Enhancing Predicate Pairing with Abstraction for Relational Verification

E. De Angelis¹, F. Fioravanti¹, A. Pettorossi², and M. Proietti³

¹ University of Chieti-Pescara ‘G. d'Annunzio'
² University of Rome ‘Tor Vergata'
³ CNR - Istituto di Analisi dei Sistemi ed Informatica
Relational verification

(1)

two different program executions

Program Monotonicity

If $P$ terminates on the input $i_1$ producing $o_1$ & $P$ terminates on the input $i_2$ producing $o_2$ & $i_1$ is less than $i_2$
then $o_1$ is less than $o_2$
Relational verification

two different programs

\[ P_1(i_1) \sim P_2(i_2) \]

**Program Equivalence**

If \( P_1 \) terminates on the input \( i_1 \) producing \( o_1 \) & \( P_2 \) terminates on the input \( i_2 \) producing \( o_2 \) & \( i_1 \) equals to \( i_2 \), then \( o_1 \) equals to \( o_2 \)
The relational property holds if & only if

\( \{ \mathrm{CHCs\ for\ } P_1 \sim P_2 \} \cup \{ \mathrm{CHCs\ for\ } P_1 \} \cup \{ \mathrm{CHCs\ for\ } P_2 \} \) is **satisfiable**
Example

CHC encoding

P1

```c
int a, b, x, y;
while (a < b) {
    x = x+a;
    y = y+x;
    a = a+1;
}
```
Example
CHC encoding

P1

<table>
<thead>
<tr>
<th>int a, b, x, y;</th>
<th>A1, B1, X1, and Y1</th>
<th>A1’, B1’, X1’, and Y1’</th>
</tr>
</thead>
<tbody>
<tr>
<td>while (a &lt; b) {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x = x+a;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y = y+x;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a = a+1;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>int a, b, x, y;</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1, B1, X1, and Y1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1', B1', X1', and Y1'</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>while (a &lt; b) {</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x = x + a; )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( y = y + x; )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( a = a + 1; )</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>}</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>P1whel</strong> (A1,B1,X1,Y1, A1’,B1’,X1’,Y1’)**</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Example

### CHC encoding

The code snippet describes a while loop that increments `a` and `x` based on the comparison of `a` and `b`. The input and output values are as follows:

<table>
<thead>
<tr>
<th>int a, b, x, y;</th>
<th>A1, B1, X1, and Y1</th>
<th>A1, B1, X1, and Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>while (a &lt; b) {</td>
<td>P1whl(A1,B1,X1,Y1, A1’,B1’,X1’,Y1’ ) ← P1whl(A1”,B1,X1”,Y1”, A1’,B1’,X1’,Y1”)</td>
<td>P1whl(A1,B1,X1,Y1, A1,B1,X1,Y1) ← A1≥B1</td>
</tr>
<tr>
<td>x = x+a;</td>
<td>A1≤B1-1, X1”=A1+X1, Y1”=Y1+X1, A1”=A1+1</td>
<td></td>
</tr>
<tr>
<td>y = y+x;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a = a+1;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where `P1whl` represents the input/output relation.
### Example

#### CHC encoding

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
</table>
| while (a < b) {  
x = x+a;  
y = y+x;  
a = a+1;  
} | if (a < b) {  
x = x+a;  
while (a < b-1) {  
y = y+x;  
a = a+1;  
x = x+a;  
}  
y = y+x;  
a = a+1;  
} |

\[
\text{P1whl}(A_1,B_1,X_1,Y_1,A_1',B_1',X_1',Y_1') \iff A_1 \leq B_1 - 1, X_1'' = A_1 + X_1, Y_1'' = Y_1 + X_1, A_1'' = A_1 + 1, \\
\text{P1whl}(A_1'',B_1,X_1'',Y_1'',A_1',B_1',X_1',Y_1') \\
\text{P1whl}(A_1,B_1,X_1,Y_1,A_1,B_1,X_1,Y_1) \iff A_1 \geq B_1
\]
## Example

### CHC encoding

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
</table>
| while (a < b) {  
  x = x+a;  
  y = y+x;  
  a = a+1;  
} | if (a < b) {  
  x = x+a;  
  while (a < b-1) {  
    y = y+x;  
    a = a+1;  
    x = x+a;  
  }  
  y = y+x;  
  a = a+1;  
} |

Example equivalence

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{P1whl}(A_1,B_1,X_1,Y_1,A_1',B_1',X_1',Y_1') \leftarrow A_1 \leq B_1-1, X_1'' = A_1 + X_1, Y_1'' = Y_1 + X_1, \ldots,$</td>
<td>$\text{P2ite}(A_2,B_2,X_2,Y_2,A_2',B_2',X_2',Y_2') \leftarrow A_2 \leq B_2-1, X_2'' = X_2 + A,$</td>
</tr>
<tr>
<td>$\text{P1whl}(A_1'',B_1,X_1'',Y_1'',A_1',B_1',X_1',Y_1')$</td>
<td>$\text{P2whl}(A_2,B_2,X_2'',Y_2,A_2',B_2',X_2',Y_2')$</td>
</tr>
<tr>
<td>$\text{P1whl}(A_1,B_1,X_1,Y_1,A_1,B_1,X_1,Y_1)$ $\leftarrow A_1 \geq B_1$</td>
<td>$\text{P2ite}(A_2,B_2,X_2,Y_2,A_2,B_2,X_2,Y_2)$ $\leftarrow A_2 \geq B_2$</td>
</tr>
<tr>
<td>$\text{P2whl}(A_2,B_2,X_2,Y_2,A_2',B_2',X_2',Y_2')$</td>
<td>$\text{P2whl}(A_2,B_2,X_2',Y_2',A_2',B_2',X_2',Y_2')$</td>
</tr>
<tr>
<td>$\text{P2whl}(A_2,B_2,X_2,Y_2,A_2',B_2',X_2',Y_2')$ $\leftarrow A_2 \leq B_2-2, Y_2''' = Y_2 + X_2, A_2''' = A_2 + 1, \ldots,$</td>
<td>$\text{P2whl}(A_2',B_2,X_2'',Y_2'',A_2',B_2',X_2',Y_2')$</td>
</tr>
<tr>
<td>$\text{P2whl}(A_2',B_2,X_2',Y_2',A_2',B_2',X_2',Y_2')$</td>
<td>$\text{P2whl}(A_2,B_2,X_2,Y_2,A_2',B_2,X_2,Y_2')$ $\leftarrow A_2 \geq B_2-1, Y_2' = Y_2 + X_2, A_2' = A_2 + 1$</td>
</tr>
</tbody>
</table>

$A_1 = A_2, \quad B_1 = B_2, \quad X_1 = X_2, \quad Y_1 = Y_2,$

$\text{P1whl}(A_1,B_1,X_1,Y_1,A_1',B_1',X_1',Y_1'), \quad \text{P2ite}(A_2,B_2,X_2,Y_2,A_2',B_2',X_2',Y_2') \rightarrow X_1' = X_2'$
Example equivalence

A1=A2, B1=B2, X1=X2, Y1=Y2,
  \textbf{P1whl}(A1,B1,X1,Y1, A1′,B1′,X1′,Y1′), \textbf{P2ite}(A2,B2,X2,Y2, A2′,B2′,X2′,Y2′) → X1′=X2′

false ← A1=A2, B1=B2, X1=X2, Y1=Y2, X1′ ≠ X2′,
  \textbf{P1whl}(A1,B1,X1,Y1, A1′,B1′,X1′,Y1′), \textbf{P2ite}(A2,B2,X2,Y2, A2′,B2′,X2′,Y2′)
Satisfiability of CHCs

State-of-the-art solvers for CHCs with Linear Integer Arithmetic (LIA) look for models of single atoms:

to prove that \( P1\text{whl} \) and \( P2\text{ite} \) are equivalent solvers should discover quadratic relations.

\[
X_1' = X_1 + \frac{(B_1 - A_1) \cdot (B_1 + A_1 - 1)}{2}
\]
Satisfiability of CHCs

State-of-the-art solvers for CHCs with Linear Integer Arithmetic (LIA) look for models of single atoms:

to prove that P1whl and P2ite are equivalent solvers should discover quadratic relations.

“solution”

buy a smarter solver, that is, a solver for non-linear integer arithmetic
drawback:
satisfiability of constraints is **undecidable**
(decide satisfiability of Diophantine equations)
Our contribution

We are able to use LIA …

… if we apply some transformations to CHCs

To make the conjunction of $P_{1whl}$ and $P_{2ite}$ enables solvers to look for models of their conjunction.

To discover linear relations among the arguments of $P_{1whl}$ and $P_{2ite}$ may help solvers to prove the satisfiability of CHCs.
Rule-based transformation of CHCs

\[ S \text{ is satisfiable if } \& \text{ only if } T \text{ is satisfiable} \]
**Transformation strategy**  (1)

*unfold* the atoms \( P_1\text{whl}(\ldots) \) and \( P_2\text{ite}(\ldots) \), that is, replace \( P_1\text{whl}(\ldots) \) and \( P_2\text{ite}(\ldots) \) with their **bodies**

\[
\begin{align*}
\text{false} & \leftarrow c, P_1\text{whl}(\ldots), P_2\text{ite}(\ldots) \\
P_1\text{whl}(\ldots) & \leftarrow d, P_1\text{whl}(\ldots) \\
P_2\text{ite}(\ldots) & \leftarrow e, P_2\text{whl}(\ldots) \\
\text{false} & \leftarrow c, d, P_1\text{whl}(\ldots), e, P_2\text{whl}(\ldots)
\end{align*}
\]
Transformation strategy (2)

Given a clause obtained by unfolding

\[
\text{false} \leftarrow c, \ d, \ P1whl(...), \ e, \ P2whl(...)\]

**define** a new predicate

\[
P1whlP2whl(...) \leftarrow P1whl(...), \ P2whl(...)\]

equivalent to the conjunction \( P1whl(...), \ P2whl(...) \)
**Transformation strategy (3)**

fold, that is, replace the atoms $P1\text{whl}(\ldots)$ and $P2\text{ite}(\ldots)$ with the new predicate $P1\text{whl}P\text{whl}(\ldots)$

```
false ← c, d, $P1\text{whl}(\ldots)$, e, $P2\text{whl}(\ldots)$
```

```
false ← c, d, e, $P1\text{whl}P\text{whl}(\ldots)$
```

Solvers will look for models of the conjunction.
Transformation strategy
Assembling new definitions

The transformation strategy is parametric with respect to a \textit{partition operator} that selects the atoms to create new predicate definitions:

- \textbf{one atom} $\rightarrow$ \textbf{Specialization}
- \textbf{two atoms} $\rightarrow$ \textbf{Predicate Pairing (PP)}

Definitions with three or more atoms can be obtained by \textbf{iterating} PP.
Enhancing predicate pairing

Abstraction-based Predicate Pairing (APP)

\[ P_{1\text{whl}}P_{2\text{whl}}(...) \leftarrow a, \ P_{1\text{whl}}(...), \ P_{2\text{whl}}(...) \]

the definition is augmented with a constraint \( a \) representing some relations among the arguments of \( P_{1\text{whl}} \) and \( P_{2\text{whl}} \).

The new constraint \( a \) is an abstraction of the constraint \( c, d, e \)

\[(c, d, e) \rightarrow a\]

occurring in the clause obtained by unfolding:

\[ \text{false} \leftarrow c, d, P_{1\text{whl}}(...) , e, P_{2\text{whl}}(...) \]
The transformation strategy is parametric with respect to the abstract constraint domain for representing the relations among the atoms of the new predicate definitions.
Example
APP with Convex Polyhedra

New predicate definitions:

\[ \text{P1whl} \text{P2ite} \Rightarrow (A, B, X, Y, A_1', B_1', X_1', Y_1', A, B, X, Y, A_2', B_2', X_2', Y_2') \]
\[ X_1' \leq X_2' - 1, \quad \text{P1whl}(A, B, X, Y, A_1', B_1', X_1', Y_1'), \quad \text{P2ite}(A, B, X, Y, A_2', B_2', X_2', Y_2') \]

Final set of CHCs:

\[ \text{false} \leftarrow A_1 = A_2, B_1 = B_2, X_1 = X_2, Y_1 = Y_2, X_1' + 1 \leq X_2', \]
\[ \text{P1whlP2ite}(A_1, B_1, X_1, Y_1, A_1', B_1', X_1', Y_1', A_2, B_2, X_2, Y_2, A_2', B_2', X_2', Y_2') \]
\[ \text{P1whlP2ite}(A, B, C, D, E, F, G, H, A, B, C, D, I, J, K, L) \leftarrow \]
\[ G \leq K - 1, A \leq B - 1, M = A + C, \]
\[ \text{P1whlP2ite}(A, B, C, D, E, F, G, H, A, B, M, D, I, J, K, L) \]
\[ \text{P1whlP2whl}(A, B, C, D, E, F, G, H, A, B, K, D, M, N, O, P) \leftarrow \]
\[ G \leq O - 1, A \leq B - 2, K = A + C, R = A + 1, T = A + C, S = D + T, X = A + 1, W = K + X, Y = D + K, \]
\[ \text{P1whlP2whl}(R, B, T, S, E, F, G, H, X, B, W, Y, M, N, O, P) \]
Implementation

VeriMAP

(1) unfold
(2) define
(3) fold

Transformation strategy

CHCs
(1) unfold
(2) define
(3) fold

Parma Polyhedra Library

Transformed CHCs

Z3

CHC solver

satisfiable
unknown
unsatisfiable

(1) unfold
(2) define
(3) fold

Transformation strategy

VeriMAP

CHCs

(1) unfold
(2) define
(3) fold

Parma Polyhedra Library

Transformed CHCs

Z3

CHC solver

satisfiable
unknown
unsatisfiable

(1) unfold
(2) define
(3) fold

Transformation strategy

VeriMAP

CHCs

(1) unfold
(2) define
(3) fold

Parma Polyhedra Library

Transformed CHCs

Z3

CHC solver

satisfiable
unknown
unsatisfiable
### Benchmark suite

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td>Verification problems</td>
</tr>
<tr>
<td>1655</td>
<td>CHCs</td>
</tr>
</tbody>
</table>

### Relational properties

<table>
<thead>
<tr>
<th>Relation</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence</td>
<td>$p_1(X,X'), \ p_2(Y,Y'), \ X = Y \rightarrow X' = Y'$</td>
</tr>
<tr>
<td>Monotonicity</td>
<td>$p(X,X'), \ p(Y,Y'), \ X \leq Y \rightarrow X' \leq Y'$</td>
</tr>
<tr>
<td>Injectivity</td>
<td>$p(X,X'), \ p(Y,Y'), \ X' = Y' \rightarrow X = Y$</td>
</tr>
<tr>
<td>Functionality</td>
<td>$p(X,f(X),X'), \ p(Y,f(Y),Y'), \ X = Y \rightarrow X' = Y'$</td>
</tr>
</tbody>
</table>
Results

**BDS** is the best, followed by OS
expressive enough for proving equivalence,
monotonicity, injectivity and functionality.

More precise domains start losing problems.

Specialization does not increase the number of problems solved and does not scale
(polyvariant specialization causes a blow-up of the number of clauses)
Conclusions

A method for combining

- transformation
- abstraction

techniques, for proving relational properties

Improves effectiveness of state-of-the art CHC solvers

TODO: a finer control of the definition introduction to keep the size of transformed programs smaller