SMT-AI: an Abstract Interpreter as Oracle for k-induction

Pierre Roux, Pierre-Loïc Garoche, Rémi Delmas

ONERA – DTIM, Toulouse

September 17, 2010

Introduction

Rice Theorem

Every non trivial property on programs is undecidable.

To cope with undecidability, we can withdraw:

- termination;
- completeness;
- automaticity.

Various methods

- model-checking: numerous methods based on a more or less explicit study of the state space of the system;
- abstract interpretation: overapproximation (abstraction) of the semantic of the program, can answer "don't know";
- deductive methods and proof assistants:
  everything can be proved... potentially by hand

A combination of analysis

k-induction

Model-checking method based on SMT solvers.

- Industrially used.
- Requires to manually strengthen the property to prove with numerous lemmas.

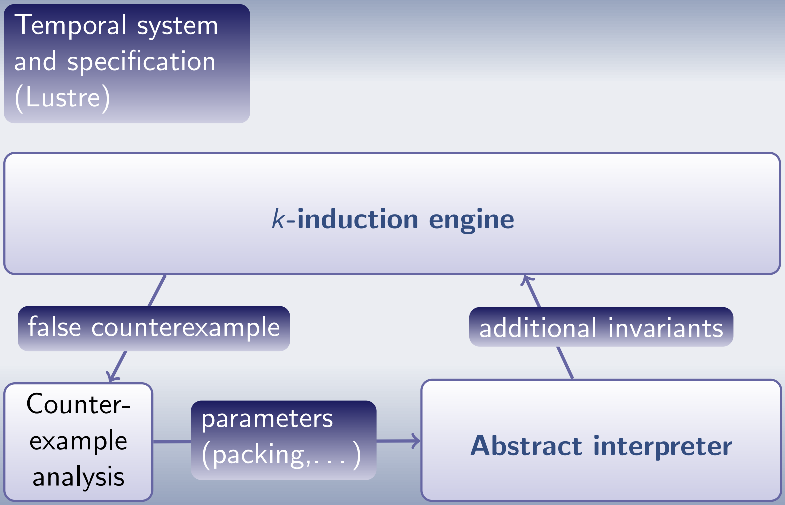
Abstract interpretation

Infers numerical invariants.

- High level of automaticity.
- Result can be imprecise.

Hence the idea of combining them.

A combination of analysis



Synchronous systems

Definition

We define a synchronous system as:

- a (potential) state space S;
- a set of inputs E;
- an initial state $I \in S$;
- a transition relation $T \subseteq S \times E \times S$.

Synchronous languages: Lustre, Scade, Mathlab Simulink

k-induction method

Definitions

$$Base(k) := I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, e_i, s_{i+1}) \wedge \neg P(s_k)$$

$$Step(k) := \bigwedge_{i=0}^{k} P(s_i) \wedge \bigwedge_{i=0}^{k} T(s_i, e_i, s_{i+1}) \wedge \neg P(s_{k+1})$$

Algorithm

- $k := 0$
- if $Base(k)$ is satisfiable, we have a counterexample
- if $Base(k)$ and $Step(k)$ are both unsatisfiable, we proved P for all reachable states
- else ($Base(k)$ unsatisfiable and $Step(k)$ satisfiable), increment k and continue

Remarks

- SAT/SMT solvers are used to decide satisfiability.
- SMT solvers aloow to handle efficiently infinite systems.
- If the property is false, we get a counterexample (bounded model-checking).

Remarks

- SAT/SMT solvers are used to decide satisfiability.
- SMT solvers aloow to handle efficiently infinite systems.
- If the property is false, we get a counterexample (bounded model-checking).
- Under some hypotheses, the method can be complete.
- But in practice, it is often needed to strengthen P with various invariants to terminate with reasonable values of k.

Abstract interpretation principle

Collecting semantic $R \subseteq S$ of a synchronous system can be defined as a least fixpoint (lfp) :

$$R = \mathrm{lfp}\left(\lambda X.I \cup \left\{s' \;\middle|\; \exists s \in X.\exists e.T(s,e,s')\right\}\right)$$

But this is usually **not computale**.

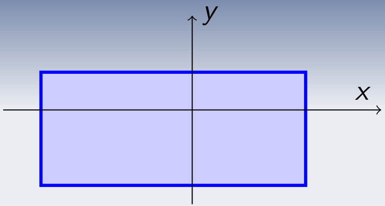
Abstract interpretation framework

- an abstract domain $S^\sharp$;
- a concretization function $\gamma : S^\sharp \to \mathcal{P}(S)$
  ($s^\sharp \in S^\sharp$ is an overapproximation of $s \in S$ si $s \in \gamma(s^\sharp)$) ;
- an abstract transition relation $T^\sharp : S^\sharp \to S^\sharp$ computable and sound with respect to T : $\left\{s' \;\middle|\; \exists s \in \gamma(s^\sharp).\exists e.T(s,e,s')\right\} \subseteq \gamma(T^\sharp(s^\sharp))$.

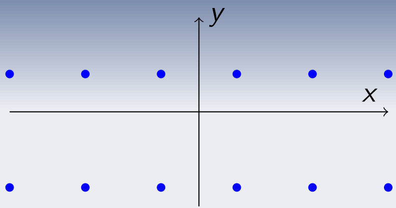
We can **compute** an overapproximation of R :

$$R^\sharp = \bigsqcup_n^\sharp {T^\sharp}^n(I^\sharp) \qquad (+\text{ widening})$$

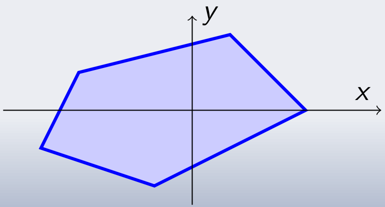
Examples of abstract domains



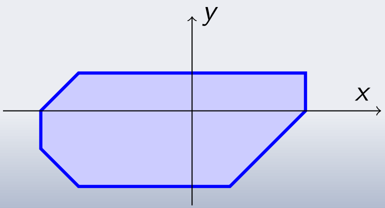
intervals



congruences



polyhedra



octagons

But relational domains are expensive.

Input language

We compile **Lustre** code to an extension of **SMT-lib** :

- close from input language of SMT solvers;
- simplicity (no more clocks, purely functional,. . .);
- well known compilation process;
- functional languages unusual in abstract interpretation litterature.

Abstract interpretation analysis

- Forward/backward analysis of SMT-lib formulas:
  - forward: we evaluate expressions;
  - backward: we constrain environments in which the expression can evaluate to expected value;
- Greates fixpoint computation;
- "paper" proof of soundness and termination.

Implementation

- relational domains thanks to APRON library;
- trace partitioning;
- interface for multiple analysis
  reusing previous results:
  - with various relational packs;
  - under different contexts;
  - ...
- 10 k lines of Caml, under GPL ;

Collaboration example

Exemple

On following code :

node collaboration(a : bool) **returns** (OK : bool)
var pre_x : int; pre_y : int; x : int; y : int; z : int; n : int; **let**
pre_x = fby(0, 1, x); pre_y = fby(0, 1, y); n = 21;
x = 0 → **if** pre_x < 2∗n **then** pre_x + 1 **else** pre_x;
y = 2∗n → **if** pre_y > 0 **then** pre_y − 1 **else** pre_y;
z = x∗y;
OK = z ≤ n∗n; **tel**

- *k*-induction :
  doesn't terminate
- abstract interpreter :
  also unable to prove invariant $OK = true$
  but infers $x + y = 42\dots$
- …which allows *k*-induction to prove $OK = true$

Future work

- Actual collaboration between k-induction and abstract interpreter.
- Improve our abstract interpreter (better reduced product,...).
- Testing on more examples and industrial case studies.

Questions

?