Constraint Specialisation in Horn Clause Verification

Bishoksan Kafle John Gallagher

Roskilde University

PEPM, Mumbai, India, 13/14-01-2015

Introduction

Goal: specialise a set of constrained Horn clauses (program) wrt a goal
Characteristics:

- propagate constraints top down (from the goal) and bottom up
- without unfolding (without size blow-up)

For this we use the theory of abstract interpretation (abstraction) and query-answer transformation(specialisation).
Key contributions:

- method for specialising the constraints in the clauses using query-answer transformation and abstract interpretation;
- demonstrate the effectiveness of transformation by applying it to Horn clause verification problems.

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

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

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

Query answer transformation (QA)

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:

$B^{\mathsf{q}}_1 \leftarrow C, H^{\mathsf{q}}.$

$\ldots$

$B^{\mathsf{q}}_i \leftarrow C, H^{\mathsf{q}}, B^{\mathsf{a}}_1, \ldots, B^{\mathsf{a}}_{i-1}.$

$\ldots$

$B^{\mathsf{q}}_n \leftarrow C, H^{\mathsf{q}}, B^{\mathsf{a}}_1, \ldots, B^{\mathsf{a}}_{n-1}.$

Goal clause

$A^{\mathsf{q}} \leftarrow \mathrm{true}.$

Property (Correctness)

$P \models A$ *iff* $P^{\mathsf{qa}}_A \models A_a$

Query answer transformation example

- Given a clause:

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

QA contains the following clauses:

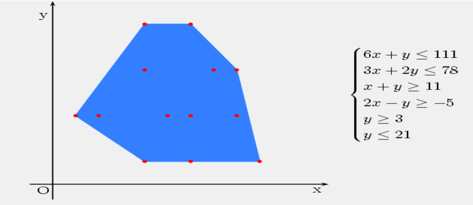
```
l_ans(A,B,C,D) :- l_query(A,B,C,D), -A+D>0,  A-E= -1,
      l_body_ans(B,C,F,G), l_ans(E,F,G,D).

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).
```

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

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.




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

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

Constraint Specialisation

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 $p^a(X) \leftarrow C^a$ (where X is the same tuple of variables in $p(X)$ and $p^a(X)$).
4. Replace the clause $p(X) \leftarrow \mathcal{B}$ in P by $p(X) \leftarrow C^a, \mathcal{B}$ in P_A.

Property (Correctness)

If P is a set of CHCs and P_A is the set obtained by strengthening the clause constraints as just described, then $P \models A \iff$ if $P_A \models A$.

Example: Specialisation by constraint propagation

Computing an over-approximation of the model of P_A^{qa}, we have the following constrained fact for predicate $l\_ans(A, B, C, D)$:

```
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.
```

Now, strengthen

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

which after simplification becomes

```
 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).
```

Overview

1 Query-answer transformation

2 Abstract Interpretation

3 Constraint specialisation

4 Application to Horn clause verification

5 Proof techniques

6 Experimental Results

7 Conclusion and Future works

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?

CHC and CLP

- CHC is a terminology for CLP used by program verification community;
- Unlike CLP, CHCs are not always regarded as executable programs, but rather as specifications or semantic representations of other formalisms;
- but the semantic equivalence of CHC and CLP means that techniques developed in one framework are applicable to the other.

CHC verification

So we exploit the following results from CLP for verification of CHCs

- 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)

Lemma 1

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

Lemma 2

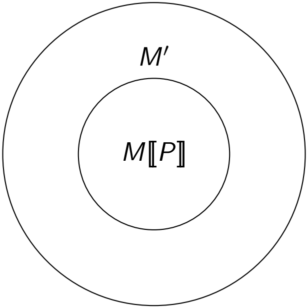
P has a model if and only if false $\notin M[\![P]\!]$.

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

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 $\notin M'$ (safe).
- If false $\in M'$ and there is a feasible computation for false in P then P is unsafe (has bug).
- Feasibility can be checked using decision procedures (e.g. SMT solvers).
- Otherwise we don't know (precision loss – refinement).



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.

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

Settings

benchmarks

- 218 (181 safe and 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 having 4 processors (each @ 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	CS + CPA	QARMC	CS + QARMC
solved (safe/unsafe)	61 (48/13)	162 (144/18)	178 (141/37)	205 (171/34)
unknown / timeout	144/12	49/7	-/40	-/13
total time (secs)	2317	1303	13367	2613
average time (secs)	10.62	5.97	61.31	11.98
%solved	27.98	74.31	81.65	94.04

QARMC (Grebenshchikov et al. PLDI12) is a verification tool based on Counter Example Guided Abstraction Refinement (CEGAR) and uses interpolation.

Experimental results

	CPA	CS + CPA	QARMC	CS + QARMC
solved (safe/unsafe)	61 (48/13)	162 (144/18)	178 (141/37)	205 (171/34)
unknown / timeout	144/12	49/7	-/40	-/13
total time (secs)	2317	1303	13367	2613
average time (secs)	10.62	5.97	61.31	11.98
%solved	27.98	74.31	81.65	94.04

Specialisation enhances the precision of our tool.

Experimental results

	CPA	CS + CPA	QARMC	CS + QARMC
solved (safe/unsafe)	61 (48/13)	162 (144/18)	178 (141/37)	205 (171/34)
unknown / timeout	144/12	49/7	-/40	-/13
total time (secs)	2317	1303	13367	2613
average time (secs)	10.62	5.97	61.31	11.98
%solved	27.98	74.31	81.65	94.04

Specialisation serves as a pre-processor to other tools.

Discussion

- The results show that constraint specialisation is effective in practice.
- We report that 109 out of 218, that is 50%, of the problems are solved by constraint specialisation alone.
- When used as a pre-processor for other verification tools, the results show improvements on both the number of instances solved and the solution time.

Overview

1. Query-answer transformation
2. Abstract Interpretation
3. Constraint specialisation
4. Application to Horn clause verification
5. Proof techniques
6. Experimental Results
7. Conclusion and Future works

Summary and Future Works

Conclusion:

- We introduced a method for specialising the constraints in constrained Horn clauses with respect to a goal using abstract interpretation and *query-answer transformation*.
- The approach propagates constraints globally, both forwards and backwards, and makes explicit constraints from the original program.
- It is a simple and generic approach which is independent of the abstract domain and the constraints theory underlying the clauses.

Summary and Future Works

- Finally, we showed effectiveness of this transformation in Horn clause verification problems.

Future work:

- we will continue to evaluate its effectiveness in a larger set of benchmarks and as a pre-processor for other existing tools.

Availibility of the tool:

- soon we will make the tool available either as a stand-alone program or as an web interface.

The end!

Thanks for your attention!