Solving non-linear Horn clauses using a linear Horn clause solver

Bishoksan Kafle, John Gallagher and Pierre Ganty

Roskilde University, Denmark and IMDEA Software Institute, Spain

HCVS'16 Eindhoven
03/04/2016

Motivation

Number of solvers based on Constrained Horn Clauses (CHCs) are available:

After fixing a constraint theory, the (Horn clause) solvers are:

- linear e.g., VeriMap, Sally etc.
- non-linear e.g., RAHFT, SeaHorn, QARMC, ELDARICA, Z3 etc.

since the underlying engine of linear solver can handle only linear clauses which restricts, in principle, their applicability

Can we solve non-linear CHCs using a linear Horn clause solver?

Notation: solver = Horn clause solver, linear solver = Horn clause solver for linear Horn clauses

Is it possible?

Yes, by interleaving program transformation (Horn clause linearisation) with linear Horn solving in an incremental manner to handle non-linear clauses.

Example: CHCs defining the Fibonacci function (*Fib*)

```
c1. fib(A, B) :-  A>=0,  A=<1, B=A.
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.
```

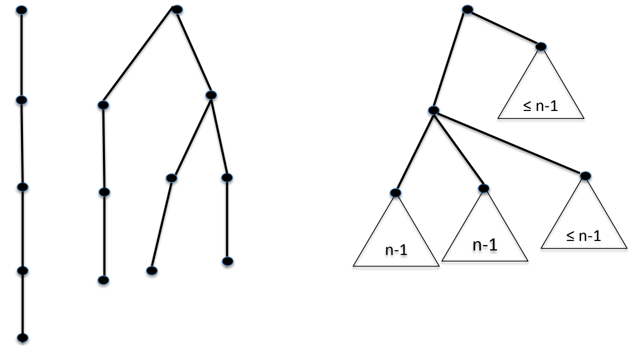
c1 and c3 are linear clauses, c2 non-linear

The Horn clause verification

to show that there is no successful derivation of *false*.

Program transformation (I)

is based on the idea of tree dimension of Horn clause derivations



dimension 0 | dimension 1 | . . . | dimension n

The dimension of tree is a measure of its (non)-linearity

Program transformation (II)

- Given a set of clauses (program) P, the notion of *tree dimension* allows us to derive a program $P^{\leq k}$ (P at most k or simply k-dim program) whose derivations trees have dimension $\leq k (k \geq 0)$

The Horn clause verification problem based on tree dimension

- show that there is no successful derivation of *false* – of any dimension.

- It is known that $P^{\leq k}$ is linearisable [Afrati et al., 2003].

this allows us to generate programs for increasing value of k, linearise and solve them.

Example: dimension bounded program

dimension of Fib's derivation trees depends on the input number.

```
c1. fib(A, B) :-  A>=0,  A=<1, B=A.
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.
```

$Fib^{\leq 0}$ (linear)

```
fib(0)(A,B) :- A>=0, A=<1, B=A.
false(0) :- A>5, B<A, fib(0)(A,B).
false[0] :- false(0).
fib[0](A,B) :- fib(0)(A,B).
```

the atom $p(k)(X)$ means any derivation tree rooted at $p(0)(X)$ will have tree dimension k

$p[k](X)$ – tree dimension $\leq k$

$Fib^{\leq 1}$ (1-linear)

```
fib(0)(A,B) :- B=A,  A=<1, A>=0.
false(0) :-  B<A,  A>5, fib(0)(A,B).
false[0] :-  false(0).
fib[0](A,B) :- fib(0)(A,B).

fib(1)(A,B) :-  B=F+D, C=A-2,
            E=A-1,  A>1, fib[0](E,F), fib(1)(C,D).
fib(1)(A,B) :-  B=F+D, C=A-2, E=A-1,
            A>1, fib[0](C,D), fib(1)(E,F).
fib(1)(A,B) :-  B=F+D,  C=A-2,  E=A-1,
            A>1, fib(0)(C,D), fib(0)(E,F).
.
.
.
```

All clauses of $Fib^{\leq 0}$ are in $Fib^{\leq 1}$

Construction and consequence

by construction all clauses of $P^{\leq k}$ are included in $P^{\leq k+1}$ $(k \geq 0)$

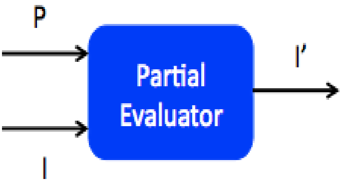
As a result

- it provides a basis for iterative strategy for dimension bounded programs.
- reuse the solution obtained for lower dimension to linearise/solve clauses of higher dimension

Linearisation (I)

based on partial evaluation (PE). PE is a source-source program transformation.

- P: non-linear clauses, I: an interpreter (linear in our case, written in some language L (as Horn clauses))
- I' is a specialised interpreter for P, which can be regarded as the transformation of P.



- same as predicate tuppling (Invited talk).

Reuse of solution and Linearisation (I)

Assume that the following is the solution for $Fib^{\leq 0}$

```
fib(0)(A,B) :- B=A,  A=<1, A>=0.
fib[0](A,B) :- B=A,  A=<1, A>=0.
false(0):- FALSE.
false[0]:- FALSE.
```

Given $Fib^{\leq 1}$

```
fib(0)(A,B) :- B=A,  A=<1, A>=0.
false(0) :-  B<A,  A>5, fib(0)(A,B).
false[0] :-  false(0).
fib[0](A,B) :- fib(0)(A,B).

fib(1)(A,B) :-  B=F+D, C=A-2,
            E=A-1,  A>1, fib[0](E,F), fib(1)(C,D).
fib(1)(A,B) :-  B=F+D, C=A-2, E=A-1,
            A>1, fib[0](C,D), fib(1)(E,F).
fib(1)(A,B) :-  B=F+D,  C=A-2,  E=A-1,
            A>1, fib(0)(C,D), fib(0)(E,F).
```

Reuse of solution and Linearisation (II)

$Fib^{\leq 1}$ after solution reuse

```
fib(0)(A,B) :- B=A,  A=<1, A>=0.
fib[0](A,B) :- B=A,  A=<1, A>=0.

fib(1)(A,B) :-  B=F+D, C=A-2,
            E=A-1,  A>1, fib[0](E,F), fib(1)(C,D).
fib(1)(A,B) :-  B=F+D, C=A-2, E=A-1,
            A>1, fib[0](C,D), fib(1)(E,F).
fib(1)(A,B) :-  B=F+D,  C=A-2,  E=A-1,
            A>1, fib(0)(C,D), fib(0)(E,F).
.
.
.
```

Then we can linearse this program.

Assumption about a linear solver

1. linear solver is a black box and is sound
2. capable of producing a tuple Status $\times$ Result where Status $\in$ {safe or unsafe} and Result $\in$ {solution, counterexample}

A solution for P is a set of constrained facts of the form: $p(X) \leftarrow \phi$ for each predicate p occurring in P

Our approach

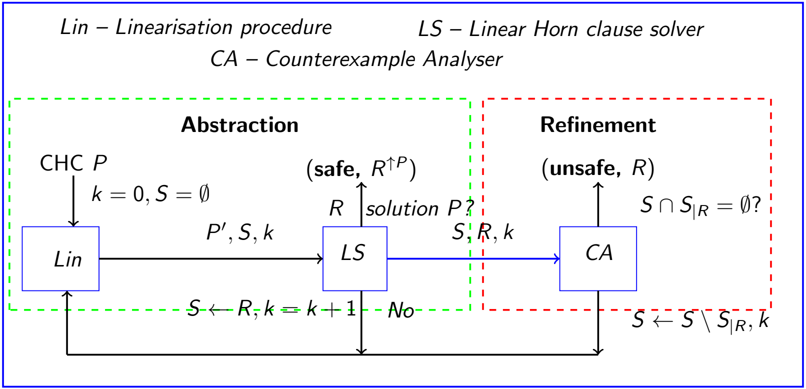


Figure : *Abstraction-refinement scheme for solving non-linear Horn clauses using a linear solver. P' is a linearised version of P's k-dim program. $S_{|R}$ is a set of constrained facts from S appearing in a counterexample.*

Example: counterexample using approximate solution

```
c1. false:- X=0, p(X).
c2. false:- q(X).
c3. p(X):- X>0.
c4. q(X):-X=0.
```

Suppose $S = \{p(X) : -TRUE\}$ for the predicate $p(X)$. Using this solution, we get

```
c1. false:- X=0, p(X).
c2. false:- q(X).
c3. p(X):- TRUE. (approximate solution)
c4. q(X):-X=0.
```

$c_1(c_3)$ is a spurious counterexample for the original program

$c_2(c_4)$ is a real counterexample

Experimental settings

1. Linear solver: Convex polyhedral analyser (CPA) – terminates but may generate *false alarms*
2. Partial Evaluator: Logen [Leuschel et al., 2003]
3. Benchmarks: 44 problems (SV-COMP'15, QARMC, Repository of Horn clauses)
4. Tool: LHornSolver (`https://github.com/bishoksan/LHornSolver`)

Goal

1. whether solving non-linear Horn clauses can be done using a solver for linear Horn clauses?
2. the relation between the solvability of a program with tree dimension
3. comparison with tools for non-linear Horn clauses

Experimental Results

- 61% of the problems are solved
- we found that the solution of $P^{\leq k}$ (for $k = 1, 2$) becomes a solution of P or counterexample was found

The results on this set of benchmarks show that

- it is feasible to solve non-linear Horn clauses using a linear solver and
- the solvability of a problem is shallow wrt. tree dimension of its derivations.

Comparison with RAHFT

Comparison with RAHFT (whose underlying engine is also CPA), Horn solver for non-linear clauses

- RAHFT solves all the problems unlike LHornSolver

This could mean

- linearisation strategy we use is not useful for solving non-linear Horn clauses
- the use of CPA in LHornSolver: no refinement is done when CPA produces a *false alarm*. In this case LHornSolver returns unknown

What next?

- experiment with different linearisation strategies for Horn clause
- use different linear solver within LHornSolver, which if returns returns with a solution or a counterexample wrt. the original program

The end!

Thanks for your attention!