Tree automata-based refinement with application to Horn clause verification

Bishoksan Kafle John Gallagher

Roskilde University

VMCAI, Mumbai, India 12/14-01-2015

Introduction

Goal: find a model of a set of Horn clauses (program)
Desired characteristics :

- scalable and terminating way of computing a model of Horn clauses (approximation)
- if we don't know if there is a model then refine the program (recover precision)

For this we use the theory of abstract interpretation (abstraction) and the theory of finite tree automata (refinement)
Key contributions:

- interaction between abstract interpretation and finite tree automata
- refinement using finite tree automata
- feasibility in practice (experiments)

Summary of our approach

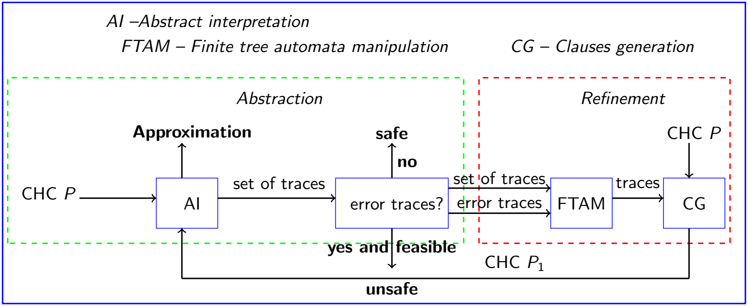


Figure : *Abstraction-refinement in Horn clause verification*

Overview

Overview

Definitions

Constrained Horn Clause (CHC)

A predicate logic formula, $p(X) \leftarrow \phi \wedge p_1(X_1), \ldots, p_k(X_k)$

- ϕ - a conjunction of constraints wrt some background theory,
- X_i, X are (possibly empty) vectors of distinct variables,
- $p_1, \ldots, p_k, p$ are predicate symbols,
- $p(X)$ is the head of the clause and
- $\phi \wedge p_1(X_1) \wedge \ldots \wedge p_k(X_k)$ is the body.

Integrity constraints

$\text{false} \leftarrow \phi \wedge p_1(X_1), \ldots, p_k(X_k)$.

Horn clause verification problem

CHC verification problem

- given a set of CHCs P (including integrity constraints encoding safety properties),
- does P have a model?

Results from CLP:

Lemma 1

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

Lemma 2

P has a model if and only if false $\notin M[\![P]\!]$ (minimum model of P).

Running example

Example: McCarthy91 function

```
c1. mc91(A,B) :- A > 100, B = A-10.
c2. mc91(A,B) :- A =< 100, C = A+11, mc91(C,D), mc91(D,B).

c3. false :- A =< 100,  B > 91, mc91(A,B).
c4. false :- A =< 100, B =< 90, mc91(A,B).
```

The goal is to show $\mathsf{false} \not\in M[\![McCarthy91]\!]$.

Notation: c_i is a clause identifier (ranked function symbol)

Overview

Finite tree automata

Definition (Finite tree automaton (FTA))

An FTA $\mathcal{A}$ is a tuple (Q, Q_f, Σ, Δ), where Q is a finite set of states, $Q_f \subseteq Q$ is a set of final states, Σ is a set of function symbols, and Δ is a set of transitions. We assume that Q and Σ are disjoint.

Definition (Deterministic FTA (DFTA))

An FTA (Q, Q_f, Σ, Δ) is called bottom-up deterministic iff Δ contains no two transitions with the same left hand side.

Trace automata for CHCs

Given a set of CHCs P and a set Σ of ranked function symbols,

$$\mathrm{id}_P : P \to \Sigma$$

(assignment of function symbols to clauses).

Definition (Trace FTA for a set of CHCs)

Define the trace FTA for P as $\mathcal{A}_P = (Q, Q_f, \Sigma, \Delta)$ where

- Q is the set of predicate symbols of P;
- $Q_f \subseteq Q$ is the set of predicate symbols occurring in the heads of clauses of P;
- Σ is a set of function symbols;
- $\Delta = \{c_j(p_1, \ldots, p_k) \to p \mid \text{where } c_j \in \Sigma,\ p(X) \leftarrow \phi, p_1(X_1), \ldots, p_k(X_k) \in P,\ c_j = \mathrm{id}_P(p(X) \leftarrow \phi, p_1(X_1), \ldots, p_k(X_k))\}$.

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

Trace FTA for McCarthy91

```
c1. mc91(A,B) :- A > 100, B = A-10.
c2. mc91(A,B) :- A =< 100, C = A+11, mc91(C,D), mc91(D,B).
c3. false :- A =< 100,  B > 91, mc91(A,B).
c4. false :- A =< 100, B =< 90, mc91(A,B).
```

Let P be the above set of CHCs. Let id_P map the clauses to $c_1, \ldots, c_4$ respectively. Then $\mathcal{A}_P = (Q, Q_f, \Sigma, \Delta)$ where:

$$
\begin{array}{lcl}
Q & = & \{\mathtt{mc91}, \mathtt{false}\} \\
Q_f & = & \{\mathtt{mc91}, \mathtt{false}\} \\
\Sigma & = & \{c_1, c_2, c_3, c_4\} \\
\Delta & = & \{c_1 \rightarrow \mathtt{mc91},\ c_2(\mathtt{mc91}, \mathtt{mc91}) \rightarrow \mathtt{mc91}, \\
 & & c_3(\mathtt{mc91}) \rightarrow \mathtt{false},\ c_4(\mathtt{mc91}) \rightarrow \mathtt{false}\}
\end{array}
$$

Figure : Example CHCs McCarthy91 and its trace automata

Generation of CHCs from a trace FTA

It is also possible to generate a set of CHCs from an FTA with the following properties.

Proposition (Correctness)

Given P and an FTA $\mathcal{A}$ whose signature is the same as that of $\mathcal{A}_P$. Let P' be the set of clauses generated from $\mathcal{A}$ and P. Then $\mathcal{L}(\mathcal{A}_{P'}) = \mathcal{L}(\mathcal{A})$.

Example: From FTA to CHC

```
c1. mc91(A,B) :- A > 100, B = A-10.
c2. mc91(A,B) :- A =< 100, C = A+11, mc91(C,D), mc91(D,B).
c3. false :- A =< 100,  B > 91, mc91(A,B).
c4. false :- A =< 100, B =< 90, mc91(A,B).
```

The set of states is { `[false]`, `[mc91]`, `[mc91,e1]` }.

```
c1 -> [mc91, e1].
c2([mc91, e1],[mc91, e1]) -> [mc91].
c3([mc91]) -> [false].
c4([mc91, e1]) -> [false].
```

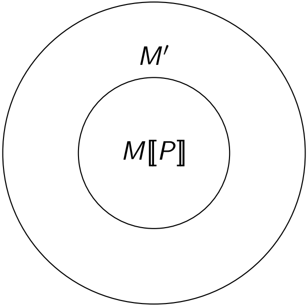
$\rho = \{[\mathtt{false}] \mapsto \mathtt{false}, [\mathtt{mc91}] \mapsto \mathtt{mc91}, [\mathtt{mc91}, \mathtt{e1}] \mapsto \mathtt{mc91\_1}\}$.

```
c1: mc91_1(A,B) :-  A>100, B=A-10.
c2: mc91(A,B) :-  A=<100,  C=A+11, mc91_1(C,D),  mc91_1(D,B).
c3: false :- A =< 100, B > 91, mc91(A,B).
c4: false :- A =< 100, B =< 90, mc91(A,B).
```

Overview

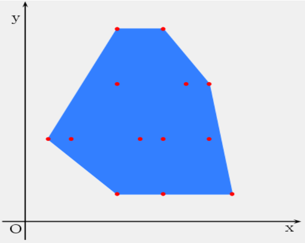
Proof by over-approximation of the minimal model

- There exists a minimal model, $M[\![P]\!]$, wrt the subset ordering,
- $M[\![P]\!]$ is equivalent to the set of atomic consequences of P (model vs. proof)
- It is sufficient to find a set of constrained facts M' such that $M[\![P]\!] \subseteq M'$, where false $\notin M'$.



Convex polyhedra approximation (CPA)

- a program analysis technique based on abstract interpretation.
- when applied to P it constructs an over-approximation M' of the minimal model of P, where M' contains at most one constrained fact $p(X) \leftarrow \mathcal{C}$ for each predicate p.
- where the constraint $\mathcal{C}$ is a conjunction of linear inequalities, representing a convex polyhedron.



$$\begin{cases} 6x + y \leq 111 \\ 3x + 2y \leq 78 \\ x + y \geq 11 \\ 2x - y \geq -5 \\ y \geq 3 \\ y \leq 21 \end{cases}$$

Example Polyhedral Abstraction

Constrained facts

```
mc91(A,B) :- [B>90, B>=A-10].
false :- [].
```

- Since there is a constrained fact for false in the overaproximation, this overapproximation is too imprecise.
- We produce a derivation for false as a trace term c3(c1).
- Checking SAT(c3(c1)) returns UNSAT $\implies$ infeasible trace (spurious counterexample).

Overview

Clause refinement algorithm

Input: A set of Horn clauses P and an infeasible trace t
Output: A set of Horn clauses P'

1. construct the trace FTA $\mathcal{A}_P$;
2. construct an FTA $\mathcal{A}_t$ such that $\mathcal{L}(\mathcal{A}_t) = \{t\}$;
3. compute the difference FTA $\mathcal{A}_P \setminus \mathcal{A}_t$;
4. generate P' from $\mathcal{A}_P \setminus \mathcal{A}_t$ and P
5. **return** P';

Proposition (Progress)

The same counterexample does not arise again in any future approximations.

But ...

Definition (Union of FTAs)

Let $\mathcal{A}^1, \mathcal{A}^2$ be FTAs $(Q^1, Q^1_f, \Sigma, \Delta^1)$ and $(Q^2, Q^2_f, \Sigma, \Delta^2)$ respectively. Then $\mathcal{A}^1 \cup \mathcal{A}^2 = (Q^1 \cup Q^2, Q^1_f \cup Q^2_f, \Sigma, \Delta^1 \cup \Delta^2)$, and we have $\mathcal{L}(\mathcal{A}^1 \cup \mathcal{A}^2) = \mathcal{L}(\mathcal{A}^1) \cup \mathcal{L}(\mathcal{A}^2)$.

Definition (Construction of difference of FTAs)

Let $\mathcal{A}^1, \mathcal{A}^2$ be FTAs $(Q^1, Q^1_f, \Sigma, \Delta^1)$ and $(Q^2, Q^2_f, \Sigma, \Delta^2)$ respectively. Let $(\mathcal{Q}', \mathcal{Q}'_f, \Sigma, \Delta')$ be the determinisation of $\mathcal{A}^1 \cup \mathcal{A}^2$. Let $\mathcal{Q}^2 = \{Q' \in \mathcal{Q}' \mid Q' \cap Q^2_f \neq \emptyset\}$. Then $\mathcal{A}^1 \setminus \mathcal{A}^2 = (\mathcal{Q}', \mathcal{Q}'_f \setminus \mathcal{Q}^2, \Sigma, \Delta')$.

- The difference construction needs determinising an FTA,
- Thanks to a practical algorithm (Gallagher et al. TR-2014) which made this operation possible, where they use compact representation for the set of transitions (product form)

Further refinement: FTA state splitting

- split states representing predicates where convex hull operations have lost precision.
- inspired by predicate splitting from CLP
- the effect is to delay join (widen) operations for precision gain
- splitting the states preserves the set of traces

Overview

Settings

benchmarks

- 216 (179 safe / 37 unsafe) problems
- repository of SV benchmarks [a] and
- other sources including Gupta et al. (2009) [Invgen], Jaffar et al. (2012) [TRACER], De Angelis et al. (2014) [VeriMap] etc.

[a] https://svn.sosy-lab.org/software/sv-benchmarks/trunk/clauses/

environment

- Implementation: 32-bit Ciao Prolog [a] with Parma Polyhedra Library (Bagnara et al. (2008))
- Computer: Intel(R) X5355 @ 2.66GHz and total memory of 6 GB. Debian 5 (64 bit) - OS,
- we set 5 minutes of timeout for each experiment.

[a] http://ciao-lang.org/

Experimental results

	CPA	CPA+R	CPA+R+Split	QARMC
solved (safe/unsafe)	160 (142/18)	182 (160/22)	195 (164/31)	178 (141/37)
unknown/ timeout	49/7	-/34	-/22	-/38
average time (secs.)	5.98	51.66	50.08	59.1
% solved	74	84.25	90.27	82.4

Figure : Experimental results on 216 (179 safe / 37 unsafe) CHC verification problems with a timeout of five minutes

Experimental results

	CPA	CPA+R	CPA+R+Split	QARMC
solved (safe/unsafe)	160 (142/18)	182 (160/22)	195 (164/31)	178 (141/37)
unknown/ timeout	49/7	-/34	-/22	-/38
average time (secs.)	5.98	51.66	50.08	59.1
% solved	74	84.25	90.27	82.4

Convex polyhedral anaysis is poweful on its own solving 74% of the problems.

Experimental results

	CPA	CPA+R	CPA+R+Split	QARMC
solved (safe/unsafe)	160 (142/18)	182 (160/22)	195 (164/31)	178 (141/37)
unknown/ timeout	49/7	-/34	-/22	-/38
average time (secs.)	5.98	51.66	50.08	59.1
% solved	74	84.25	90.27	82.4

22 more problems can be solved by refining the program with the increase in time.

Experimental results

	CPA	CPA+R	CPA+R+Split	QARMC
solved (safe/unsafe)	160 (142/18)	182 (160/22)	195 (164/31)	178 (141/37)
unknown/ timeout	49/7	-/34	-/22	-/38
average time (secs.)	5.98	51.66	50.08	59.1
% solved	74	84.25	90.27	82.4

splitting increases the precision of analysis.

Experimental results

	CPA	CPA+R	CPA+R+Split	QARMC
solved (safe/unsafe)	160 (142/18)	182 (160/22)	195 (164/31)	178 (141/37)
unknown/ timeout	49/7	-/34	-/22	-/38
average time (secs.)	5.98	51.66	50.08	59.1
% solved	74	84.25	90.27	82.4

compares favourably with QARMC (Grebenshchikov et al. PLDI12).

Overview

Summary and Future Work

Conclusion:

- we presented abstraction (using abstraction interpretation) refinement (using finite tree automata) in Horn clauses;
- our refinement phase is independent of the abstract domain used;
- the practicality of our approach was demonstrated on a set of Horn clause verification problems;

Future work:

- investigate the elimination of a larger set of infeasible traces in each refinement step, possibly by
  - generalising a trace using interpolation or
  - discovering a set of infeasible traces.

The end!

Thanks for your attention!