Removing Unnecessary Variables from Horn Clause Verification Conditions

E. De Angelis (1), F. Fioravanti (1)
A. Pettorossi (2), M. Proietti (3)

(1) DEC, University "G. d’Annunzio" of Chieti-Pescara, Italy
(2) DICII, University of Rome Tor Vergata, Roma, Italy
(3) CNR-IASI, Roma, Italy

HCVS'16 – Eindhoven
Talk Outline

• Partial Correctness properties

• Verification Conditions Generation
  – using specialization of Constrained Horn Clauses (CHC)
    a.k.a. Constraint Logic Programs (CLP)

• Removing unnecessary variables from CHC
  – Non-Linking variables Removal strategy
    • call dependent
  – Constrained FAR algorithm
    • call independent
    • Variable liveness analysis

• Experimental evaluation
Partial Correctness and VCs

Given the partial correctness property (Hoare triple)

\[
\{ x \geq 0 \} \quad \text{int } x, y; \\
\text{main ()} \{ \\
\quad \text{int } z = x + 1; \\
\quad \text{while } (z \leq 9) \{ z = z + 1; \} \\
\quad y = z; \\
\} \quad \{ y > 0 \}
\]

**Verification Conditions:** formulas whose satisfiability implies correctness

\[\text{incorrect : - } X1 \geq 0, \ \text{newp1}(X1,Y1, X2,Y2), \ Y2=\leq 0.\]
\[\text{newp1}(X1,Y1,X2,Z2) : - Z1=\leq X1+1, \ \text{newp2}(X1,Y1,Z1,X2,Y2,Z2).\]
\[\text{newp2}(X1,Y1,Z1,X2,Y2,Z3) : - Z1= \leq 9, \ Z2=\leq Z1+1, \ \text{newp2}(X1,Y1,Z2,X2,Y2,Z3)\]
\[\text{newp2}(X1,Y1,Z1,X1,Y1,Z1) : - Z1 \geq 10.\]

VCs satisfiability can (possibly) be checked by using Horn solvers and Satisfiability Modulo Theory (SMT) solvers like

- CHA (Gallagher et al.), Duality (McMillan), Eldarica (Ruemmer et al.),
- MathSAT (Cimatti et al.), QARMC/HSF (Rybalchenko et al.),
- SeaHorn (Gurfinkel et al.), TRACER (Jaffar et al.),
- VeriMAP (De Angelis et al.), Z3 (Bjorner & De Moura),
VCs GENeration

Standard approach
- VCGEN algorithm is tailored to the syntax and the semantics of the imperative programming language
- **Cons**: changing the programming language or its semantics usually requires **rewriting** the VCGEN algorithm

Semantics-based approach
[Cousot SAS'97, Gallagher et al. SAS'98, J Strother Moore CHARME'03, Rosu et al '14]
- VCGEN algorithm is **parametric** wrt programming language semantics
- **Pro**: use the same VCGEN algorithm for different programming languages and semantics

Our semantics-based approach
- uses CHC encoding of program, semantics and logic
- VCs generated by CHC specialization
  - correctness of VC generation follows from correctness of the rules
- Parametricity wrt programming language and class of properties
- Flexibility and efficiency
Encoding Imperative Programs

- Imperative language: subset of CIL (C Intermediate Language)
  - assignments, conditionals, jumps, recursive function calls, abort
  - loops translated to conditionals and jumps
- Commands encoded as facts: **at(Label, Cmd)**

Program *Prog*

```c
int x, y;

void main() {
    int z=x+1;
    while (z<=9) {
        z=z+1;
    }
    y=z;
}
```

CLP encoding of *Prog*

```prolog
fun(main,[],[],1).
at(1,asgn(z,plus(x,1))).
at(2,ite(lteq(z,9),3,5)).
at(3,asgn(z,plus(z,1))).
at(4,goto(2)).
at(5,asgn(y,z)).
at(h,halt).
```
Encoding the Operational Semantics

Configurations: \( \text{cf}(\text{LC, Env}) \) program execution state

- **LC** labeled command: a term of the form \( \text{cmd}(L,C) \)
  - \( L \) label, \( C \) command

- **Env** environment: a pair \( (D,S) \)
  - \( D \) global environment, \( S \) local environment
  - Environments as lists of pairs \( [(x,X),(y,Y),(z,Z)] \)

**Operational semantics**: transition relation \( \text{tr} \) between configurations

\[
\text{tr}( \text{cf}(L_{C1},E_{1}), \text{cf}(L_{C2},E_{2}) )
\]

Multiple steps reachability (reflexive, transitive closure of \( \text{tr} \))

\[
\text{reach}(C,C).
\text{reach}(C,C_2) :- \text{tr}(C,C_1), \text{reach}(C_1,C_2).
\]
Encoding the Operational Semantics

**Assignment**  \( x = e; \)

\[
\text{tr( } \text{cf(cmd(L, asgn(X,expr(E))), (D,S)), } \text{cf(cmd(L1,C), (D1,S1)) :- source configuration target configuration target configuration }
\]
\[
\text{eval(E,(D,S),V), evaluate expression }
\]
\[
\text{update(((D,S),X,V,(D1,S1)), update environment }
\]
\[
\text{nextlab(L,L1), next label }
\]
\[
\text{at(L1,C). next command }
\]
Partial correctness property

\[ \{ x \geq 0 \} \; \text{Prog} \; \{ y > 0 \} \]

CHC encoding of (in)correctness.

program I

```
incorrect :- initConf(Cf), reach(Cf,Cf1), errorConf(Cf1).

...  
initConf(cf(C, [(x,X),(y,Y)])) :- at(1,C), X>=0.
errorConf(cf(C, [(x,X),(y,Y)])) :- at(h,C), Y=< 0.
```

**Thm. Correctness of CLP Encoding**

property does not hold   iff   incorrect \( \in \) M(I)

where: M(I) least LIA model of the CLP program I

Undecidable problem. Even if decidable, very hard to check.

Unfold/Fold program specialization for “removing the interpreter” and producing VCs.
Partial Correctness and VCs

Given the partial correctness property (Hoare triple)

\[
\{x \geq 0\} \quad \text{int } x,y; \\
\text{main ()} \\
\quad \text{int } z=x+1; \\
\quad \text{while } (z<=9) \{z=z+1;\} \\
\quad y=z; \\
\} \quad \{y > 0\}
\]

Verification Conditions as constrained Horn clauses

incorrect :- \( X_1 \geq 0, \text{newp1}(X_1,Y_1,X_2,Y_2), Y_2 \leq 0 \).

program execution
(call to the main() function)

\text{newp1}(X_1,Y_1,X_2,Z_2) :- Z_1 = X_1+1, \text{newp2}(X_1,Y_1,Z_1,X_2,Y_2,Z_2).

loop initialization

\text{newp2}(X_1,Y_1,Z_1,X_2,Y_2,Z_2) :- Z_1 \leq 9, Z_3 = Z_1+1, \text{newp2}(X_1,Y_1,Z_3,X_2,Y_2,Z_2)

loop iteration

\text{newp2}(X_1,Y_1,Z_1,X_1,Y_1,Z_1) :- Z_1 \geq 10.

loop exit
Unnecessary variables

• It is well-known that transformational approaches may produce unnecessary variables

• Two solutions from LP (adapted to CHC) for removing (some) unnecessary variables
  – Non-linking variables strategy
    • call dependent
  – Constrained FAR algorithm
    • call independent
    • variable liveness analysis
Non-Linking variables Removal

Let $C$ be a clause of the form

$$H :- c, L, B, R$$

A variable occurring in $B$ is **non-linking** in $C$ if it does not occur in the rest of the clause.

Non-linking variables can be removed from the call.

**Verification Conditions after VCG**

incorrect :- $X1 >= 0$, newp1($X1, Y1, X2, Y2$), $Y2 <= 0$.
newp1($X1, Y1, X2, Y2$) :- $Z1 = X1 + 1$, newp2($X1, Y1, Z1, X2, Y2, Z2$).
newp2($X1, Y1, Z1, X2, Y2, Z2$) :- $Z1 = < 9$, $Z3 = Z1 + 1$, newp2($X1, Y1, Z3, X2, Y2, Z2$).
newp2($X1, Y1, Z1, X1, Y1, Z1$) :- $Z1 >= 10$.

**Verification Conditions after application of the NLR strategy**

incorrect$_{NLR}$ :- $X1 >= 0$, newp3($X1, Y2$), $Y2 <= 0$.
newp3($X1, Z2$) :- $Z1 = X1 + 1$, newp4($X1, Z1, Z2$).
newp4($X1, Z1, Z2$) :- $Z1 = < 9$, $Z3 = Z1 + 1$, newp4($X1, Z3, Z2$).
newp4($X1, Z1, Z1$) :- $Z1 >= 10$. 
NLR strategy

**Input:** a set VC of CHCs
**Output:** $\text{VC}_{\text{NLR}}$

$\text{VC}_{\text{NLR}} := \emptyset$;
Defs := \{incorrect\textsubscript{NLR} :- incorrect \};

while there exists d in Defs to be processed do
  Cls = UNFOLDING(d, VC);
  Defs = Defs U DEFINITION-INTRODUCTION(Cls);
  $\text{VC}_{\text{NLR}} = \text{VC}_{\text{NLR}} U \text{FOLDING}(\text{Cls, Defs})$;
  mark d as processed;
done

**Thm. Termination and correctness of the NLR strategy**

(i) the NLR strategy terminates
(ii) incorrect $\in M(\text{VC})$ iff incorrect\textsubscript{NLR} $\in M(\text{VC}_{\text{NLR}})$
NLR strategy in action

\[ \text{incorrect}_{\text{NLR}} : - \text{incorrect} \]

- **UNFOLDING** (replace leftmost atom incorrect with the body of its definition)
  \[ \text{incorrect}_{\text{NLR}} : - X1 \geq 0, \text{newp1}(X1,Y1,X2,Y2), Y2 < 0. \]

- **DEFINITION-INTRODUCTION** (add a clause with a new head predicate and linking vars)
  \[ d1: \text{newp3}(X1,Y2) : - \text{newp1}(X1,Y1,X2,Y2) \]

- **FOLDING** (replace an instance of the body of a definition by its head)
  \[ \text{incorrect}_{\text{NLR}} : - X1 \geq 0, \text{newp3}(X1,Y2), Y2 < 0. \]

- **UNFOLDING** (of d1)
  \[ \text{newp3}(X1,Z2) : - Z1 = X1 + 1, \text{newp2}(X1,Y1,Z1,X2,Y2,Z2). \]

- **DEFINITION-INTRODUCTION**
  \[ d2: \text{newp4}(X1,Z1,Z2) : - \text{newp2}(X1,Y1,Z1,X2,Y2,Z2). \]

- **FOLDING**
  \[ \text{newp3}(X1,Z2) : - Z1 = X1 + 1, \text{newp4}(X1,Z1,Z2). \quad \text{.... continues ...} \]
NLR strategy in action

- **UNFOLDING**
  
  newp4(X1,Z1,Z2) :- Z1=<9, Z3=Z1+1, newp2(X1,Y1,Z3, X2, Y2, Z2).
  
  newp4(X1,Z1,Z1) :- Z1>=10.

- **DEFINITION-INTRODUCTION**  (no new definition, **reuse** already introduced definition)
  
  d2: newp4(X1,Z1,Z2) :- newp2(X1,Y1,Z1,X2, Y2, Z2).

- **FOLDING**
  
  newp4(X1,Z1,Z2) :- Z1=<9, Z3=Z1+1, newp4(X1,Z3,Z2).
  
  Verification Conditions after NLR

  incorrect_{NLR} :- X1>=0, newp3(X1,Y2), Y2=<0.
  newp3(X1,Z2) :- Z1=X1+1, newp4(X1,Z1,Z2).
  newp4(X1,Z1,Z2) :- Z1=<9, Z3=Z1+1,newp4(X1,Z3,Z2)
  newp4(X1,Z1,Z1) :- Z1>=10.
What if there are calls to the same predicate having different sets of linking variables?

- \( r(X) :- X>0, p(X,Y,Z). \)
  \( s(Y) :- Y=1, p(X,Y,Z). \)

- We could introduce a definition for every different set of variables
  - \( d1: \text{newp1}(X) :- p(X,Y,Z). \)
  - \( d2: \text{newp2}(Y) :- p(X,Y,Z). \)

Risk of exponential increase of the number of definitions!

- Assume that \( d1 \) is currently the only definition for \( p(X,Y,Z) \)
  instead of introducing \( d2 \), we replace \( d1 \) with
    - \( d3: \text{newp3}(X,Y) :- p(X,Y,Z). \)

intersection of non-linking variables (i.e. union of head variables)

- Thus, VCs after NLR have the same size (number of predicates and clauses) of the input VCs, but hopefully less variables.
Constrained FAR - motivation

Verification Conditions after NLR

...  
newp4(X1,Z1,Z2) :- Z1<=9, Z3=Z1+1,newp4(X1,Z3,Z2)  
newp4(X1,Z1,Z1) :- Z1>=10.

• variable X1 plays no role in the (model of) newp4  
  … it does not occur in the constraints and it does not “change”

  newp4(X1,Z1,Z2) holds  iff  newp4(X1, Z1,Z2) holds

• … but X1 could not be removed by NLR

We extend to CHC the FAR algorithm [Leuschel et al, '96]
Constrained FAR

- An erasure $E$ is a set of pairs $(p,k)$ where $p$ is a predicate symbol of arity $n$ and $1 \leq k \leq n$
- Given an erasure $E = \{(p,2), (q,1)\}$ and clause $C: r(X,Y,Z) :- X=Z, p(X,Y), q(Z)$.
  the erased clause $C_E: r(X,Y,Z) :- X=Z, p(X), q$.

- Erasure $E$ is **safe** for $P$ iff for all $(p,k) \in E$ and for all $p(X_1,...,X_n) :- c, G$ in $P$
  - $X_k$ is a variable and $A \models \forall X_k \exists Y c$ where $Y = vars(c) - \{X_k\}$
  - $X_k$ is not constrained to any other variable in $H$
  - $X_k$ is not constrained to any variable in $G_E$

- If $E$ is a safe erasure for program $P$ then for all atoms $B$
  $$B \in M(P) \iff B_E \in M(P_E)$$
**Constrained FAR algorithm**

Let $E = \{(p,k) \mid p \text{ of arity } n \text{ and } 1 \leq k \leq n\}$ be the full erasure.

repeat
  if $E$ is an unsafe erasure due to some $(p,k) \in E$
  then $E = E - \{(p,k)\}$
until $E$ is a safe erasure

**Thm. Termination and correctness of the cFAR algorithm**

The cFAR algorithm terminates and

$$\text{incorrect} \in M(P) \text{ iff } \text{incorrect}_E \in M(P_E)$$
NLR vs cFAR

- NLR and cFAR are incomparable in general
- cFAR cannot erase variables that occur multiple times in the head of a clause

q(Z) :- p(X,Y,Z).
p(X,X,Z).

... but NLR can

newq(Z) :- newp(Z).
newp(Z).
Experimental evaluation

- 320 verification problems written in the C language
  - from TACAS SV-COMP, other public benchmarks
- Z3 with default options (slicing on)

<table>
<thead>
<tr>
<th></th>
<th>VCG; Z3</th>
<th>VCG; NLR; Z3</th>
<th>VCG; NLR; cFAR; Z3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$ Correct answers</td>
<td>196</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>$s$ safe problems</td>
<td>144</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>$u$ unsafe problems</td>
<td>52</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$to$ Timeouts</td>
<td>124</td>
<td>117</td>
<td>108</td>
</tr>
<tr>
<td>$n$ Total problems</td>
<td>320</td>
<td>124</td>
<td>117</td>
</tr>
<tr>
<td>$t_{VCG}$ VCG time</td>
<td>40.65</td>
<td>20.48</td>
<td>4.57</td>
</tr>
<tr>
<td>$t_{NLR}$ NLR time</td>
<td>–</td>
<td>58.39</td>
<td>9.53</td>
</tr>
<tr>
<td>$t_{cFAR}$ cFAR time</td>
<td>–</td>
<td>–</td>
<td>304.84</td>
</tr>
<tr>
<td>$st$ Z3 solving time</td>
<td>2704.95</td>
<td>988.15</td>
<td>649.56</td>
</tr>
<tr>
<td>$t$ Total time</td>
<td>2745.60</td>
<td>1067.02</td>
<td>968.50</td>
</tr>
<tr>
<td>$at$ Average time</td>
<td>14.01</td>
<td>152.43</td>
<td>107.61</td>
</tr>
</tbody>
</table>

Table 1: Verification results obtained by using Z3 on the output generated by applying VCG and the auxiliary transformations NLR and cFAR. The timeout limit time is five minutes. Times are in seconds.
Conclusions

- Removing unnecessary variables may help Horn solvers

- Future work
  - Apply to VCs generated by other tools
  - Experiment with different solvers

- Benchmarks, VCs and tool at http://map.uniroma2.it/vcgen/