Interpolant tree automata and their application in Horn clause verification

Bishoksan Kafle and John P. Gallagher

Roskilde University, Denmark

VPT'16 Eindhoven, 2/4/2016

Horn clause verification problem

Constrained Horn clause (CHC)

- $p(X) \leftarrow \phi \wedge p_1(X_1), \ldots, p_k(X_k)$ (encodes program's behavior);
- $\texttt{false} \leftarrow \phi \wedge p_1(X_1), \ldots, p_k(X_k)$ (integrity constraint, encodes program's property) where `false` is interpreted as false

CHC verification problem

- find a model of a set of CHCs P.
- a program is safe if it has a model, unsafe if it has no model.

Running example: Fibonacci function encoded as Horn clauses

```
c1. fib(A, B):- A>=0,  A=<1, B=1.
c2. fib(A, B) :- A > 1, A2 = A - 2, fib(A2, B2),
           A1 = A - 1, fib(A1, B1), B = B1 + B2.
c3. false:- A>5, fib(A,B), B<A.
```

We need to show

- there is no feasible derivation of `false` in Fibonacci or
- `false` $\notin M[\![\mathit{Fibonacci}]\!]$ (minimal model of *Fibonacci*)

Formulation 1: deductive or proof based

P has a model if and only if $P \not\models \mathit{false}$.

Techniques: trace abstraction refinement [Heizmann et al. 2009, Wang et al. 2015]

Formulation 2: model based

P has a model if and only if $\mathit{false} \notin M[\![P]\!]$ (minimal model of P).

Techniques: Abstract interpretation [Cousot and Cousot 1977]

Horn clause and Finite tree automata (FTA)

```
c1. fib(A, B):- A>=0,  A=<1, B=1.
c2. fib(A, B) :- A > 1, A2 = A - 2, fib(A2, B2),
           A1 = A - 1, fib(A1, B1), B = B1 + B2.
c3. false:- A>5, fib(A,B), B<A.
```

Example (Trace FTA)

$\mathcal{A}_P = (Q, Q_f, \Sigma, \Delta)$ where:

$$\begin{array}{lcl} Q & = & \{\mathtt{fib}, \mathtt{false}\} \\ Q_f & = & \{\mathtt{false}\} \\ \Sigma & = & \{c_1, c_2, c_3\} \\ \Delta & = & \{c_1 \to \mathtt{fib},\ c_2(\mathtt{fib}, \mathtt{fib}) \to \mathtt{fib}, \\ & & c_3(\mathtt{fib}) \to \mathtt{false}\} \end{array}$$

The elements of $\mathcal{L}(\mathcal{A}_P)$ are called trace-terms or trace-trees or simply traces for P.

We can also generate Horn clauses from FTA.

Our previous approach

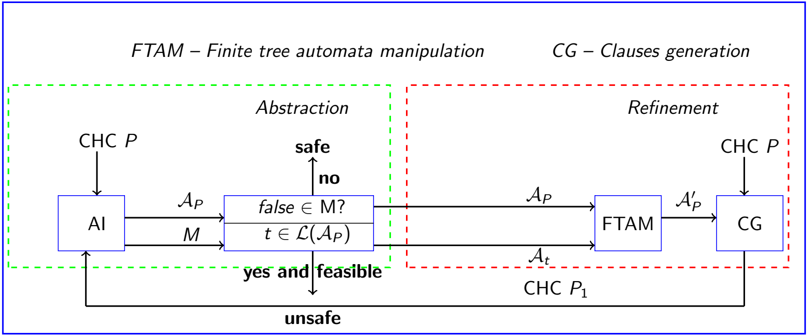


Figure : *Abstraction-refinement scheme in Horn clause verification. M is an approximation produced as a result of abstract interpretation.* $\mathcal{A}'_P$ *recognizes all traces in* $\mathcal{L}(\mathcal{A}_P) \setminus \mathcal{L}(\mathcal{A}_t)$.

Trace abstraction refinement

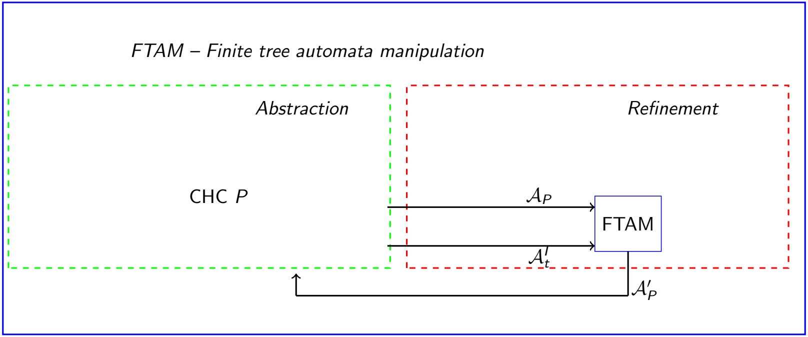


Figure : *Trace abstraction refinement scheme in Horn clause verification. $\mathcal{A}'_P$ recognizes all traces in $\mathcal{L}(\mathcal{A}_P) \setminus \mathcal{L}(\mathcal{A}_t^I)$.*

Our previous approach and trace abstraction-refinement

- both abstract interpretation and trace (counterexample) generalisation play a crucial role in verification
- in this sense, our approaches miss the aspect of each others.

Our contribution is a combination of these techniques.

Our approach

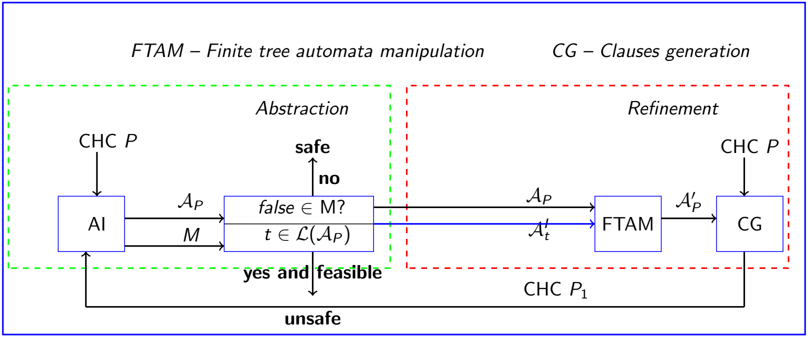
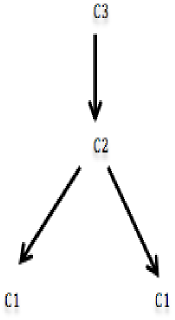
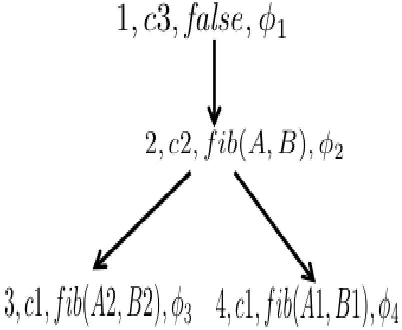


Figure : The combination: $\mathcal{A}'_P$ now recognizes all traces in $\mathcal{L}(\mathcal{A}_P) \setminus \mathcal{L}(\mathcal{A}^I_t)$.

Trace tree and AND-Tree

c3(c2(c1,c1))





$\phi_1 \equiv \mathtt{A} > 5 \wedge \mathtt{B} < \mathtt{A}; \phi_2 \equiv \mathtt{A} > 1 \wedge \mathtt{A2} = \mathtt{A} - 2 \wedge \mathtt{A1} = \mathtt{A} - 1 \wedge \mathtt{B} = \mathtt{B1} + \mathtt{B2};$
$\phi_3 \equiv A2 \geq 0 \wedge \mathtt{A2} \leq 1 \wedge \mathtt{B2} = 1; \phi_4 \equiv \mathtt{A1} \geq 0 \wedge \mathtt{A1} \leq 1 \wedge \mathtt{B1} = 1.$

AND-Tree is feasible if its constraints are satisfiable.

Construction of interpolant tree automata

Definition (Interpolant)

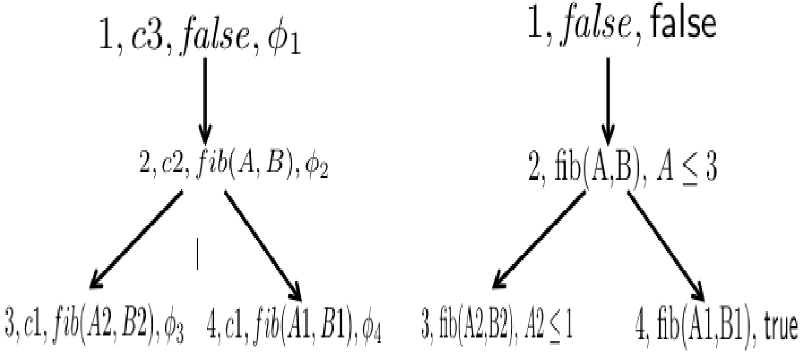
Given two formulas ϕ_1, ϕ_2 such that $\phi_1 \wedge \phi_2$ is unsatisfiable, a (Craig) interpolant is a formula I with

1. $\phi_1 \rightarrow I$;
2. $I \wedge \phi_2 \rightarrow \text{false}$; and
3. $\text{vars}(I) \subseteq \text{vars}(\phi_1) \cap \text{vars}(\phi_2)$.

Example (Interpolant example)

Let $\phi_1 \equiv A2 \leq 1 \wedge A > 1 \wedge A2 = A - 2 \wedge A1 = A - 1 \wedge B = B1 + B2$
and $\phi_2 \equiv A > 5 \wedge B < A$
such that $\phi_1 \wedge \phi_2$ is unsatisfiable.
$I \equiv A \leq 3$ is an interpolant.

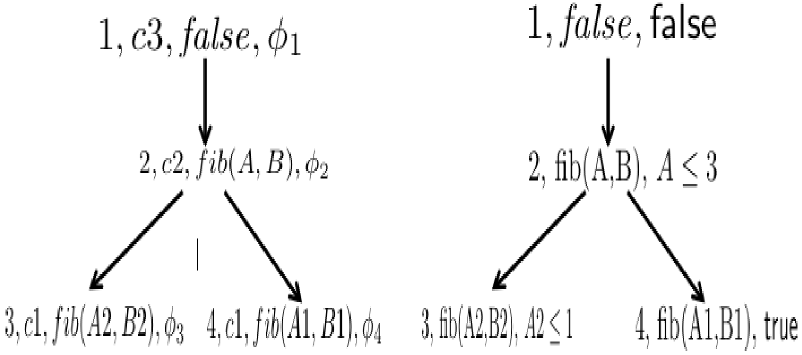
AND-Tree and its tree interpolant



Let I_j represents an interpolant of the node j. Then we have: $I_1 \equiv$ false; $I_4 \equiv I(\phi_4, \phi_3 \wedge \phi_1 \wedge \phi_2)$; $I_3 \equiv I(\phi_3, \phi_1 \wedge \phi_2 \wedge I_4)$; $I_2 \equiv I(I_3 \wedge I_4 \wedge \phi_2, \phi_1)$.

Interpolant automata (I)

Interpolant automaton for $c_3(c_2(c_1, c_1))$



$$
\begin{array}{lcl}
Q & = & \{\texttt{fib}^2, \texttt{fib}^3, \texttt{fib}^4, \texttt{false}\} \\
Q_f & = & \{\texttt{false}\} \\
\Sigma & = & \{c_1, c_2, c_3\} \quad (\textit{all function symbols of } \mathrm{P})
\end{array}
$$

mapping from each node in the tree to the original predicate

Interpolant automata (II)

Δ are derived

- Given $c : p(X) \leftarrow \phi, p_1(X_1), \ldots, p_k(X_k) \in P$
- if $TI(p^j)(X) \leftarrow \phi, TI(p_1^{j_1})(X_1), \ldots, TI(p_k^{j_k})(X_k)$ then add $c(p_1^{j_1}, \ldots, p_k^{j_k}) \rightarrow p^j$ to Δ

For example Δ contains $c_2(\mathtt{fib}^3, \mathtt{fib}^2) \rightarrow \mathtt{fib}^2$ because

- `c2.  fib(A, B) :- A > 1, A2 = A - 2, fib(A2, B2), A1 = A - 1, fib(A1, B1), B = B1 + B2`
- TI: fib(A,B) $\equiv$ A$\leq$3 (at node 2) and fib(A,B) $\equiv$ A$\leq$1 (at node 3)
- consider the mapping fib at any node corresponds to fib of the original program
- so the implication $A > 1, A2 = A - 2, A2 \leq 1, A1 = A - 1, A1 \leq 3, B = B1 + B2 \rightarrow A \leq 3$ holds

Experimental setting

- 68 verification problems (SVCOMP'15, repository of Horn clause problems [1])
- computer: OS X, 2.3 GHz Intel, 8 GB RAM, timeout: 5mins
- implementation: Ciao interfaced with PPL library and Yices SMT solver and FTA library
- our current tool: RAHIT (Refinement of abstraction in Horn clauses with Interpolant Tree Automata)
- comparison:
  - RAHFT (our previous approach) : the effect of removing a set of traces rather than the single one
  - TAR (trace abstraction refinement, Wang et al. 2015): the effect of polyhedral abstraction

[1] https://github.com/sosy-lab/sv-benchmarks/tree/master/clauses/LIA/Eldarica

Experiments (I)

	Time RAHFT	#Itr. RAHFT	Time RAHIT	#Itr. RAHIT
avg.	10.55	2.33	11.40	2.08
solved	82%		89%	

- RAHIT more effective than RAHFT : more tasks solved with fewer iterations (result of trace generalisation) but takes longer time (the cost of computing interpolant automaton)

Experiments (II)

	Time RAHIT	#Itr. RAHIT	Time TAR	#Itr. TAR
avg.	8.78	0.93	9.52	38.64
solved	86%		73%	

- RAHIT more effective than TAR: solves more tasks, in few iterations, less time (emphasizes the power of abstract interpretation)

Conclusion

- The proposed combination shows improvements over the previous approaches
- Next, use SMT solvers for computing interpolant

The end!

Thanks for your attention!