Kind-AI: When abstract interpretation and SMT-based model-checking meet

Pierre-Loïc Garoche – Onera – U. of Iowa
joint work with T. Kashai and C. Tinelli

04/13/2012
**CONTEXT: SAFETY PROPERTIES FOR CONTROLLER**

- Open/Closed system analysis
- Controller
- Implementation model
- Low level implementation

- Simulink + Proofs
- Model 
  - Lustre + Spec
  - C code + ACSL Spec

- Control theorists
- Computer scientists

- PVS 4 Lustre (RC & NASA)
  - Kind (U.of Iowa)
  - Stuff (ONERA & RCF)

- PVS 4 C (GT & NASA)
**CONTEXT: SAFETY PROPERTIES FOR CONTROLLER**

- Open/Closed system analysis
- Controller implementation model
- Low level implementation

**Simulink**

- + Proofs

**Model**

- lustre
  - + Spec

**C code**

- + ACSL Spec

**Control theorists**

- Computer scientists

**Stuff**

- (ONERA & RCF)

**PVS 4 Lustre**

- (RC & NASA)

**PVS 4 C**

- (GT & NASA)

Kind (U.of Iowa)

- PVS 4 Lustre (RC & NASA)


Motivation:

- prove a safety property over a transition system
- interested in numerical invariants

Available elements/Application

- k-induction engine for the transition system
- numerical abstract domains, ie. APRON
- application to Lustre models analysis
Numerical invariants

- Intervals
- Polyhedra
- Linear templates
- Linear expression under implication, eg. cond_1 and cond_2 \implies linear expression
WHAT FOR?

- Identify an over-approximation of reachable states
  - prove target properties expressed as such invariants
  - enrich the description of the system by make explicit the implicit properties
  - or address more complex user-defined properties by considering only interesting states
- Constrains k-induction
A B S T R A C T I N T E R P R E T A T I O N

- Ideal approach to compute numerical invariants
- But …
Ideal approach to compute numerical invariants
But …
  - results and time to get them depend on
    1. the abstraction used
    2. and speed-up parameters (widening, narrowing)
Abstract interpretation

- Ideal approach to compute numerical invariants
- But ...
  - results and time to get them depend on
    1. the abstraction used
    2. and speed-up parameters (widening, narrowing)
  - (could be) painful to define
Abstract interpretation – The usual picture

\langle E, \sqsubseteq_E \rangle

Set of formulas

\subseteq_E

I
\[
\langle E, \sqsubseteq_E \rangle
\]
Set of formulas
Abstract Interpretation – The usual picture

\[ \langle E, \sqsubseteq_E \rangle \]

Set of formulas
Abstract interpretation – The usual picture

\[ \langle E, \sqsubseteq_E \rangle \]
Set of formulas
ABSTRACT INTERPRETATION – THE USUAL PICTURE

$\langle E, \sqsubseteq_E \rangle$
Set of formulas

$\sqsubseteq_E$

$\text{lfp}_E g_E$

$g_E(I)$

$g_E(I)$

$\sqsubseteq_A$

$\langle A, \sqsubseteq_A \rangle$
Abstract Interpretation – The Usual Picture

\[ \text{Set of formulas} \]

\[ \langle E, \sqsubseteq_E \rangle \]

\[ \langle A, \sqsubseteq_A \rangle \]
Abstract Interpretation – The usual picture

\[ \langle E, \sqsubseteq_E \rangle \]

Set of formulas

\[ \langle A, \sqsubseteq_A \rangle \]
Abstract interpretation – The usual picture

\[ \langle E, \sqsubseteq_E \rangle \]
Set of formulas

\[ \langle A, \sqsubseteq_A \rangle \]
Abstract Interpretation – The usual picture

\[ \langle E, \sqsubseteq_E \rangle \]
Set of formulas

\[ \langle A, \sqsubseteq_A \rangle \]
ABSTRACT INTERPRETATION – THE USUAL PICTURE

\[
\langle E, \sqsubseteq_E \rangle \quad \text{Set of formulas}
\]

\[
\langle A, \sqsubseteq_A \rangle
\]

\[
lfp g_A \quad \gamma
\]

\[
g_A(\alpha(I)) \quad \alpha(I) g_A
\]

\[
lfp g_E \quad \alpha(E)
\]

\[
g_E(I) \quad \alpha(I)
\]

\[
g_E \quad \alpha(I)
\]

\[
\gamma \quad g_A
\]

\[
\sqsubseteq_E \quad \sqsubseteq_A
\]
Basic Ingredients

- Initial semantics expressed as fixpoint of a function $\varrho_E : E \rightarrow E$ over a lattice $\langle E, \sqsubseteq_E \rangle$. Easy for safety analysis: collecting semantics of a transition system $(\Sigma, I, \leadsto_T)$

\[
\text{lfp}_I \lambda X. X \cup \{ x' | x \in X, x \leadsto_T x' \}
\]

- Abstract representation of semantics values, here set of states: abstract domain $\langle A, \sqsubseteq_A \rangle$

- Relationship between original values and abstract ones, ie. a Galois connexion

\[
\alpha : E \rightarrow A \quad \gamma : A \rightarrow E
\]

- Sound abstract transformers to mimic the concrete transitions in the abstract $\varrho_A : A \rightarrow A$

\[
\text{lfp}_{\alpha(I)} \varrho_A
\]
Abstract domains

Intervals

Polyhedra

Congruences

Octagons
Abstract Transformers

Usually the transition relation $\sim_T: \Sigma \to \Sigma$ is defined using smaller operators

- control flow ops: branching statements, loops, function calls, automaton transitions for FSM
- data flow ops: assigns of a variable, clock issues
- expression wise: depending on the available types, boolean operators, arithmetics operators, bitwise operators, or more complex data operators (arrays, trees, graphs, lists)
- memory wise: access to the value or the function of the pointer address
- etc …
either the Galois connection is implementable. We can define a best transformer for each $op_E$.

$$op_{A_b}(a_1, \ldots a_n) = \alpha (op_E (\gamma(a_1), \ldots, \gamma(a_n)))$$

It is sound versus the Galois connection:

$$\forall c_1, \ldots, c_n \in E, a_1, \ldots a_n \in A$$

$$\forall i \in [1, n], c_i \sqsubseteq_E \gamma(a_i) \implies op_E(c_1, \ldots, c_n) \sqsubseteq_E \gamma (op_{A_b}(a_1, \ldots, a_n))$$
either the Galois connection is implementable. We can define a best transformer for each \( \text{op}_E \).

\[
\text{op}_{A_b}(a_1, \ldots a_n) = \alpha (\text{op}_E (\gamma(a_1), \ldots, \gamma(a_n)))
\]

It is sound versus the Galois connection:

\[
\forall c_1, \ldots, c_n \in E, a_1, \ldots a_n \in A
\]

\[
\forall i \in [1, n], c_i \sqsubseteq_E \gamma(a_i) \implies \text{op}_E(c_1, \ldots, c_n) \sqsubseteq_E \gamma(\text{op}_{A_b}(a_1, \ldots, a_n))
\]

or it is not: we have to produce some \( \text{op}_A \) satisfying the soundness condition.

\[
\text{e.g. } \text{op}_A(a_1, \ldots, a_n) = \top_A \text{ is sound.}
\]
WHAT DO WE HAVE?

- A set of abstract domains provided by APRON
  - environment with intervals $x \mapsto [a, b], y \mapsto [c, d]$
  - linear relations among variables (loose/strict polyhedra, octagons)
  - associated concretization function $\gamma$ mapping abstract value to predicate of state variables in FOL: $\gamma(a)[x]$

- An axiomatisation of the system semantics $(\Sigma, I, \rightsquigarrow_T)$ expressed in FOL (targeting SMT)
  
  $$I[x] \quad T[x, y]$$

- An abstraction function from states to abstract elements:
  $$\alpha_Q : \Sigma \rightarrow A$$
What do we want: generate automatically an abstract transformer for $\textit{op}_E$:
A sound function $\textit{op}_A : A \rightarrow A$ based on
- the concretization function $\gamma : A \rightarrow E$
- the concrete operator $\textit{op}_E : E \rightarrow E$
The abstract transformer $g_A$ maps an abstract state $a$ to a bigger element describing more reachable states.

**Input:** $a \in A$

\[
F[\vec{x}, \vec{y}] := \gamma(a)[\vec{x}] \land T[\vec{x}, \vec{y}] \land \neg \gamma(a)[\vec{y}]
\]

**if** $F$ **is unsatisfiable** **then**

**return** $a$

**else**

let $\vec{v}, \vec{u}$ two states satisfying $F[\vec{x}, \vec{y}]$

**return** $a \sqcup_A \alpha_Q(\vec{u})$
\[ \langle E, \sqsubseteq_E \rangle \quad \text{and} \quad \langle A, \sqsubseteq_A \rangle \]

\[ \gamma(a) \sqsubseteq_E [A_{\vec{u}}] \]

\[ \gamma(a) \quad [A_{\vec{u}}] \]

\[ g_{A_{\vec{u}}} \]

\[ \alpha \gamma \circ g \circ \gamma(a) \]

\[ \sqsubseteq_A \]

\[ \sqsubseteq_E \]
Soundness
\[\langle E, \sqsubseteq_E \rangle \quad \langle A, \sqsubseteq_A \rangle\]
The fixpoint computation starts from an abstract of initial states.

\[ I_A := \perp \]

while \((I[x] \land \neg \gamma(I_A)[x] \text{ is satisfiable})\) do

let \(\bar{v}\) be a state satisfying \(I \land \neg \gamma(I_A)\)

\[ I_A := I_A \uplus_A \alpha_Q(\bar{v}) \]

return \(I_A\)
The tool takes a Lustre model and generates numerical invariants

- uses all domains of APRON
- uses Kind front-end to parse Lustre and obtain the axiomatisation in SMT
- is parametric wrt the iteration strategies and widening thresholds
- is integrated with Kind to generate invariants but can be runned independantly
- open-source, written in OCaml
Kind-AI cont’d

Kind-AI can be parametrized by

- packing primitives: \((oct : x z)\) \((poly : x y z)\)
- partitioning primitives: \(\{expr_1; expr_2 : packs\}\)

Provided models will be injected in all partitions they satisfy

\[ model \models \neg expr_1 \land expr_2 \]

\[ \implies \text{model is injected in partitions} \ \neg expr_1 \land expr_2. \]

Could also handle partitions over (small) finite range: \(\{x : ()\}\) for \(x\) bounded.
Example

```plaintext
node parallel_counters (a, b, c:bool) returns (x, y: int; obs:bool);  
var n1, n2:int;  
let  
n1 = 10000;  
n2 = 5000;  
x = 0 -> if (b or c) then 0 else  
  if a and (pre x) < n1 then (pre x) + 1 else pre x;  
y = 0 -> if c then 0 else  
  if a and (pre y) < n2 then (pre y) + 1 else pre y;  
obs = (x = n1) implies (y = n2);  
```
Example cont’d
Example cont’d
Example cont’d
Example cont’d

Graph showing a shaded triangle with vertices at (0,0), (1,0), and (0,1).
EXAMPLE CONT’D
EXAMPLE CONT’D
Example cont’d
Example cont’d
EXAMPLE CONT’D
EXAMPLE CONT’D
Example cont’d
EXAMPLE CONT’D
Example cont’d
Using a multiproperty technique for induction, concretization is expressed as a conjunction of identified sub-formula:

$$\gamma(a) = P_1 \land \ldots \land P_n$$

At each iteration of the fixpoint computation, we identify stable subparts, ie. invariants.

Example: $x \in [a, b]$ is concretized to $x \geq a \land x \leq b$. For increasing values of $x$, $x \geq a$ can be produced before the fixpoint is reached.

Kahsai, Garoche, Tinelli and Whalen. Incremental verification with mode variable invariants in state machines
At the fourth iteration, the following properties are proven:

- \( x \geq 0 \)
- \( y \geq 0 \)
- \( y < n_2 \implies x \leq y \)
A generic approach for synthesizing abstract interpreters
- needs the encoding of the transition systems in logic with entailment
- and abstract domains that can be concretized to this logic
Instanciation on Lustre models analysis
- APRON domains
- Kind k-induction Lustre axiomatization
Generates a flow of (guaranteed) invariants before reaching the final fixed point.