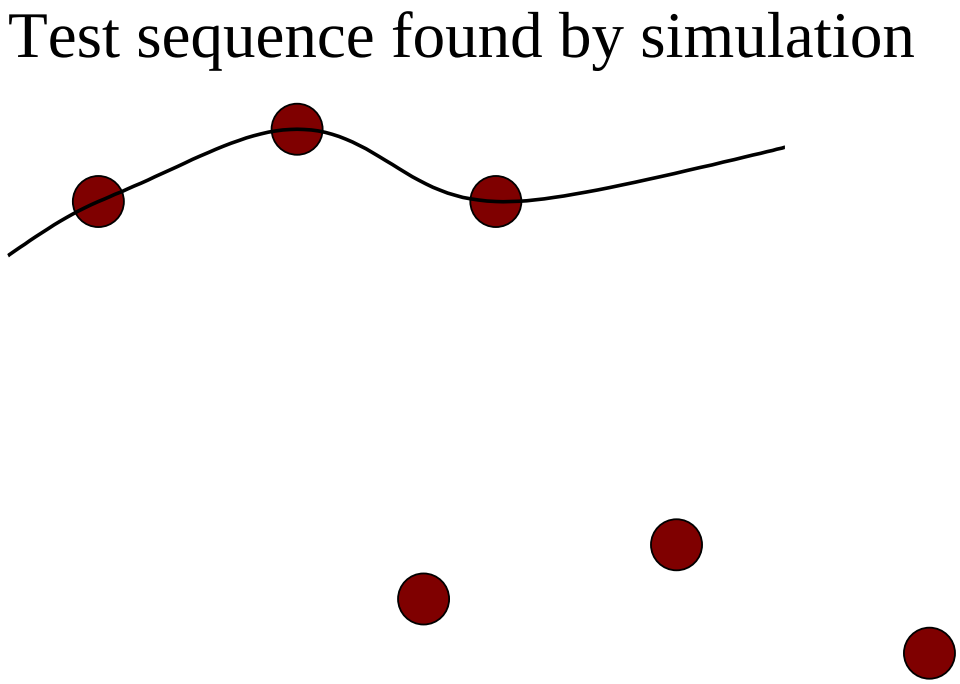
Test sequence found by simulation



Test sequence amplified by bounded model checking

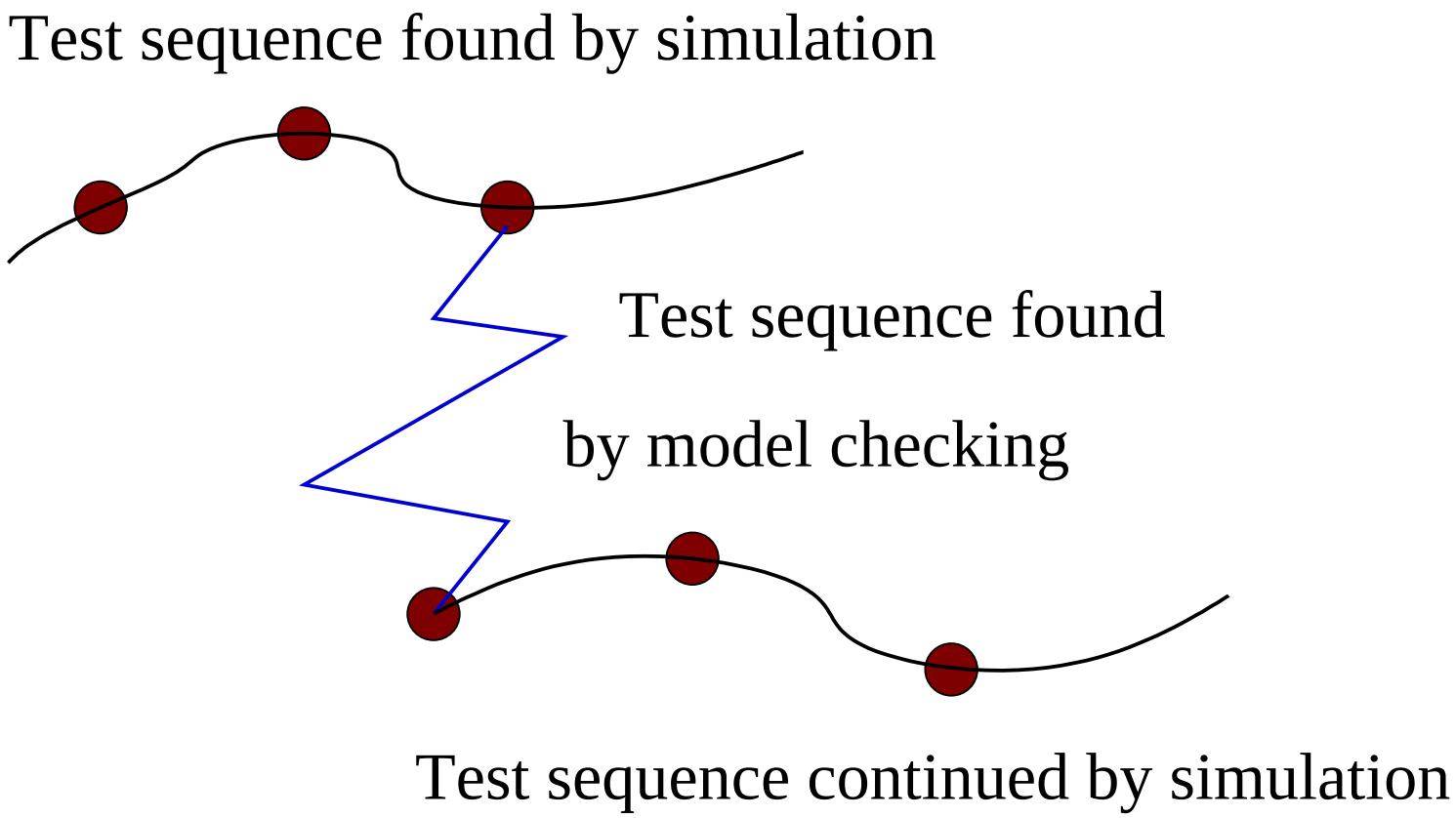
Extending The Reach Of Simulation

Random simulation can have trouble reaching some parts of the state space



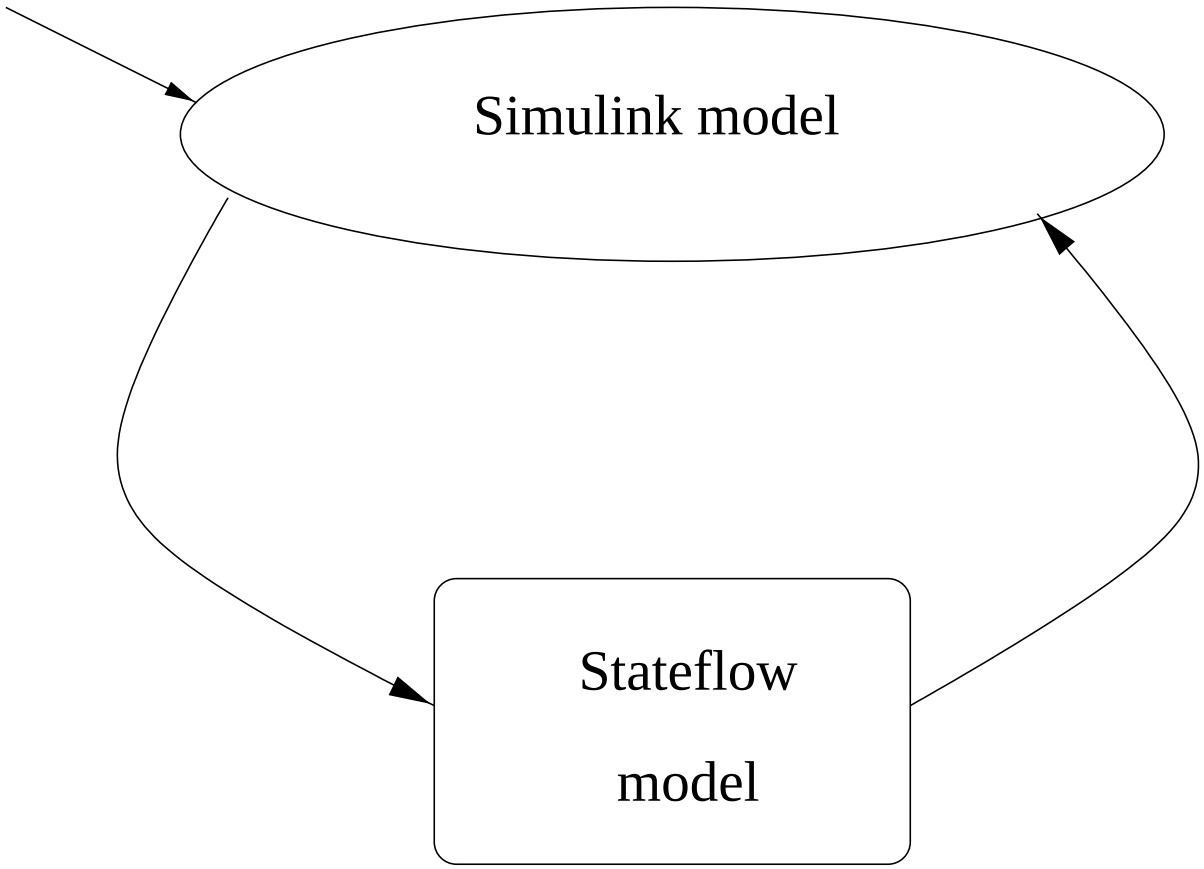
Extending The Reach Of Simulation

So use BMC to jumpstart entry into those parts



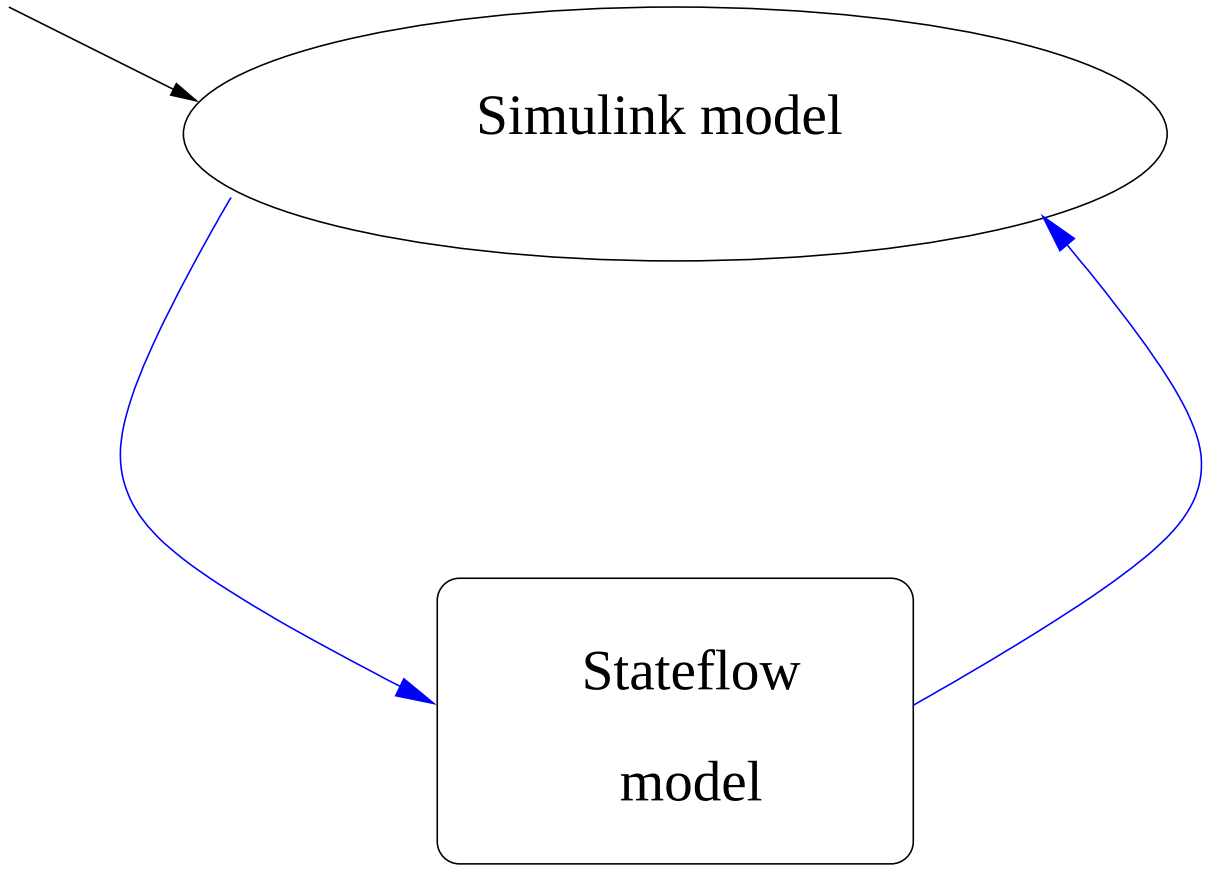
Analyzing Hybrid Systems

(e.g., Matlab Simulink/Stateflow)



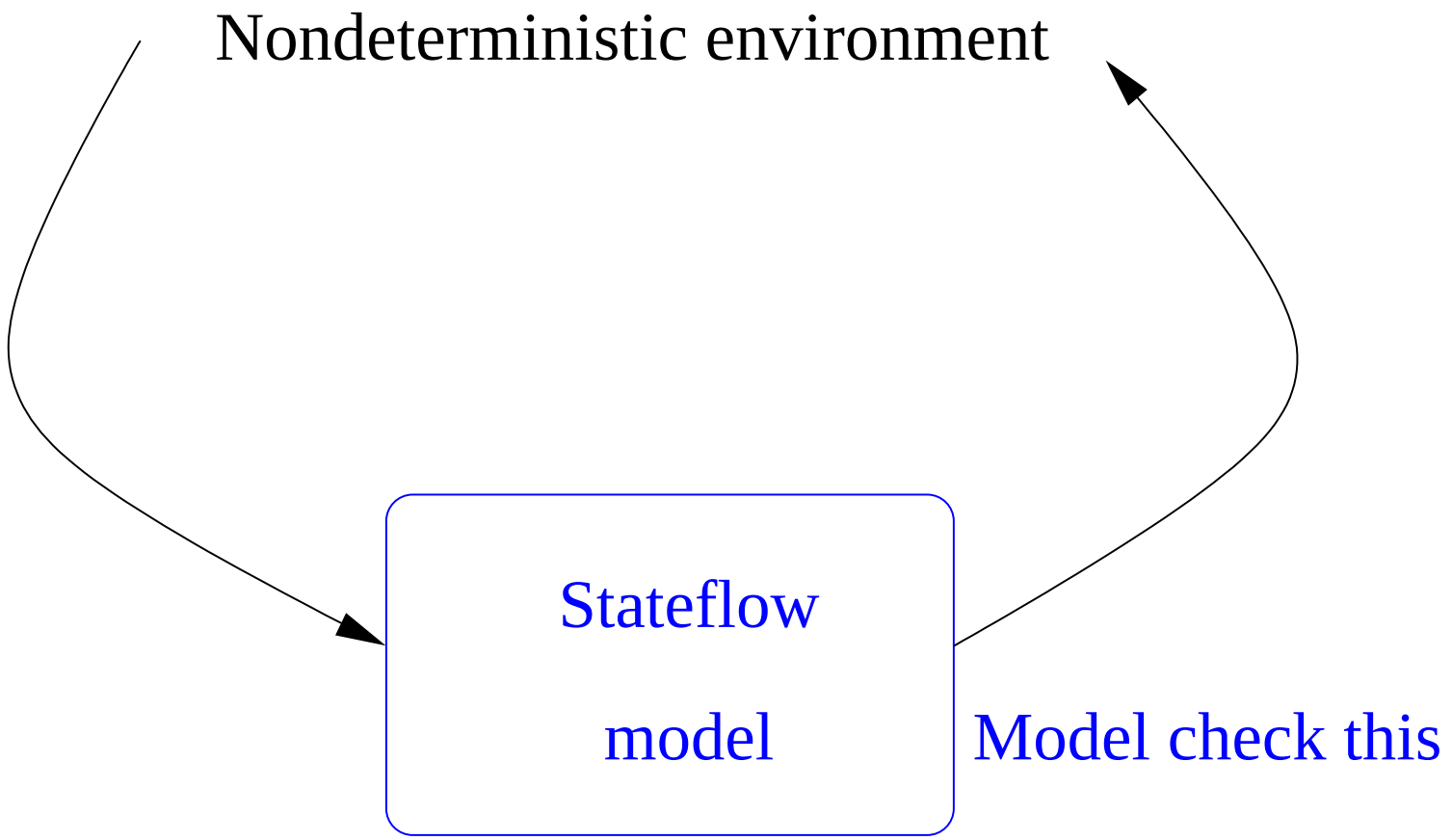
Mixed continuous/discrete (i.e., hybrid) system

Simulate One Trajectory at a Time



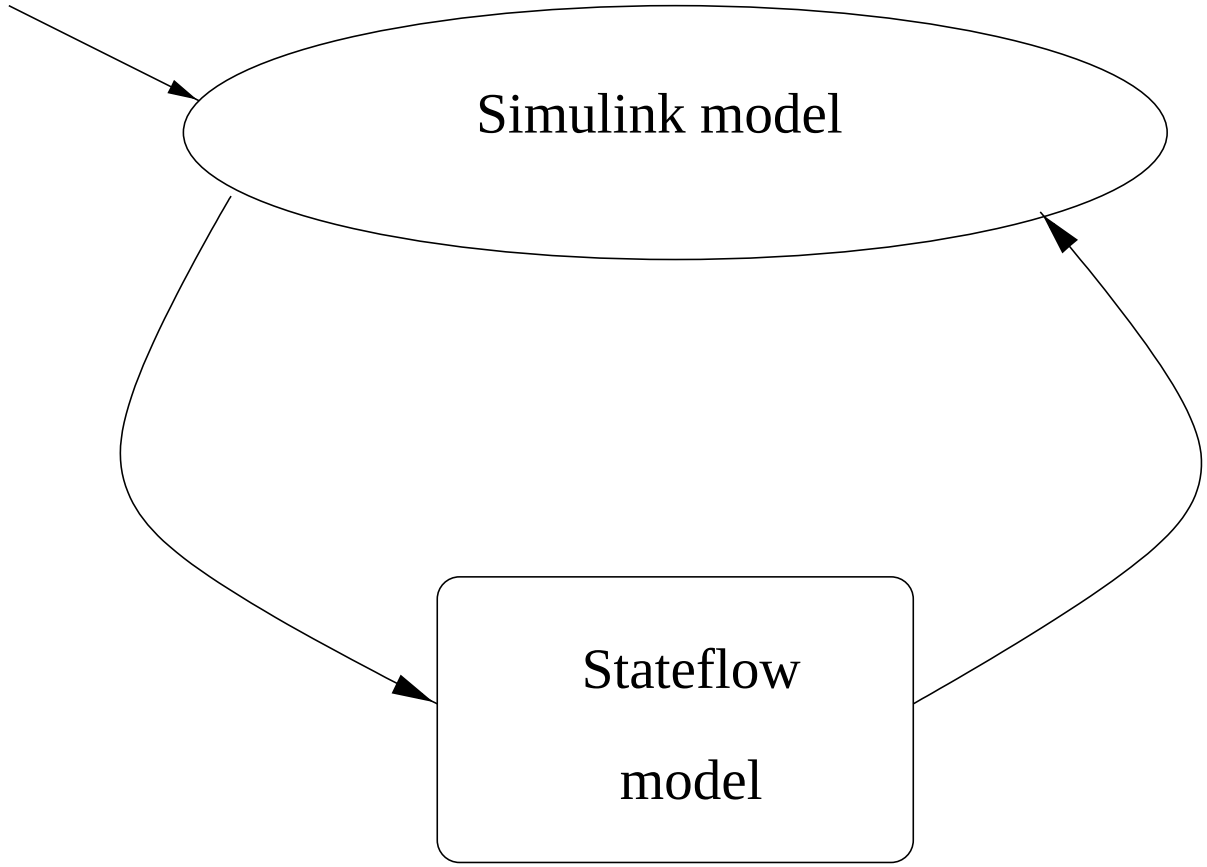
Just like testing: when have you done enough?

Model Check With Nondeterministic Environment



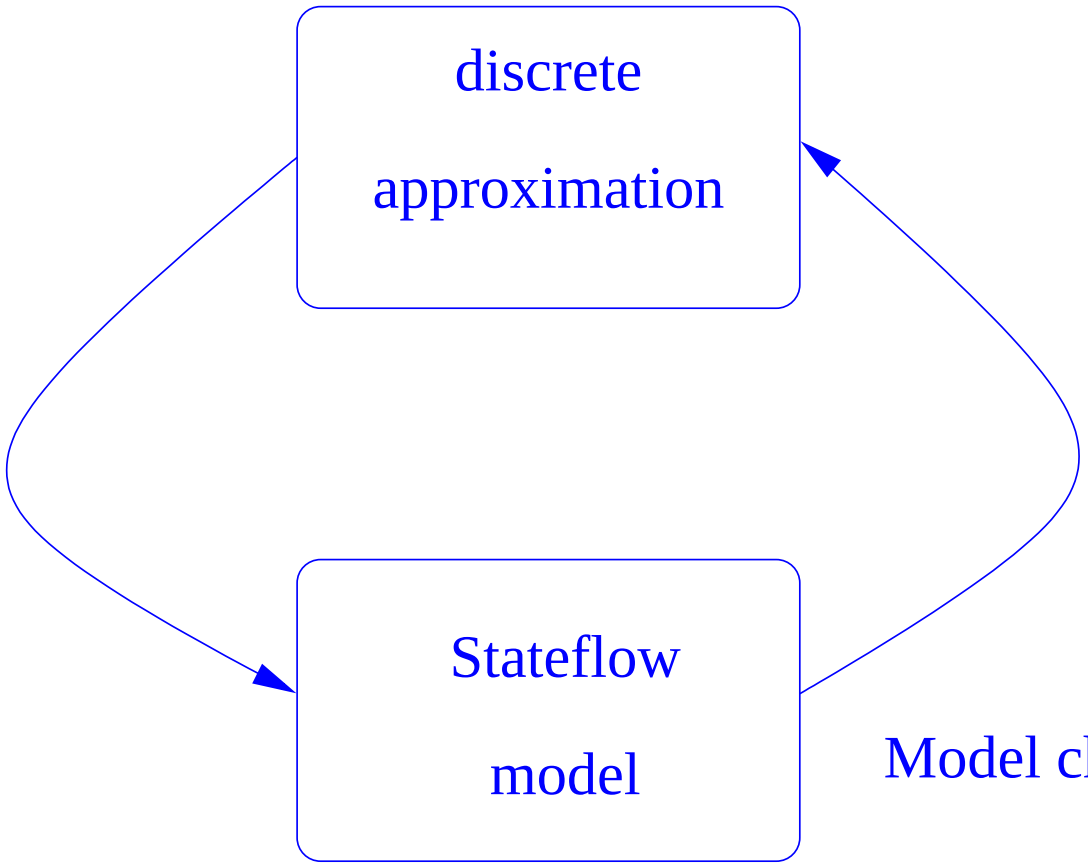
Too crude to establish useful properties

Analyze By The Classical Methods Of Hybrid Systems



OK, but restricted

Model Check With Sound Discretization Of The Continuous Environment



Model check all of this

Just right

Overview of Hybrid Abstraction

- A method for formal analysis of hybrid systems
- Given a (closed) hybrid system and a safety property
- Automatically construct a sound discrete approximation
- And model check that discrete system
- The approximation is constructed by hybrid abstraction
  - Replace polynomials over continuous variables by discrete variables ranging over qualitative signs: $\{$ neg, zero, pos $\}$
  - Compute the abstracted transition relation by automated theorem proving over real closed fields
- Method developed by Ashish Tiwari
  - Tiwari and Khanna HSCC '02

Hybrid Abstraction

- The continuous environment is given by some collection of (polynomial) differential equations on $\mathbf{R}^n$
- Divide these into regions where the first j derivatives are sign-invariant (m polynomials, $(m \times j)^3$ regions)
  - I.e., data abstraction from $\mathbf{R}$ to $\{-, 0, +\}$
  - For each mode $l \in \mathbf{Q}$: if q_{pi}, q_{pj} abstract p_i, p_j and $\dot{p_i} = p_j$ in mode l, then apply rules of the form:
    - ⋆ if $q_{pi} = +$ & $q_{pj} = +$, then q'_{pi} is $+$
    - ⋆ if $q_{pi} = +$ & $q_{pj} = 0$, then q'_{pi} is $+$
    - ⋆ if $q_{pi} = +$ & $q_{pj} = -$, then q'_{pi} is either $+$ or 0
    - ⋆ ...

Data Abstraction for Hybrid Systems

- Larger choices of j give successively finer abstractions
- Usually enough to take $j = 1$ or 2
- Method is complete for some (e.g., nilpotent) systems
- Parameterized also by selection of polynomials to abstract on
  - The eigenvectors are a good start
  - Method is then complete for linear systems
- Construction is automated using decision procedures for real closed fields (e.g., Cylindric Algebraic Decomposition—CAD)
- Also provides a general underpinning to qualitative reasoning as used in AI

Example: Thermostat

Consider a simple thermostat controller with:

- Discrete modes: Two modes, $q = on$ and $q = off$
- Continuous variable: The temperature x
- Initial State: $q = off$ and $x = 75$
- Discrete Transitions:

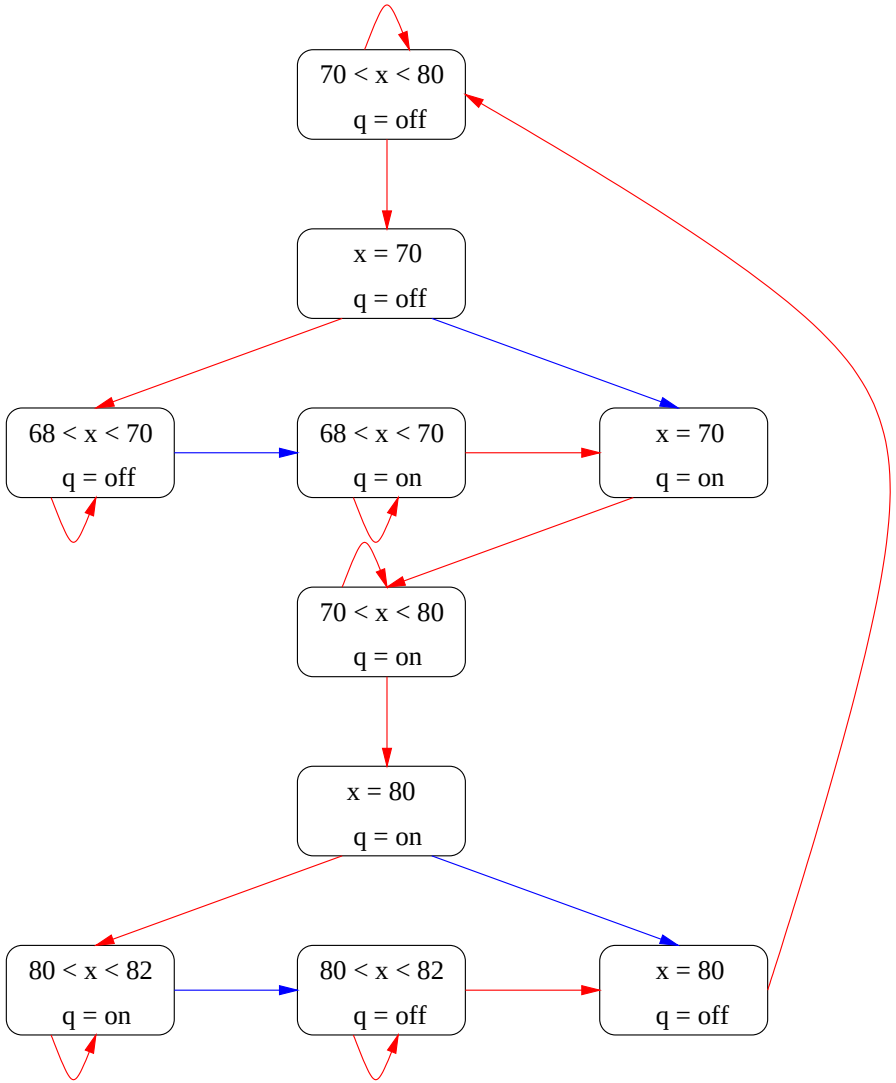
$$q = off \ and \ x \leq 70 \quad \longrightarrow \quad q' = on$$
$$q = on \ and \ x \geq 80 \quad \longrightarrow \quad q' = off$$

- Continuous Flow:

$$q = off \ and \ x > 68 \quad \longrightarrow \quad \dot{x} = -Kx$$
$$q = on \ and \ x < 82 \quad \longrightarrow \quad \dot{x} = K(h - x)$$

We want to prove $68 \leq x \leq 82$

Abstract Thermostat System



Enhancements to Hybrid Abstraction

- Need automated theorem proving over real closed fields
  - Procedure is called many thousands of times (up to 3^{2p} where p is number of polynomials)
- Used to use QEPCAD as decision procedure for RCF
  - Slow and brittle
- Context of use does not require completeness
  - Failure tolerant theorem proving
  - Incompleteness (failure to prove a true theorem) just makes the approximation more approximate
- Ashish Tiwari has developed a method optimized for this application

New (Partial) Decision Procedures for RCF

- Combines superposition calculus for constructing Gröbner bases of polynomial equations
- And ordered chaining calculus for inequalities
- Much faster than QEPCAD
  - E.g., 2 ms. vs 4240 ms. on

$$\{r_a + 5r_b^2 - 20r_b = 0, 25r_b^3 - 100r_b^2 + 20r_b - 1 = 0, r_a > 0, r_b > 0, 4 - v > 0, v + 4 > 0, 1 - a > 0, h - 1 > 0, a + 1 > 0, 10 - g > 0, g - 4a - 4v > 0, 1 - 2h > 0, g - r_a v - r_b a = 0\}$$

- Actually more complete in practice than QEPCAD
- Also does witness generation (proof objects)

Pruning Infeasible States

- Some abstract states are infeasible
  - E.g., x is neg while $x - 3$ is pos
- Prune these out by calling the decision procedure
  - As each new abstract state is generated: concretize, and check for feasibility
  - If infeasible, extract proof object (minimal conflict) so never generate states with similar conflicts in future
- Huge reduction in number of abstract states considered
- And in number of calls on decision procedure

Abstracted Transition Relations are Analyzed Symbolically

- Previously, used the explicit state model checker of SAL 1
  - And checked for feasibility during reachability
  - Slow and defeated by big systems
- SAL 2 has symbolic (using CUDD), bounded (using SAT), and Infinite-Bounded (using ICS) model checkers
- Hybrid analysis uses the symbolic model checker
- Hundreds of times faster than explicit state

Abstraction Is Compositional

- Each module is abstracted separately
- Scalable!
- Requires that model has limited interaction
  - Variables appearing in a guard or condition must come from a single module
- Not a problem in practice

Example: Powertrain Model from Ford

- As described in Alonkrit Chutinan and Ken Butts in MoBIES Baseline Report (April 2002)
- 6 continuous variables (and some defined in terms of these)
- 50 polynomials
- About 300 lines of HybridSAL (hand coded from Matlab file)
- Hybrid abstracter takes around 5 minutes (real time) to create the abstractions (about 9,000 lines of SAL)
  - Completely automatically
- SAL symbolic model checker takes about 3 minutes to check abstracted system (10^{14} reachable states)
  - Completely automatically

Powertrain Model

- Two main components

  Plant: model of the physical transmission and engine

  Controller: model of the shift scheduler

- The inputs to the model

  tps: throttle position

  grade: the angle of the road

- Model dynamics
  - Controller chooses a gear based on the current velocity of the car and the throttle position
  - This information is transmitted to the transmission via a change in the clutch pressure
  - Transmission eventually shifts based on clutch and reaction torques

Modeling the Powertrain in HybridSAL

- Composition of three modules
  - ShiftScheduler, PlantModeSelector, and PlantDynamics
- PlantDynamics
  - Specified using an “inside-out” representation
  - This avoids explicit enumeration of the modes of the system
- InvariantSection
  - Used to specify global invariants (constraints) on the state variables and parameters

Properties of interest

State consistency: [Ken Butts: Mechatronics 2000]

- The controller and the plant “agree” on the current mode (gear) of the system
- I.e., synchronized plant and controller states

Chattering:

- For a fixed tps and grade, the transmission does not switch between 1-2-1 or 2-1-2

Results of HybridSAL Analysis

State consistency property:

- One direction of the state consistency problem was proved
- The other direction produced a counter-example
  - The counter-example can be eliminated by changing the priorities on two (nondeterministic) transitions in the ShiftScheduler
  - The original model described in the MoBIES baseline report has a delay loop in the discrete component, which we did not model
- The above experiments were done under a variety of assumptions on the tps and grade
  - tps and grade are symbolic constants (values unknown, but unchanging)
  - tps and grade can increase arbitrarily from some initial configuration

Results of HybridSAL Analysis (ctd)

Chattering (1-2-1, 2-1-2):

- We proved that these behaviors do no occur when
  `0 <= tps <= 100` and `0 <= grade <= 0.5`
  and tps and grade do not change with time

Other Applications (BioSpice)

Delta Notch Signaling:

- 8 continuous variables
- With Ronojoy Ghosh and Claire Tomlin (Stanford)
- Verified stable configurations of 4-cell cluster
- See HSCC '03

Sporolation in B. subtilis:

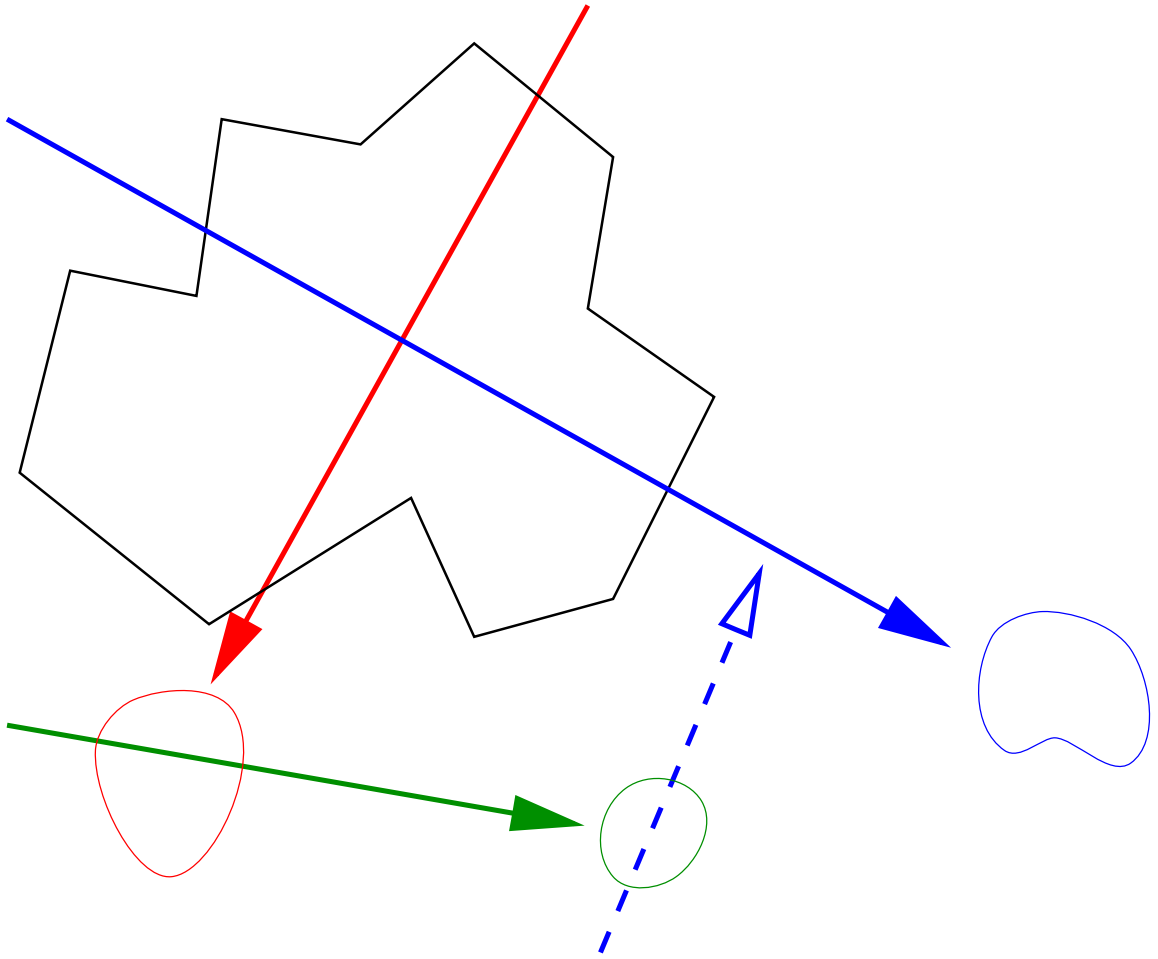
- 15 continuous variables
- With Denise Wolf and Adam Arkin (Berkeley)
- Many differential equations, but no rate constants
- HybridSAL is used for parameter estimation
- Put symbolic rate constants in HybridSAL model
- Add constraints until behavior is biologically sound
- Will investigate properties of the organism
  (e.g., fault tolerance) when this is complete

Putting It All Together: Maybe a Paradigm Shift

- It is now fairly routine to have model checkers as backends to theorem provers (e.g., PVS), or proof assistants as front ends to model checkers (e.g., Cadence SMV)
- But we envisage a larger collection of symbolic computational procedures
  - Decision procedures, abstractors, invariant generators, model checkers, static analyzers, test generators, ITPs
- Interacting through a scriptable tool bus
- The bus manages symbolic and concrete artifacts
  - Test cases, abstractions, theorems, invariants

  Over which it performs evidence management
- Focus shifts from verification to symbolic analysis
  - Iterative application of analysis to artifacts to yield new artifacts, insight and evidence

Integrated, Iterated Analysis



Summary: Technology

- The technology of automated deduction (and the speed of commodity workstations) has reached a point where we can solve problems of real interest and value to developers of embedded systems
- Embodied in our systems

  PVS.csl.cri.com: comprehensive interactive theorem prover

  ICS.csl.sri.com: embedded decision procedures

  SAL.csl.sri.com: model checking toolkit (explicit, symbolic, bounded, infinite-bounded), and (soon) tool-bus
- And in numerous papers accessible from `http://fm.csl.sri.com`, including our Roadmap

A Bigger Vision: 21st Century Mathematics

- The industrialization of the 19th and 20th century was based on continuous mathematics
  - And its automation
- That of the 21st century will be based on symbolic mathematics
  - Whose automation is now feasible

  Allows analysis of systems too complex and numerically too indeterminate for classical methods
- Example: symbolic systems biology
  - Knockouts in E.Coli (SRI; Maude)
  - Cell differentiation in C.Elegans (Weizmann; Play-in/out)
  - Delta-Notch signaling (SRI, Stanford; Hybrid SAL)
  - Sporolation in B.Subtilis (SRI; Hybrid SAL)