Verification of Hybrid Systems in Coq

H. Geuvers, A. Koprowski, D. Synek, E. van der Weegen
BRICKS AFM4
Advancing the Real use of Proof Assistants

Foundations group, Intelligent Systems, ICIS
Radboud University Nijmegen
The Netherlands

Dutch Model Checking Day
April 2, 2009,
University of Twente

Overview

- What is Coq?
- What is a Hybrid System?
- Example: Thermostat
- Semantics: Transitions and traces
- Proving properties of Hybrid Systems by the Abstraction method
- What we have done in Coq and what we plan to do.

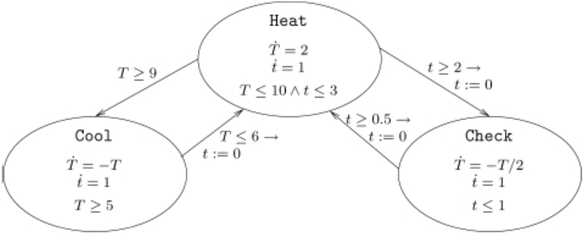
What is Coq?

Coq is a proof assistant based on type theory

- Definitions, Lemmas, Proofs
- A proof p of a formula A is a term $p : A$.
  proof-checking = type checking
- Small kernel (the type checker) + Proof engine on top (to interactively create terms)
- One can define (inductive and abstract) data types
  Define executable functions over these in Coq
- Program extraction to OCaml / Haskell
  $p : \forall x : A. \exists y : B. R(x, y)$ extract $f : A \to B$ satisfying the specification.

What is a Hybrid System?

Alur, Henziger et al.: Hybrid Automaton, Hybrid System
Locations, Invariants,
Jumps, Guards, Reset functions,
Continuous behaviour (Flow),
Thermostat example



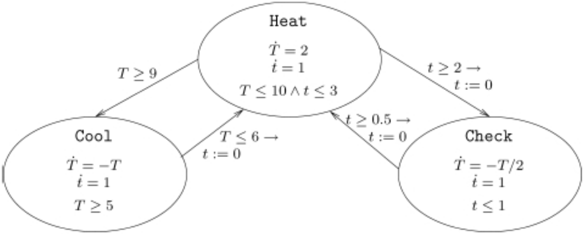
What is a Linear Hybrid System?

$\langle L, \mathcal{X}, X_0, \mathcal{I}, \mathcal{F}, \mathcal{T} \rangle$

- L finite set of locations
- $\mathcal{X} \subset \mathbb{R}^n$ continuous state space
- $X := L \times \mathcal{X}$ state space, $X_0 \subset X$, initial states
- $\mathcal{I}$ assigns to $l \in L$ a set of linear predicates $\mathcal{I}(l) \subset \mathcal{X}$, the invariant at l.
- $\mathcal{F}$ assigns to $l \in L$ a continuous vector field $\mathcal{F}(l) : \mathcal{X} \times \mathbb{R} \to \mathbb{R}^n$. At location l, $\dot{\vec{x}} = \mathcal{F}(l)(\vec{x}, 1)$.
- $\mathcal{T}$ assigns to a pair of locations $\langle l, l' \rangle$ a pair $\langle g, r \rangle$, where g is a predicate, the guard condition, and r is a linear map, the reset function.

Non-determinism

Thermostat example



Invariant $T \leq 10 \wedge t \leq 3$ says when it is allowed to be in Heat
Guard $T \geq 9$ says when it is allowed to move to Cool

Hybrid Systems as Specifications

Hybrid System	=	Specification to be met by the controller.

Spec usually allows a lot of freedom (non-determinism) for the controller.

Hybrid Systems as Specifications

Hybrid System	=	Specification to be met by the controller.

Spec usually allows a lot of freedom (non-determinism) for the controller.

Goal	=	Prove that a controller that satisfies the spec, keeps the system out of bad states

Reachability Problem

Why do this in Coq?

- Verification of Hybrid systems involves discretization, floating point arithmetic approximations, . . . , is this all correct?

Why do this in Coq?

- Verification of Hybrid systems involves discretization, floating point arithmetic approximations, . . . , is this all correct?
- We have a library of (constructive) exact real arithmetic in Coq: CoRN,
  - real number functions as computable functions (exp, log, sin, cos, . . .)
  - arbitrarily close approximations of real numbers (real number expressions)
  - numerical approximations to solutions of differential equations

Can CoRN be used for these type of applications?

Semantics of a Hybrid System

There are two types of transitions

Continuous transition

$$(l, \vec{x}) \rightarrow_C (l, \vec{y})$$

One location, elapse of time t, continuous variables progress according to the flow $\mathcal{F}(l)$

Discrete transition

$$(l, \vec{x}) \rightarrow_D (l', \vec{y})$$

From location l to l', no elapse of time, guard conditions, continuous variables $\vec{x}$ reset to $\vec{y} := r\vec{x}$.

Semantics of a Hybrid System

A trace is a sequence of continuous and discrete steps:

$$(l_1, \vec{x}_1) \rightarrow_C (l_2, \vec{x}_2) \rightarrow_D (l_3, \vec{x}_3) \rightarrow_C (l_4, \vec{x}_4) \rightarrow_C (l_5, \vec{x}_5) \ldots$$

Semantics of a Hybrid System

A trace is a sequence of continuous and discrete steps:

$$(l_1, \vec{x}_1) \rightarrow_C (l_2, \vec{x}_2) \rightarrow_D (l_3, \vec{x}_3) \rightarrow_C (l_4, \vec{x}_4) \rightarrow_C (l_5, \vec{x}_5) \ldots$$

A Hybrid System specifies a collection of traces. We want to prove properties about these.
Thermostat example: Prove that $T \geq 4.5$ always in all possible traces.
(= Correctness proof of the Thermostat controller)

Semantics of a Hybrid System

Solving differential equations??

Semantics of a Hybrid System

Assume for every location l a solution $\Phi(\vec{x_0}, t)$ to the differential equation $\dot{x(\vec{t})} = \mathcal{F}(l)(x(\vec{t}), 1)$, with begin value $x(\vec{0}) = \vec{x_0}$.
So Φ is a flow function:

$$\begin{aligned} \Phi(\vec{x}, 0) &= \vec{x} \\ \Phi(\vec{x}, t+q) &= \Phi(\Phi(\vec{x}, t)), q) \end{aligned}$$

Semantics of a Hybrid System

Assume for every location l a solution $\Phi(\vec{x_0}, t)$ to the differential equation $\dot{x(\vec{t})} = \mathcal{F}(l)(x(\vec{t}), 1)$, with begin value $x(\vec{0}) = \vec{x_0}$.
So Φ is a flow function:

$$\begin{array}{rcl} \Phi(\vec{x}, 0) & = & \vec{x} \\ \Phi(\vec{x}, t+q) & = & \Phi(\Phi(\vec{x}, t)), q) \end{array}$$

For the Thermostat:

Cool:	$\Phi((x,y),t) = (x\,e^{-t}, y+t)$
Check:	$\Phi((x,y),t) = (x\,e^{-t/2}, y+t)$
Heat:	$\Phi((x,y),t) = (x+2t, y+t)$

Characterization of continuous and discrete steps

$$(l, \vec{x}) \rightarrow_C (l, \vec{y}) \quad := \quad \exists t \geq 0 (\Phi_l(\vec{x}, t) = \vec{y} \wedge \forall s \in [0, t] : \mathcal{I}_l(\Phi_l(\vec{x}, s)))$$

Characterization of continuous and discrete steps

$$(l, \vec{x}) \rightarrow_C (l, \vec{y}) \quad := \quad \exists t \geq 0(\Phi_l(\vec{x}, t) = \vec{y} \wedge \forall s \in [0, t] : \mathcal{I}_l(\Phi_l(\vec{x}, s)))$$

$$(l, \vec{x}) \rightarrow_D (l', \vec{y}) \quad := \quad \mathcal{T}\langle l, l' \rangle = \langle g, r \rangle \wedge g(l, \vec{x}) \wedge \vec{y} = r(\vec{x}) \wedge \mathcal{I}(l')(\vec{y})$$

Trace: Combination of Continuous steps and Discrete steps.
Goal: Verify a property for all traces.

Proving Correctness via the Abstraction method

- Hybrid Transition System: $(\mathrm{State}, \rightarrow_C, \rightarrow_D, \mathrm{State}_0)$
- Abstract System (Finite Automaton): $(\mathrm{AState}, \rightarrow_A, a_0)$
- Abstraction function $\mathrm{Abs} : \mathrm{State} \rightarrow \mathrm{AState}$ with $\mathrm{Abs}(t_0) = a_0$ for $t_0 \in \mathrm{State}_0$.
- Lemma Correctness:

$$\begin{array}{cc} t \rightarrow_{DC} t' & \text{in HS} \\ \Downarrow & \\ \mathrm{Abs}(t) \rightarrow_A \mathrm{Abs}(t') & \text{in AHS} \end{array}$$

Proving Correctness via the Abstraction method

- Lemma Correctness:

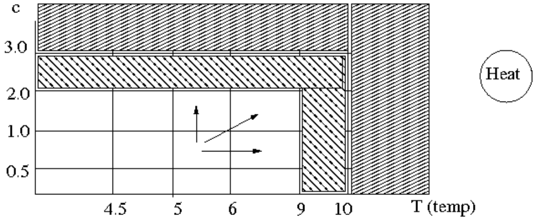
$$\begin{array}{ccc} t \longrightarrow_{DC} t' & & \text{in HS} \\ \Downarrow & & \\ \mathrm{Abs}(t) \longrightarrow_{A} \mathrm{Abs}(t') & & \text{in AHS} \end{array}$$

So: Reachability in HS $\Rightarrow$ Reachability in AHS

So: Safety of AHS $\Rightarrow$ Safety of HS
[Checked by Model Checker]

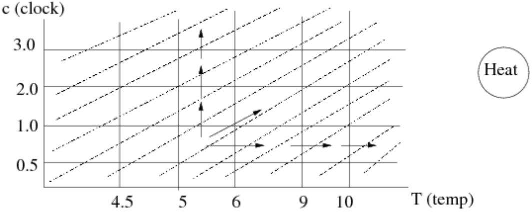
Abstraction via predicates: Thermostat example

The basic predicates are:
$T \geq 4.5$, $T \geq 5$, $T \geq 6$, $T \leq 9$, $T \leq 10$
$c \geq 0.5$, $c \leq 1$, $c \geq 2$, $c \leq 3$.
This gives rise to the following abstract state space
(for location Heat).
Some transitions are indicated.



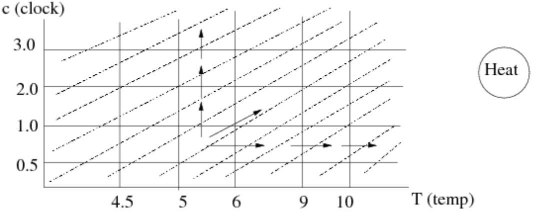
Beware of transitivity

Which abstract traces do we consider?



Beware of transitivity

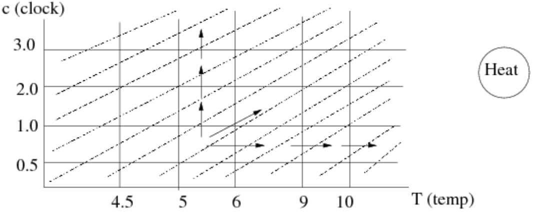
Which abstract traces do we consider?



If we just take the transitive closure of $\mathrm{Abs}(s_0) \longrightarrow \mathrm{Abs}(s_1)$ we get far too many traces. (Still correct, but you can't prove anything!)

Beware of transitivity

Which abstract traces do we consider?

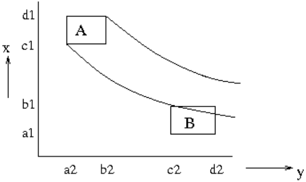


If we just take the transitive closure of $\mathrm{Abs}(s_0) \to \mathrm{Abs}(s_1)$ we get far too many traces. (Still correct, but you can't prove anything!)
Solution: Restrict the Abstract traces to

$$\mathrm{Abs}(s_0) \to_C \mathrm{Abs}(s_1) \to_D \mathrm{Abs}(s_2) \to_C \mathrm{Abs}(s_3) \ldots$$

Moving from the HS to the AHS

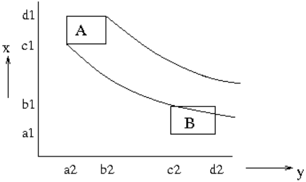
$A \longrightarrow B$ in AHS if $\exists (x, y) \in A \, \exists t \geq 0 (\Phi(x, y, t) \in B)$



This is complicated, in general undecidable ...

Moving from the HS to the AHS

$A \to B$ in AHS if $\exists (x,y) \in A \, \exists t \geq 0 (\Phi(x,y,t) \in B)$



This is complicated, in general undecidable ...
But in concrete situations, we have:

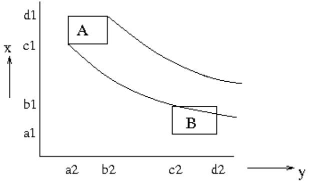
- "independency of variables":

$$\Phi(x,y,t) = (\phi_1(x,t), \phi_2(y,t))$$

- monotonicity of $\phi_1(x,-)$ and $\phi_2(y,-)$.
- concrete inverses to $\phi_1(x,-)$ and $\phi_2(y,-)$.

Moving from the HS to the AHS

$A \to B$ in AHS if $\exists (x,y) \in A \, \exists t \geq 0((\phi_1(x,t), \phi_2(y,t)) \in B)$



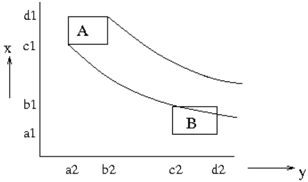
$$\exists (x,y) \in A \, \exists t \geq 0((\phi_1(x,t), \phi_2(y,t)) \in B)$$

if and only if

$$\phi_1^{-1}(c_1, b_1) < \phi_2^{-1}(a_2, d_2) \wedge \phi_1^{-1}(d_1, a_1) > \phi_2^{-1}(b_2, c_2)$$

where ϕ_i^{-1} is the inverse of ϕ_i:
$\phi_i(x, \phi_i^{-1}(x,z)) = z$
$\phi_i^{-1}(x, \phi_i(x,t)) = t$

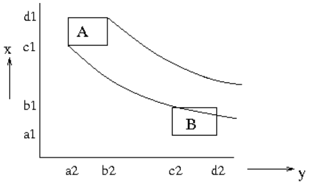
Moving from the HS to the AHS



For the Check location:

$$\phi_1^{-1}(x,z) = \log x^2 - \log z^2 \text{ and } \phi_2^{-1}(y,z) = z - y.$$

Moving from the HS to the AHS



For the Check location:

$$\phi_1^{-1}(x,z) = \log x^2 - \log z^2 \text{ and } \phi_2^{-1}(y,z) = z - y.$$

So:

$$\exists (x,y) \in A \, \exists t \geq 0((\phi_1(x,t), \phi_2(y,t)) \in B)$$

if and only if

$$\log c_1^2 - \log b_1^2 < d_2 - a_2 \wedge \log d_1^2 - \log a_1^2 > c_2 - b_2$$

How do we solve this?

Solving inequalities in Coq

For concrete values $a, b, c, d \in \mathbb{R}$,

$$\log c^2 - \log b^2 < d - a$$

can be "decided" by

Solving inequalities in Coq

For concrete values $a, b, c, d \in \mathbb{R}$,

$$\log c^2 - \log b^2 < d - a$$

can be "decided" by

- fixing an ε,
- approximate $\log c^2 - \log b^2$ and $d - a$ "upto ε", obtaining rational intervals I_1 and I_2,
- If $I_1 > I_2$, return 'no', otherwise, return 'yes'

Solving inequalities in Coq

For concrete values $a, b, c, d \in \mathbb{R}$,

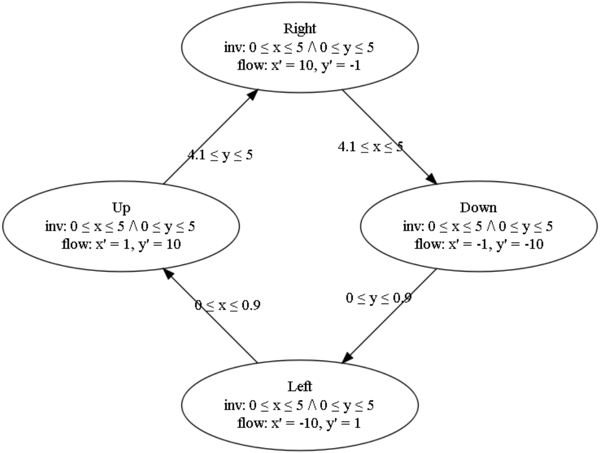
$$\log c^2 - \log b^2 < d - a$$

can be "decided" by

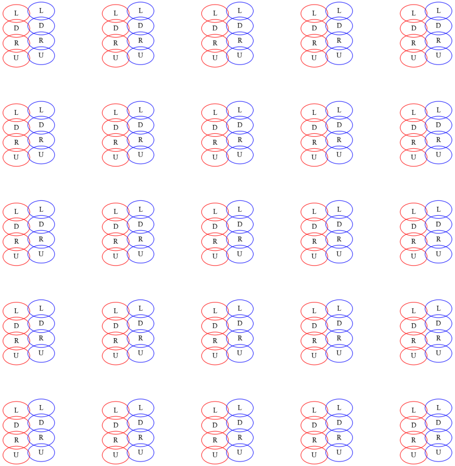
- fixing an ε,
- approximate $\log c^2 - \log b^2$ and $d - a$ "upto ε", obtaining rational intervals I_1 and I_2,
- If $I_1 > I_2$, return 'no', otherwise, return 'yes'

So, if we are undecisive, we do put an arrow between the abstract states ... an abstraction should be an over-approximation.

The rotator example



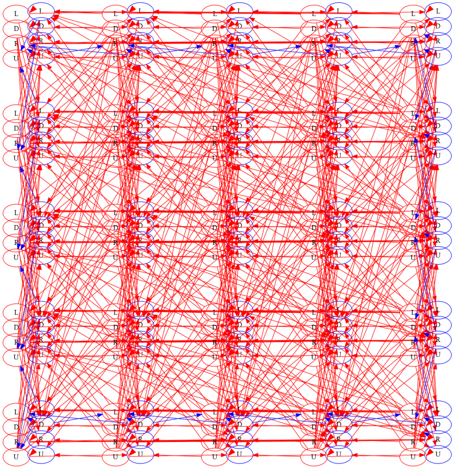
The rotator example: State space



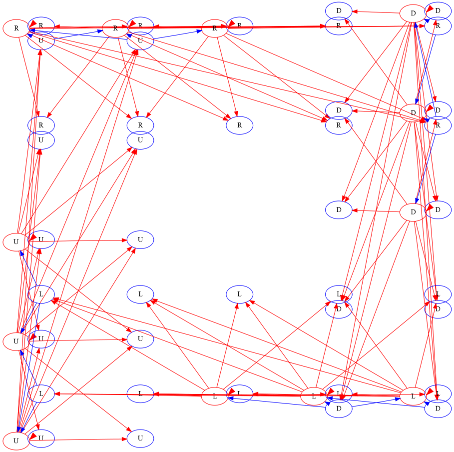
Blue: next step is a “discrete” step

Red: next step is a “continuous” step

The rotator example: All edges



The rotator example: Reachable states and edges



The middle state is unreachable.

How does this actually work in Coq?

1. Specify a concrete Hybrid System,
2. Specify the Abstract states (rectangles)
3. Specify the Safety condition
4. Give the inverses to the flow functions and prove they are inverses.
5. Coq generates the AHS, the abstraction function and its correctness proof.
6. Coq generates a proof of "Reach(AHS) = Safe $\Rightarrow$ HS is safe".
7. Computing Reach(AHS) (in Coq) proves the safety (automatic)

What we plan to do / problems

1. Generate AHS + Abs function from the Specification
   NB Abstraction predicates can be derived from the Spec.
2. Support for generating inverses and proving they are inverses
   NB Many function are partial or partially monotone
3. Extract fast model checking to OCaml: "certified reachability algorithm".
4. Deal with flow functions where variables are not independent or not locally monotone
5. Use numeric approximations to solutions of differential equations.

Thank you!