Relational Verification through Horn Clause Transformation

Emanuele De Angelis (U. Chieti-Pescara, Italy)

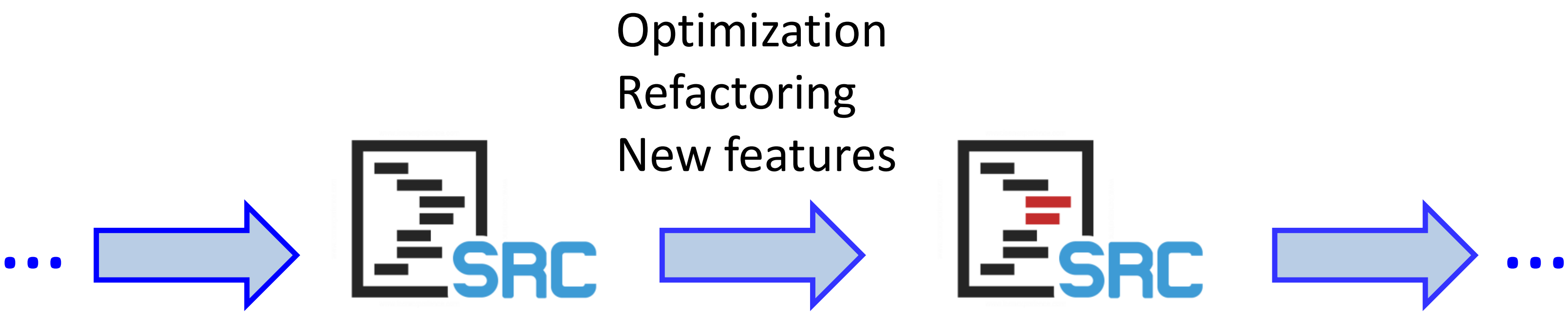
Fabio Fioravanti (U. Chieti-Pescara, Italy)

Alberto Pettorossi (U. Tor Vergata, Rome, Italy)

Maurizio Proietti (IASI-CNR, Rome, Italy)

Relational Properties

- Stepwise program development



- Proving relations between fragments of program versions (e.g., equivalence) may be easier than proving the correctness of the final version from scratch.

An Example

```
void sum_upto() {
  z1=f(x1);
}
int f(int n1){
  int r1;
  if (n1 <= 0) {
    r1 = 0;
  } else {
    r1 = f(n1 - 1) + n1;
  }
  return r1;
}
```

$$z1 = \sum_{n=0}^{x1} n$$

(Non-tail) recursive

```
void prod() {
  z2 = g(x2,y2);
}
int g(int n2, int m2){
  int r2;
  r2=0;
  while (n2 > 0) {
    r2 += m2;
    n2--;
  }
  return r2;
}
```

$$z2 = x2 * y2$$

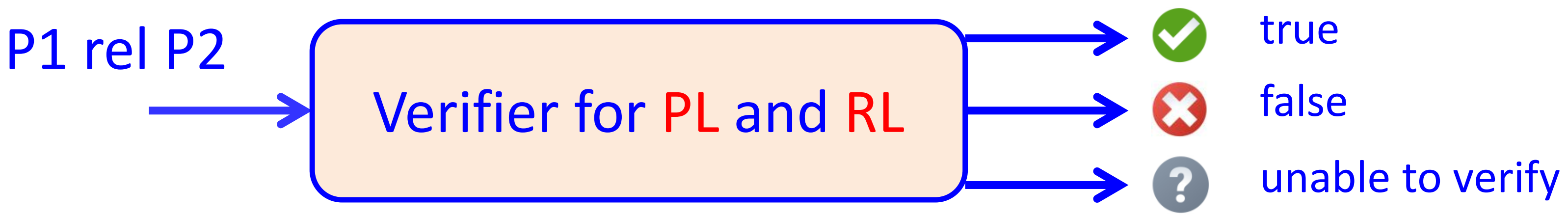
Iterative

- Relational property
  if x1=x2 and x2≤y2 before execution of `sum_upto` and `prod` *and* execution terminates, *then* z1≤z2

Verification of Relational Properties

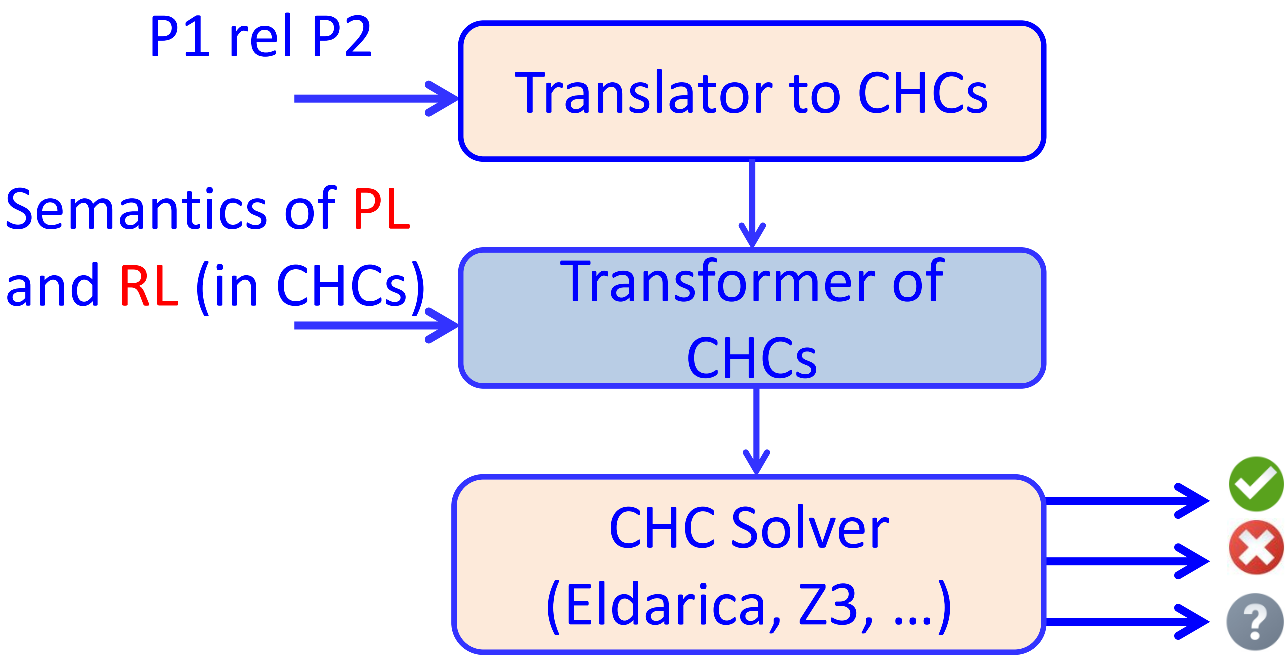
- State-of-the-art verification methods for relational properties are specific for the given programming language PL and class of properties RL [Benton 2004, Barthe *et al.* 2011, Felsing *et al.* 2014]

P1, P2: programs in programming language PL
rel: property in logic RL



Verification through Horn Clause Transformation

Horn Clauses with Constraints (CHCs) as a meta-language for programs, properties, and semantics.



Parametric w.r.t. PL and RL.

Overview

1. Translation of relational properties to Constrained Horn Clauses (CHCs)
2. CHC specialization: generating clauses for a specific verification problem
3. Improving verifiers by predicate pairing and constraint propagation
4. Experimental evaluation

1. Encoding Relational Properties into Constrained Horn Clauses

Relational properties

- C-like programming language on integers and arrays

- Transition semantics

  $<l{:}c, \delta> \Rightarrow <l'{:}c', \delta'>$

  where l:c is a labeled command, δ is an environment mapping variable identifiers to values. $<l{:}c, \delta>$, $<l'{:}c', \delta'>$ are *configurations.*

- Terminating computation

  $\langle P, \delta\rangle \Downarrow \eta$ iff $<l_0{:}c_0, \delta> \Rightarrow^* <l_h{:}\mathrm{halt}, \eta>$

- Relational Property P1, P2 programs with disjoint variables, φ,ψ constraints

  $\{\varphi\}\ P1 \sim P2\ \{\psi\}$

  is valid iff for all disjoint environments δ1, δ2,

  if $\models \varphi[\delta 1 \cup \delta 2]$, $\langle P1, \delta 1\rangle \Downarrow \eta 1$, $\langle P2, \delta 2\rangle \Downarrow \eta 2$ *then* $\models \psi[\eta 1 \cup \eta 2]$

Example, cont'd

```
void sum_upto() {
  z1=f(x1);
}
int f(int n1){
  int r1;
  if (n1 <= 0) {
    r1 = 0;
  } else {
    r1 = f(n1 - 1) + n1;
  }
  return r1;
}
```

```
void prod() {
  z2 = g(x2,y2);
}
int g(int n2, int m2){
  int r2;
  r2=0;
  while (n2 > 0) {
    r2 += m2;
    n2--;
  }
  return r2;
}
```

$$z1 = \sum_{n=0}^{x1} n$$

(Non-tail) recursive

$$z2 = x2 * y2$$

Iterative

Relational Property:

$\{x1=x2 \wedge x2 \leq y2\}$ `sum_upto` $\sim$ `prod` $\{z1 \leq z2\}$

Constrained Horn Clauses (CHCs)

- First order formulas of the form:

  $\forall(A_1 \wedge \ldots \wedge A_n \wedge c \rightarrow A_0)$

  where $A_0, A_1, \ldots, A_n$ are atomic formulas and c is a formula in a theory *Th* of constraints. In this talk: theory of Arrays and Linear Integer Arithmetics (LIA).

- Logic programming syntax: $A_0 \leftarrow c, A_1,\ldots,A_n$

- Goal: false $\leftarrow c, A_1,\ldots,A_n$

- A set of CHCs is satisfiable iff it has a model.

Encoding the Transition Semantics in CHCs

- Transition: $<l{:}c,\ \delta> \;\Rightarrow\; <l'{:}c',\ \delta'>$ [Assignment: x = expr]

  tr(cf(cmd(L,asgn(X,Expr)),Env), cf(cmd(L1,C),Env1)) $\leftarrow$

  eval(Expr,Env,V), update(Env,X,V,Env1), nextlab(L,L1), at(L1,C)

  + clauses for the other commands

- Reflexive-transitive closure $\Rightarrow^*$:

  reach(C,C) $\leftarrow$

  reach(C,C2) $\leftarrow$ tr(C,C1), reach(C1,C2)

- Terminating computation $\langle P, \delta\rangle \Downarrow \eta$ [input/output relation of P]:

  p(X,X') $\leftarrow$ initConf(C,X), reach(C,C'), finalConf(C',X')

  - initConf(C,X): X is the value of the variables in the initial configuration C
  - finalConf(C',X'): X' is the value of the variables in the initial configuration C'

Translating Relational Properties into CHCs

- $\{\varphi\}$ P1 ~ P2 $\{\psi\}$

  Prop: false ← pre(X,Y), p1(X,X'), p2(Y,Y'), neg_post(X',Y')

  φ P1 P2 $\neg\psi$

  X,Y,X',Y': *tuples of values for the variables of* P1, P2, resp.

- T_{Prop} = {*Prop*} ∪ {clauses for *p1* and *p2*}

  Correctness of Translation:

  $\{\varphi\}$ P1 ~ P2 $\{\psi\}$ is valid *iff* T_{Prop} is satisfiable

  φ $\neg\psi$

- Example: false ← X1=X2, X2≤Y2, Z1'>Z2',

  sum_upto(X1,Z1,X1',Z1'), prod(X2,Y2,Z2,X2',Y2',Z2')

Limitations of the Translation to CHCs

- T_{Prop} includes a lot of complex structures and predicates:
  - complex terms encoding configurations:

    cf(cmd(L,asgn(X,Expr)),[(x,1),(y,0),(a,[2,3,4])])

  - recursive predicates over lists encoding functions on the environment:

    update([(X,N)|Bs],X,V,[(X,V)|Cs]) ← update(Bs,X,V,Cs)

- State-of-the-art CHC solvers hardly terminate on T_{Prop}.

2. CHC Specialization

Generating Specialized CHCs

- Simplify T_{Prop} by specializing it wrt the specific programs and property $\{\varphi\}$ P1 $\sim$ P2 $\{\psi\}$.

- T_{SP} is a set of Horn clauses over Arrays and Linear Integer Arithmetic (LIA) constraints. No complex terms or lists.

- Specialization preserves satisfiability:

  T_{Prop} satisfiable *iff* T_{SP} satisfiable

Example Cont'd: CHC Specialization

false ← X1=X2, X2≤Y2, Z1'>Z2',
sum_upto(X1,Z1,X1',Z1'), prod(X2,Y2,Z2,X2',Y2',Z2')

+ clauses for sum_upto and prod

CHC Specializer

Specialized predicates

false ← X1=X2, X2≤Y2, Z1'>Z2', su(X1,Z1'), pr(X2,Y2,Z2')
su(X,Z) ← f(X,Z)
f(N,Z) ← N≤ 0, Z=0
f(N,Z) ← N≥1, N1=N−1, Z=R+N, f(N1,R)
pr(X,Y,Z) ← W=0, g(X,Y,W,Z)
g(N,P,R,R) ← N≤ 0
g(N,P,R,R2) ← N≥1, N1=N−1, R1=P+R, g(N1,P,R1,R2)

No complex terms, no reference to the operational semantics, only variables and constraints over them (CHCs over LIA).

Limitations of the Specialized CHCs

- To show the satisfiability of

  false $\leftarrow$ c(X,Y), p1(X), p2(Y)

  a CHC solver looks for c1(X), c2(Y) such that in $T_{SP} \cup$ Th:

  p1(X) $\rightarrow$ c1(X)
  p2(Y) $\rightarrow$ c2(Y)
  c1(X), c2(Y), c(X,Y) $\rightarrow$ false

- To show the satisfiability of

  false $\leftarrow$ X1=X2, X2≤Y2, Z1'>Z2', su(X1,Z1'), pr(X2,Y2,Z2')

  a CHC solver has to show that:

  su(X1,Z1') $\rightarrow$ Z1'≤ 1+ … + X1
  pr(X2,Y2,Z2') $\rightarrow$ Z2' ≥ X2∗Y2
  Z1'≤ 1+ … + X1, Z2' ≥ X2∗Y2, X1=X2, X2≤Y2, Z1'>Z2' $\rightarrow$ false

- Impossible for CHC solvers over LIA!
  Nonlinear constraints cannot be derived.

3. Predicate Pairing

Inferring Inter-Predicate Relations via CHC Transformation

- Introduce new predicates standing for conjunctions:

false ← c(X,Y), p1(X), p2(Y)

false ← c(X,Y), p12(X,Y)

- Predicate pairing derives new clauses for conjunctions of predicates by unfold/fold transformations and preserves satisfiability.

- To prove satisfiability find constraint **d**(X,Y) such that:

  p12(X,Y) → **d**(X,Y)
  d(X,Y), c(X,Y) → false

- **d**(X,Y) captures relations between the variables of p1 and the variables of p2.

Example Cont'd: Predicate Pairing

$false \leftarrow X1=X2,\ X2 \leq Y2,\ Z1' > Z2',$
$\quad su(X1,Z1'),\ pr(X2,Y2,Z2')$

$su(X,Z) \leftarrow f(X,Z)$

$f(N,Z) \leftarrow N \leq 0,\ Z=0$

$f(N,Z) \leftarrow N \geq 1,\ N1=N-1,\ Z=R+N,\ f(N1,R)$

$pr(X,Y,Z) \leftarrow W=0,\ g(X,Y,W,Z)$

$g(N,P,R,R) \leftarrow N \leq 0$

$g(N,P,R,R2) \leftarrow$
$\quad N \geq 1,\ N1=N-1,\ R1=P+R,$
$\quad g(N1,P,R1,R2)$

→ **Predicate Pairing** ↓

$false \leftarrow N \leq Y,\ W=0,\ Z1' > Z2',$
$\quad fg(N,Z1',Y,W,Z2')$

$fg(N,Z1',Y,Z2',Z2') \leftarrow N \leq 0,\ Z1'=0$

$fg(N,Z1',Y,W,Z2') \leftarrow$
$\quad N>1,\ N1=N-1,\ Z1'=R+N,\ M=Y+W,$
$\quad fg(N1,R,Y,M,Z2')$

- $fg(N,Z1',Y,0,Z2') \rightarrow$ **$N>Y \vee Z1' \leq Z2'$**
  (**$N>Y \vee Z1' \leq Z2'$**) $\wedge\ N \leq Y \wedge W=0 \wedge Z1' > Z2' \rightarrow false$
- Non-linear arithmetic relations not needed for proving satisfiability.
  CHC solvers over LIA (Eldarica, Z3) can prove satisfiability.

Constraint Propagation

- Strengthen the constraints in the clauses by propagating the constraints occurring in the goals [De Angelis *et al.* 2014]
- Example:

false ← X<0, p(X)

p(X) ← X=Y+1, p(Y)

p(X) ← X≥0, r(X)

r(X) ← …

→ **Constraint Propagation** ↓

false ← X<0, p'(X)

p'(X) ← X<0, X=Y+1, p'(Y)

~~p'(X) ← X<0, X≥0, r(X)~~

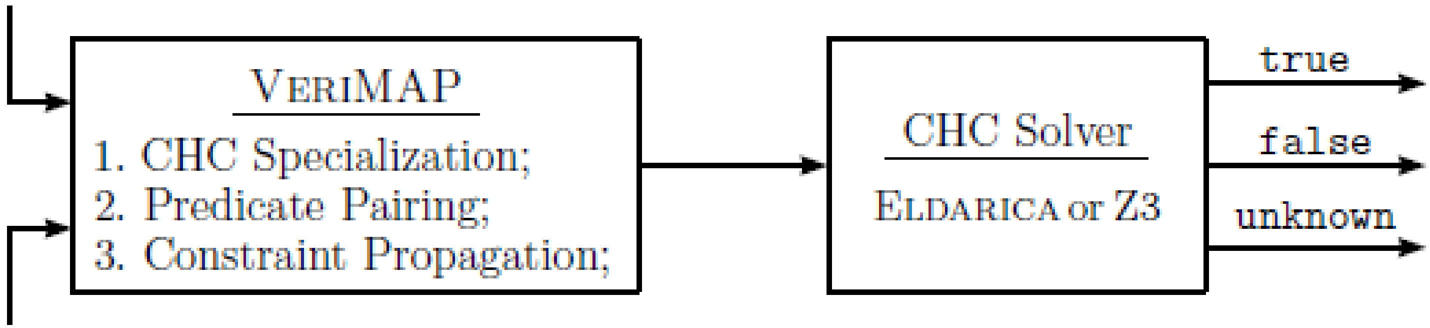
Satisfiable

(take an interpretation where p'(X) is FALSE)

4. Experimental Evaluation

Implementation in VeriMAP

{φ} P1 ~ P2 {ψ}



CHC semantics of PL

http://www.map.uniroma2.it/VeriMAP.

Verification Problems

Types of Verified Properties and Programs

- ITE: Equivalence of two iterative programs
- REC: Equivalence of two recursive programs
- I-R: Equivalence of an iterative and a recursive program
- ARR: Equivalence of two programs on arrays
- LEQ: Arithmetic comparison ($\leq$) between variables
- MON: Monotonicity
- INJ: Injectivity
- FUN: Functional dependencies among variables
- COMP: Relations between sequential composition of programs

Experimental Results

Problem Category	M	Sp time	PP time	CP time	(1) Sp + Eld		(2) Sp + Z3		(3) Sp + PP + Eld		(4) Sp + PP + Z3		(5) Sp + PP+CP+Eld		(6) Sp + PP+CP+Z3	
					N	time	N	time	N	time	N	time	N	time	N	time
ITE	21	0.10	5.32	0.46	9	23.94	6	0.88	15	19.09	12	17.00	18	16.83	21	11.36
REC	18	0.12	2.88	0.31	8	6.40	8	4.39	14	6.67	14	3.19	14	6.67	15	3.12
I-R	4	0.11	2.30	0.37	0	—	0	—	1	15.88	1	7.19	4	6.83	4	2.68
ARR	5	0.33	0.10	1.07	0	—	0	—	1	11.09	3	1.74	1	11.09	3	1.74
LEQ	6	0.10	0.80	0.17	0	—	0	—	0	—	0	—	2	6.32	3	1.11
MON	18	0.05	2.38	(*) 0.15	6	9.62	4	0.25	11	9.77	8	0.97	11	9.77	14	1.43
INJ	11	0.05	1.31	0.15	2	11.38	0		6	55.80	5	1.89	6	55.80	10	1.70
FUN	7	0.05	3.62	0.10	5	4.52	5	0.24	7	5.23	7	0.59	7	5.23	7	0.59
COMP	10	0.26	0.65	19.61	0	—	0	—	3	24.40	6	4.51	6	16.15	9	3.70
Total number: avg. time:	100	0.11	2.67	2.24	30	12.32	23	1.85	58	16.53	56	5.53	69	14.83	86	4.41

Table 1. M is the number of verification problems. N is the number of solved problems. Times are in seconds. The timeout is 5 minutes. Sp is CHC Specialization, PP is Predicate Pairing, CP is Constraint Propagation, Eld is ELDARICA. (*) One problem in the category MON timed out.

Conclusions

- Our method for relational verification:
  - Translation to CHCs;
  - Satisfiability-Preserving Transformations of CHCs;
  - CHC Solving
- Parametric wrt programming language
- Fully automatic and effective on small-sized programs

Future work

- Proving relations across programming languages to validate program translation/compilation
- Applications (e.g., non-interference)

Thank You for Your Attention!