Abstraction, specialisation and refinement in Horn clause verification

Bishoksan Kafle **John Gallagher**

Roskilde University

HCVS, Vienna, 17-07-2014

Overview

Overview

1. Horn clause verification
2. Background
3. Our approach to CHC verification
   - Query answer transformation
   - Specialisation by constraint propagation
   - Convex Polyhedral Analaysis (CPA)
4. Experimental Results
5. Conclusion and Future works

Definitions

Constrained Horn Clause (CHC)

A predicate logic formula, $H(X) \leftarrow \phi \wedge B_1(X_1), \ldots, B_k(X_k)$

- ϕ - a conjunction of constraints with respect to some background theory,
- X_i, X are (possibly empty) vectors of distinct variables,
- $B_1, \ldots, B_k, H$ are predicate symbols,
- $H(X)$ is the head of the clause and
- $\phi \wedge B_1(X_1) \wedge \ldots \wedge B_k(X_k)$ is the body.

Integrity constraints

false $\leftarrow \phi \wedge B_1(X_1), \ldots, B_k(X_k)$.
where false is always interpreted as *false*.

CHC Verification

CHC verification problem

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

CHCs verification techniques and tools

- CHC has gained interest from CLP and software verification communities

CLP

- approximation of the minimal model of a CLP program using abstract interpretation (AI)
- specialisation wrt a goal
- model preserving transformations etc.

Verification

- AI
- counter example guided abstraction refinement (CEGAR) etc.

- Tools: VeriMAP, HSF(C) , TRACER etc.

CHCs is a software verification community's terminology for CLP
From now on set of CHCs, CLP and CHC program are used interchangeably

Characteristics

	Commonalities	Characteristics	Issues
AI	derive invariants	scalable, not property guided	domain choice, false alarms
Specialisation	by transformation	model preserving, property guided	generalisation operators
CEGAR	by cEx analysis	property guided	cEx generalisation

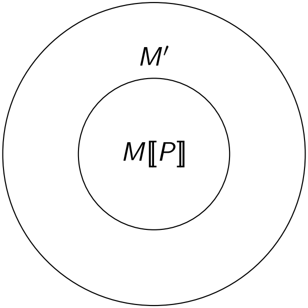
In essence, CHC verification boils down to deriving required program invariants but each of these techniques usually miss the aspect of each others

Does our paper give any answer?

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 $\notin M'$.



Proof by specialisation / Transformation

Given P_0 and an atom A, we wish to prove A is not a consequence of P_0

P_0 P_1 ...... P_k

P_k contains no clause with head A

we wish to prove A is a consequence of P_0

P_0 P_1 ...... P_k

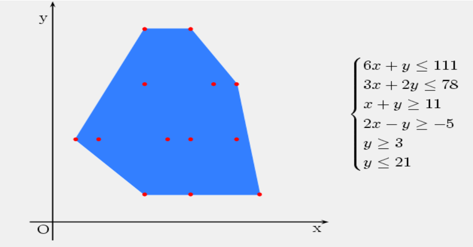
P_k contains a clause with head $A \leftarrow true$

- $P \models A$ if and only if $P' \models A$, P' is a specialisation of P.

Analysis

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.



source: http://bugseng.com/products/ppl/abstractions

Overview

Our approach

We carry out following steps in an iterative manner:

1. constraints strengthening (in the clauses),
2. abstract interpretation (over convex polyhedra) ,
3. predicate splitting
4. program refinement

Running Example

```
false :- A>0, B=0, C=0, D=0, l(B,C,D,A).

l(A,B,C,D) :- -A+D>0, A-G= -1, l_body(B,C,E,F), l(G,E,F,D).
l(A,B,C,D) :- A-D>=0, B+C-3*D>0.
l(A,B,C,D) :- A-D>=0, -B-C+3*D>0.

l_body(A,B,C,D) :- A-C= -1, B-D= -2.
l_body(A,B,C,D) :- A-C= -2, B-D= -1.
```

Query answer transformation (QA)

Simulates goal-directed computation in a goal-independent framework.

- For each predicate p in P, define two predicates p^{q} and p^{a}.

```
l(A,B,C,D) :- -A+D >0, A-G= -1, l_body(B,C,E,F), l(G,E,F,D).

l_ans(A,B,C,D) :- l_query(A,B,C,D), -A+D>0,  A-E= -1,
      l_body_ans(B,C,F,G), l_ans(E,F,G,D).

l_body_query(A,B,_,_) :- l_query(C,A,B,D), -C+D>0,
C+ -_= -1.
l_query(A,B,C,D) :- l_query(E,F,G,D),  -E+D>0,
      E-A= -1, l_body_ans(F,G,B,C).
```

- Given P and a query A, derive P_A^{qa} (QA for P wrt. A)
- $P \models A$ iff $P_A^{qa} \models A$

Specialisation by constraint propagation

Input: P and an atomic formula A,
output: a specialised set of CHCs P_A.

- Compute an over-approximation of the model of P_A^{qa}, expressed as a set of constrained facts $p^*(X) \leftarrow C$.

```
l_ans(A,B,C,D) :- 2*B-C>=0, D>0, -B+2*C>=0, -B-C+3*D> -3,
3*A-B-C=0.
```

- Replace

```
l(A,B,C,D) :- -A+D >0, A-G= -1, l_body(B,C,E,F), l(G,E,F,D).
```

  by

```
 l(A,B,C,D) :- 2*A-B>=0, -A+D>0, -A+B>=0,
 3*A-B-C=0, A-E= -1, l_body(B,C,F,G),  l(E,F,G,D).
```

- $P \models \text{false}$ iff $P_{\text{false}} \models \text{false}$.

Specialised set of CHCs Sp(P)

```
c1. false :- A>0, B=0, C=0, D=0, l(B,C,D,A).

c2. l(A,B,C,D) :- 2*A-B>=0, -A+D>0, -A+B>=0, 3*A-B-C=0,
                              A-E= -1, l_body(B,C,F,G),  l(E,F,G,D).
c3. l(A,B,C,D) :- 3*A-3*D>0, D>0,  2*A-B>=0, -3*A+3*D> -3,
                                -A+B>=0, 3*A-B-C=0.

c4. l_body(A,B,C,D) :- -A+2*B>=0, 2*A-B>=0, A-C= -1, B-D= -2.
c5. l_body(A,B,C,D) :- -A+2*B>=0, 2*A-B>=0, A-C= -2, B-D= -1.
```

CPA Result on Sp(P)

```
l_body(A,B,C,D) :- B-D>= -2, -B+D>=1, -A+2*B>=0,
                       2*A-B>=0, A+B-C-D= -3.
false :- true.
l(A,B,C,D) :- D>0, 2*A-B>=0, -A+B>=0, -3*A+3*D> -3,
             3*A-B-C=0.
```

- presence of constrained fact for *false* $\rightarrow$ *P* may not be safe
- CPA returns counter example trace c1(c3) in the form of trace term

Counterexample analysis

check trace for feasibility by collecting constraints from the clauses,

- if feasible then our analysis terminates and returns bug
- else refine Sp(P)

Interpolant

Given two constraints C_1, C_2 such that $C_1 \wedge C_2$ is unsatisfiable, an interpolant is a constraint I with (i) $C_1 \rightarrow I$, (ii) $I \wedge C_2$ is unsatisfiable and (iii) I contains variables common to both C_1 and C_2.

predicate splitting

Let I(X) be such a constraint over set of variables X, and $p(X) \leftarrow c(X)$ a constrained fact, then we split this fact into $p(X) \leftarrow c(X), I(X)$ and $p(X) \leftarrow c(X), \neg I(X)$.

Predicate splitting

- `I(A,B,C,D) = A-3*B+C+D=<0` is the interpolant computed from the trace `c1(c3)` for predicate `l(A,B,C,D)`.
- splitting `l(A,B,C,D) :- D>0,2*A-B>=0,-A+B>=0,-3*A+3*D> -3, 3*A-B-C=0.` with the interpolant produces (after constraint simplification)

```
l(A,B,C,D) :- -4*A+4*B-D>=0,D>0,-3*A+3*D> -3, 2*A-B>=0,
3*A-B-C=0.
```

- Based on these extended set of constraint facts and Sp(P) we generate a new CHC through specialization (Gallagher (1993) [Tutorial on specialisation of logic programs])

Summary of our approach

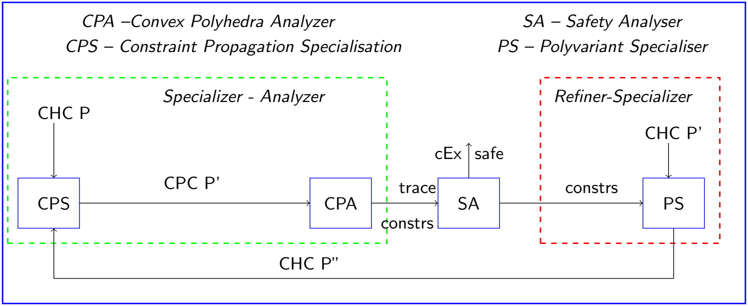


Figure : *Tool chain overview (CHC verification).*

Overview

Settings

benchmarks

- repository of SV benchmarks [a] and
- other sources including Gupta et al. (2009) [Invgen], Beyer (2013) [SV-COMP 2013], 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 having 4 processors (each @ 2.66GHz) and total memory of 6 GB. Debian 5 (64 bit) - OS,
- we set 2 minutes of timeout for each experiment.

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

Experimental results

	CPA	CPA+CPS	CPA+CPS+R
solved (safe/unsafe)	61(48/13)	160 (142/18)	181 (158/23)
unknown/ timeout	142/12	49/7	-/35
total time (secs)	1717	1293	3410
average time (secs)	7.94	5.98	18.73

Table : Experimental results on 216 (179/37) CHC verification problems, CPA - convex polyhedra analysis, CPS - specialisation, R - refinement, "-" not relevant.

Discussion

- the overall result shows that it compares favourably with other advanced verification tools like HSF(C) , VeriMAP, TRACER etc. in both time and the number of problems solved, see De Angelis et al. (2014) TACAS paper for the comparison with other tools.
- this shows the feasibility of our approach.
- problems over integers and fixed bits are sometimes challenging to us since we model programs over reals

Overview

Summary and Future Works

- presented an approach for CHC verification based on combination of several techniques
- specialisation without unfolding clauses
- experimental results on some benchmark problems prove the feasibility of our approach
- understand better the connection between program specialization and CEGAR

Questions?

Thanks for your attention!

Query answer transformation (QA)

Simulates goal-directed computation in a goal-independent framework.

Given P and an atom A, the QA for P wrt. A, denoted P^{qa}_A, contains:

Answer clauses

For each clause $H \leftarrow C, B_1, \ldots, B_n$ ($n \geq 0$) in P, P^{qa}_A contains the clause $H^a \leftarrow C, H^q, B^a_1, \ldots, B^a_n$.

Query clauses

For each clause $H \leftarrow C, B_1, \ldots, B_i, \ldots, B_n$ ($n \geq 0$) in P, P^{qa}_A contains:

$B^q_1 \leftarrow C, H^q$.

...

$B^q_i \leftarrow C, H^q, B^a_1, \ldots, B^a_{i-1}$.

...

$B^q_n \leftarrow C, H^q, B^a_1, \ldots, B^a_{n-1}$.

Goal clause

$A^q \leftarrow \text{true}$.

The procedure is as follows: the inputs are a set of CHCs P and an atomic formula A.

1. Compute a P_A^{qa}, containing predicates p^{q} and p^{a} for each predicate p in P.
2. Compute an over-approximation of the model of P_A^{qa}, expressed as a set of constrained facts $p^*(X) \leftarrow C$, where $*$ is q or a. We assume that each predicate p^* has exactly one constrained fact in the model
3. For each clause $p(X) \leftarrow \mathcal{B}$ in P, let the model of p^{a} be $p^{\mathsf{a}}(X) \leftarrow C^{\mathsf{a}}$ (where X is the same tuple of variables in $p(X)$ and $p^{\mathsf{a}}(X)$).
4. Replace the clause $p(X) \leftarrow \mathcal{B}$ in P by $p(X) \leftarrow C^{\mathsf{a}}, \mathcal{B}$ in P_A.

Property

If P is a set of CHCs and P_{false} is the set obtained by strengthening the clause constraints as just described, then $P \models \mathsf{false}$ if and only if $P_{\mathsf{false}} \models \mathsf{false}$.

Sound and completeness (I)

Lemma

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

- holds for arbitrary interpretations (only assuming that the predicate false is interpreted as false)
- does not depend on the constraint theory

Soundness

- $P \vdash A$ implies $P \models A$
- means that $P \vdash$ false is a sufficient condition for P to have no model, by above Lemma
- corresponds to using a sound proof procedure to find or check a counterexample

Sound and completeness (II)

But soundness is not enough for P to have a model since we need to establish $P \not\models \text{false}$

Completeness

- we approach this problem by using *approximations* to reason about the non-provability of false
- applying the theory of abstract interpretation to a complete proof procedure for atomic formulas (the "fixed-point semantics" for constraint logic programs Jaffar et al. (1994)
- In effect, we construct by abstract interpretation a proof procedure that is *complete* (but possibly not sound) for proofs of atomic formulas
- $P \nvdash \text{false}$ implies $P \not\models \text{false}$ and thus establishes that P has a model