Using CLP Simplifications to Improve Java Bytecode Termination Analysis

Fausto SPOTO, Lunjin LU, and Fred MESNARD

Dipartimento di Informatica, Università di Verona, Italy
Oakland University, U.S.A.
IREMIA, université de la Réunion, France

March 2009
JuliaWeb is a termination analyzer for Java Bytecode: http://julia.scienze.univr.it/termination

In this paper:

- we propose a set of simplifications of the CLP programs generated by the termination analysis in JuliaWeb;
- we prove those transformations correct w.r.t. termination;
- we experiment with those transformations.
Example

block6391(IL0, IL1, IS0) :-
{IS0=OS1, IL1=OS4, IS0=OS0, IL1=OL1, IL0=OL0,
 OS3 >= 0, OS2 >= 0, IL0-OS3 >= 1, IL0-OS2 >= 1},
block6392(OL0, OL1, OS0, OS1, OS2, OS3, OS4).

block6391(IL0, IL1, IS0) :-
{IS0=OS1, IL1=OS4, IS0=OS0, IL1=OL1, IL0=OL0,
 OS3 >= 0, OS2 >= 0, IL0-OS3 >= 1, IL0-OS2 >= 1},
block6560(OL0, OL1, OS0, OS1, OS2, OS3, OS4).

- Predicates are named blocki or entryj;
- Two arrows connect block 6391 with blocks 6392 and 6560;
- Two local variables L0 and L1 are in scope: L0 implements this and L1 other;
- At the beginning of block 6391, there is only one stack element S0, while there are 5 at its end.
Let \( \vec{m}, \vec{n} \in \mathbb{Z}^* \) and \( C \) be the clause
\[
p(\vec{i}) \leftarrow c, q(\vec{o})
\]
where \( c \) is a linear constraint over the variables \( \vec{i} \cup \vec{o} \).

- \( q(\vec{n}) \) is derived from \( p(\vec{m}) \) using \( C \), written \( p(\vec{m}) \rightarrow_C q(\vec{n}) \), if there is a solution \( \theta \) of \( c[\vec{i} \mapsto \vec{m}] \) such that \( q(\vec{n}) = q(\vec{o})\theta \).

- A derivation of \( p_0(\vec{n}_0) \) is \( p_0(\vec{n}_0) \rightarrow p_1(\vec{n}_1) \rightarrow \cdots \rightarrow p_k(\vec{n}_k) \) such that \( p_{i+1}(\vec{n}_{i+1}) \) is derived from \( p_i(\vec{n}_i) \) for all \( 0 \leq i < k \).

- A resolution is a maximal derivation.

NB: our semantics is \textit{ground} CLP.
An entry \( p \) terminates in a program \( P \) if, for every \( \vec{n} \in \mathbb{Z}^* \), all resolutions of \( p(\vec{n}) \) by using the clauses of \( P \), with predicates in the strongly connected component of \( p \), are finite. Otherwise, \( p \) diverges.

Let \( P_1 \) and \( P_2 \) be programs. \( P_1 \) terminates more than \( P_2 \), written \( P_1 \sqsupset P_2 \), if whenever an entry of \( P_1 \) terminates in \( P_1 \), it also terminates in \( P_2 \).

\( P_1 \) and \( P_2 \) are termination-equivalent, written \( P_1 \equiv P_2 \), if \( P_1 \) terminates more than \( P_2 \) and vice versa.

NB: our definition formalizes a loop-local notion of termination.
A clause $p(\vec{i}) :- c, q(\vec{o})$ occurs in a loop if $p$ and $q$ belong to the same strongly connected component of predicates.

**Proposition**

Let $P$ be a program and $P_s$ be the same program deprived of those clauses that do not occur in a loop. Then $P \equiv P_s$. 
If a program contains clauses \( p(\vec{m}) : \neg c_1, q(\vec{n}) \) and \( q(\vec{v}) : \neg c_2, r(\vec{w}) \), we can unfold them into the clause \( p(\vec{m}) : \neg (c_1 \land c_2 \land \vec{n} = \vec{v}) \land r(\vec{w}) \).

Done systematically for all occurrences of \( q \) on the right of the clauses of \( P \), and followed by the removal of the clauses defining \( q \), we say that we unfold \( q \) away from \( P \).

**Proposition**

Let \( P \) be a program and \( q \) a non-entry predicate in \( P \) with no clause of the form \( q(\vec{n}) : \neg c, q(\vec{m}) \). Let \( P_s \) be \( P \) where \( q \) has been unfolded away. Then \( P \equiv P_s \).
PS3: Removing unsupported or subsumed clauses

\[ p(\vec{i}) :- c, \text{ } q(\vec{o}) \text{ is } \textit{unsupported} \text{ if } q \text{ is undefined.} \]

\[ p(\vec{i}) :- c_1, \text{ } q(\vec{o}) \text{ } \textit{subsumes} \text{ } p(\vec{i}) :- c_2, q(\vec{o}) \text{ if } c_1 \text{ entails } c_2. \]

**Proposition**

Let \( P \) be a program and \( P_s \) be \( P \) deprived from unsupported or subsumed clauses. Then \( P \equiv P_s \).
Let \( c \) be a constraint:

- \( c^v \) is the \( v \)-dedicated part of \( c \) \( = \exists_{\{iv,ov\}} \cdot c \)
- \( c^{-v} \) is the \( v \)-independent part of \( c \) \( = \exists_{\{iv,ov\}} \cdot c \)

An operation that removes a variable from a predicate:

\[
p(iv_1, \ldots, iv_n) \ominus v = \begin{cases} 
p(iv_1, \ldots, iv_{i-1}, iv_{i+1}, \ldots, iv_n) & \text{if } v \equiv v_i \\
p(iv_1, \ldots, iv_n) & \text{otherwise.} \end{cases}
\]

The transformation:

\[
Comp^{-v} = \{ p(\tilde{i}) \ominus v :- c^{-v}, q(\tilde{o}) \ominus v \mid p(\tilde{i}) :- c, q(\tilde{o}) \in Comp \}
\]

removes \( v \) from a strongly connected component \( Comp \).
Proposition

Let $p_0$ be an entry diverging in $\text{Comp}$. Then $p_0$ also diverges in $\text{Comp}^{-v}$.

In general, $\text{Comp}$ is not termination-equivalent to $\text{Comp}^{-v}$, as shown by the counter-example 4.8 of the paper.

In what follows, we identify two cases where removal of a variable maintains termination equivalence. Common condition:

- a variable $v$ is isolated in a strongly connected component $\text{Comp}$ if, for every $p(i) :- c, q(o) \in \text{Comp}$, $c = c^v \land c^{-v}$.
An isolated variable \( v \) in a strongly connected component \( \text{Comp} \) is right-open if, for every \( p(\vec{i}) \) :- \( c, q(\vec{o}) \in \text{Comp} \), we have that \( c^v \) is either true or \( iv = ov \), or \( ov \geq const \), \( ov = const \) or \( ov \leq const \) (or equivalent), where \( const \) is an integer constant. Left-openness is defined analogously.

Proposition

Let \( v \) be right- or left-open in a strongly connected component \( \text{Comp} \). If an entry diverges in \( \text{Comp}^{-v} \) then it diverges in \( \text{Comp} \).
Example

L1 is isolated and right-open in the component:

```
entry3880(IL0,IL1) :-
    {IL1 = OL1, OL0 >= 0, IL0 >= 2, IL0 - OL0 >= 1},
    entry3880(OL0,OL1).
```

Hence L1 can be removed:

```
entry3880(IL0) :-
    {OL0 >= 0, IL0 >= 2, IL0 - OL0 >= 1},
    entry3880(OL0).
```
An isolated variable $v$ is *uniform* in a strongly connected component $\text{Comp}$ if there is $x \in \mathbb{Z}$ such that, for every $p(\vec{i}) :- c, q(\vec{o}) \in \text{Comp}$, the valuation $\{iv \mapsto x, ov \mapsto x\}$ is a solution of $c^v$.

**Property**

Let a variable $v$ be uniform in a strongly connected component $\text{Comp}$. If an entry diverges in $\text{Comp}^{-v}$ then it diverges $\text{Comp}$. 
Example

block3853(IL0,IL1,IL2) :-
    {IL2 - OL2 = -1, IL1 = OL1, IL0 = OL0, IL1 - IL2 >= 1},
    block3853(OL0,OL1,OL2).
block3853(IL0,IL1,IL2) :-
    {IL2 - OL2 = -1, IL1 = OL1, OL0 = 1, IL1 - IL2 >= 2},
    entry3849(OL0,OL1,OL2).
entry3849(IL0,IL1,IL2) :-
    {IL2 = OL2, IL1 = OL1, IL0 = OL0, IL0 >= 1},
    block3853(OL0,OL1,OL2).

L0 is isolated. Taking $x = 1$ shows that L0 is uniform. Hence L0 can be removed.
### Experiments

<table>
<thead>
<tr>
<th>program</th>
<th>meth.</th>
<th>orig</th>
<th>loops</th>
<th>fold</th>
<th>subsum</th>
<th>open</th>
<th>unif</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ack</td>
<td>5</td>
<td>7.11</td>
<td>0.21</td>
<td>0.21</td>
<td>0.21</td>
<td>0.21</td>
<td>0.21</td>
</tr>
<tr>
<td>precision</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BubbleS</td>
<td>5</td>
<td>19.07</td>
<td>1.55</td>
<td>0.71</td>
<td>0.71</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>precision</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>NQueens</td>
<td>222</td>
<td>-</td>
<td>210.31</td>
<td>156.32</td>
<td>92.29</td>
<td>47.77</td>
<td>34.34</td>
</tr>
<tr>
<td>precision</td>
<td>-</td>
<td>171</td>
<td>171</td>
<td>171</td>
<td>171</td>
<td>171</td>
<td>171</td>
</tr>
<tr>
<td>JLex</td>
<td>137</td>
<td>-</td>
<td>228.51</td>
<td>335.85</td>
<td>374.82</td>
<td>121.95</td>
<td>81.21</td>
</tr>
<tr>
<td>precision</td>
<td>-</td>
<td>84</td>
<td>87</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>Kitten</td>
<td>947</td>
<td>-</td>
<td>200.39</td>
<td>226.79</td>
<td>152.47</td>
<td>93.70</td>
<td>79.35</td>
</tr>
<tr>
<td>precision</td>
<td>-</td>
<td>811</td>
<td>827</td>
<td>827</td>
<td>827</td>
<td>827</td>
<td>827</td>
</tr>
</tbody>
</table>
We have presented:

- some termination-equivalent simplifications of the CLP programs that are automatically generated during termination analysis of Java bytecode programs;
- some real case of analysis showing that these simplifications decrease the time for building a proof by some order of magnitude.