Constrained Horn Clause Verification

Bishoksan Kafle **John Gallagher**

Roskilde University

ENTRA-SENSATION Workshop, Aalborg, 13-05-2014

Overview

Overview

Motivation

Imperative language to CHCs

Tools, Techniques

Our approach to CHC verification

Conclusion and Future works

Why Constrained Horn Clause (CHC) verification?

CHC is a

- suitable intermediate language to express system's behavior
- suitable target language for translating a variety of
  - languages i.e. imperative, functional, concurrent etc.
  - computational models e.g. state machines, transition systems, Markov chain etc.
- a large number of research community working on this including Microsoft.
- success story : Windows device driver verification etc.

Overview

Motivation

Imperative language to CHCs

Tools, Techniques

Our approach to CHC verification

Conclusion and Future works

Translation to CLP form

- semantics based translation (systematic)
- based on partial evaluation of imperative language's interpreter (e.g. XC)

Imperative Program

```
i=0; a=0; b=0;
assume(n > 0);
while (i < n){
   if (*){
     a=a+1;
     b=b+2;
   }else{
    a=a+2;
 b=b+1;  }
   i++; }
assert(a+b == 3*n);
```

CLP Program

```
false:- N>0,I=0,A=0,B=0, l(I,A,B,N).

 l(I,A,B,N):-I < N, l_body(A,B,A1,B1),
       I1 = I+1, l(I1,A1,B1,N).
 l(I,A,B,N):- I >=N, A + B > 3 * N.
 l(I,A,B,N):- I >=N, A + B < 3 * N.
 l_body(A0,B0,A1,B1):- A1 = A0+1,
                         B1 = B0+2.
 l_body(A0,B0,A1,B1):- A1 = A0+2,
                       B1 = B0+1.
```

Definitions

Constrained Horn Clause (CHC)

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

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

CHC is a software verification community's terminology for CLP
From now on CHC and CLP are used interchangeably

CHC Verification

CHC verification problem

- given a set of CHCs P,
- is to check whether there exists a model of P
- P has a model if and only if $P \not\models$ false.

Representation of Interpretations

- An interpretation of P: a set of *constrained facts* of the form $A \leftarrow \mathcal{C}$, where
- A is an atomic formula $p(Z_1, \ldots, Z_n)$ where $Z_1, \ldots, Z_n$ are distinct variables, and
- $\mathcal{C}$ is a constraint over $Z_1, \ldots, Z_n$.

Models

Minimal models

- A model of P is an interpretation that satisfies each clause.
- There exists a minimal model with respect to the subset ordering, denoted $M[\![P]\!]$,
- the minimal model $M[\![P]\!]$ is equivalent to the set of atomic consequences of P (model vs. proof)
- $P \models p(v_1, \ldots, v_n)$ if and only if $p(v_1, \ldots, v_n) \in M[\![P]\!]$
- $M[\![P]\!]$ can be computed as the least fixed point (*lfp*) of an immediate consequences operator, $T_P^{\mathcal{C}}$

Proofs

Proof by over-approximation of the minimal model

- It is sufficient to find a set of constrained facts M' such that $M[\![P]\!] \subseteq M'$, where false $\notin M'$.

Proof by specialisation

- A specialisation of P with respect to an atom A is the transformation of P to another set of CHCs P' such that $P \models A$ if and only if $P' \models A$.
- can be viewed as program optimization
- In our context, w.r.t. to the atom false

Analysis

Convex polyhedron (hull) approximation (CHA)

- CHA is 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.

Overview

Motivation

Imperative language to CHCs

Tools, Techniques

Our approach to CHC verification

Conclusion and Future works

Tools and Techniques

- CHC verification has gained interests from CLP and software verification communities
- Several techniques such as Abstract Interpretation (AI), Counter Example Guided Abstraction Refinement (CEGAR) and program specialization have been proposed in the literature to solve CHC program.
- Tools: Z3, QARMC (CEGAR) , TRACER, VeriMAP (specialisation) etc.

Pitfall

- Several challenging problems, no single technique is powerful enough
- but some of these techniques perform better in some cases while others in some other cases,
- that is, they usually miss the the aspect of the other.
- So their combination could give a better result?

Our approach is to combine the strength of techniques developed for CLP and Software Verification in the same framework

Overview

Motivation

Imperative language to CHCs

Tools, Techniques

Our approach to CHC verification

Conclusion and Future works

Summary of our approach

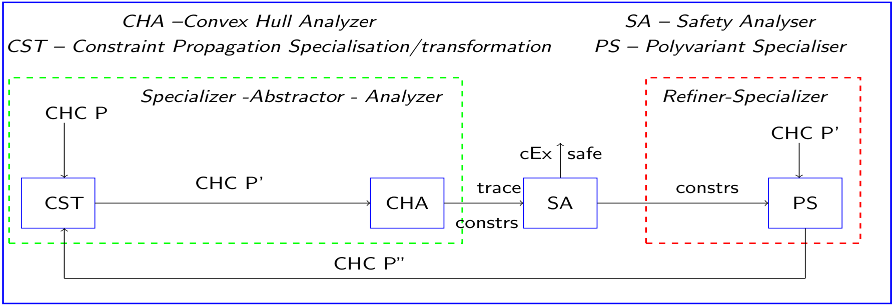

Figure : *Tool chain overview (CHC verification).*

Approach

- abstract interpretation over convex polyhedra domain (main engine),
  - loses precision due to merge and widening operators
  - CLP transformation techniques such as unfolding, predicate splitting, specialization could make analysis results better
- specialisation of the constraints in CHCs using abstract interpretation of query answer transformed clauses (simulates the computation tree semantics of CLP), and
- refinement by predicates splitting (guided by abstract trace)
- generation of new program based on extended set of constraints (CEGAR simulation)

Running Example

Original CHC P

```
false :- 1*A>0,1*B=0,1*C=0,1*D=0,l(B,C,D,A).

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

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

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

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

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

Specialised CHC Sp(P)

```
c1. false :- 1*A>0, 1*B=0, 1*C=0, 1*D=0, l(B,C,D,A).

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

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

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

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

Constraint facts for QA(P)

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

CHA Analysis

CHA Result on Sp(P)

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

- presence of constrained fact for *false* $\rightarrow$ P not safe
- CHA returns counter example trace c1(c3) in the form of trace term
- check trace for feasibility by collecting constraints from the clauses,
  - if feasible then our analysis terminates and returns bug
  - else refine Sp(P)

Tool chain overview

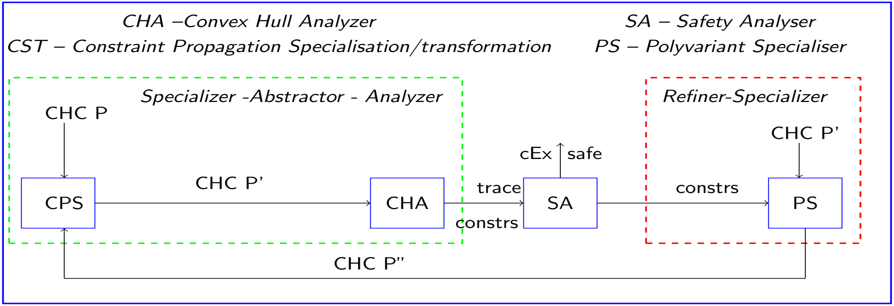

Figure : *Tool chain overview (CHC verification).*

Counter example analysis

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) :- 1*D>0,2*A+ -1*B>=0,-1*A+1*B>=0,-3*A+3*D> -3, 3*A+ -1*B+ -1*C=0.` with the interpolant produces (after constraint simplification)

```
l(A,B,C,D) :-   -4*A+4*B+ -1*D>=0,1*D>0,-3*A+3*D> -3,
                          2*A+ -1*B>=0, 3*A+ -1*B+ -1*C=0.
l(A,B,C,D) :-   4*A+ -4*B+1*D>0,-1*A+1*B>=0,-3*A+3*D> -3,
                       2*A+ -1*B>=0,3*A+ -1*B+ -1*C=0.
```

- Based on these extended set of constraint facts and Sp(P) we generate a new CHC through specialization which is more precise than the original problem.
- In the next iteration we are able to show the presence of a bug and thus our procedure terminates.

Overview

Summary and Future Works

Conclusion:

- presented an approach for CHC verification based on constraint propagation by specialization, program transformation, convex hull analysis and property-based specialisation that splits predicates, leading in turn to more precise convex polyhedral analyses
- experimental results on some challenging benchmark problems from software verification repository prove the feasibility of our approach

For the future:

- explore new ways of refining polyhedra abstraction
- understand better the connection between program specialization and CEGAR
- interface with SMT solvers (for satisfiability checking w.r.t. to some background theory and interpolants generation)

Thanks for your attention!

Questions?