Test-Based Inference of Polynomial Loop-Bound Functions

Rody Kersten

Radboud University Nijmegen

9th KeY Symposium, Gernsbach
May 25th, 2010
Presentation Outline

1. Introduction
2. Considered Loops
3. Test-Based Inference Procedure
4. Communication with KeY
5. Prototype
6. Conclusions
Presentation Outline

1. Introduction
2. Considered Loops
3. Test-Based Inference Procedure
4. Communication with KeY
5. Prototype
6. Conclusions
Introduction

Bounding the number of loop iterations is important for:

- WCET analysis
- Heap-space consumption analysis
- Termination analysis
- Loop transforming compiler optimisations
Loop-Bound Function (LBF)

- Expresses an upper bound on the amount of loop iterations depending on (some of) the program variables and data sizes.
- Can be used to bound the number of iterations for arbitrary values of these variables.
Loop-Bound Function: Example

```
1 while (i < 15) {
2   i++;
3 }
```

- The LBF for this loop is $15 - i$
- Can be used to calculate the number of loops for arbitrary $i$
- Proving this bound also proves termination
Quadratic Example

- Using our method, polynomial LBFs can be inferred.
- For instance, the quadratic LBF $x^2 - xi - j + 1$ for the following loop:

```plaintext
while (x>0 && i>0 && i<x && j>0 && j<=x) {
    if (j==x) { i++; j = 0; }
    j++;
}
```
Presentation Outline

1. Introduction
2. Considered Loops
3. Test-Based Inference Procedure
4. Communication with KeY
5. Prototype
6. Conclusions
Considered Loops

We consider loops with conditions in the following form:

\[ C := sC \mid C_1 \land C_2 \mid C_1 \lor C_2 \]
\[ sC := e_1 [\lt, \gt, \le, \ge, \eq, \ne] e_2 \]

- where \( e_i \) are arithmetical expressions
- i.e. propositional logic expressions over numerical (in)equalities
Considered Loops: Example

1 while (start<end && (end<40 || end>100))
2 start++;
The LBF for this loop is a *piecewise* polynomial!

Two reasons:

- As for almost every loop, the LBF has a piece where the bound is 0
- Because of the disjunction in the condition it has two non-null pieces

The LBF for this example is:

\[
\begin{align*}
\text{end} - \text{start} & \quad \text{if} \enspace \text{start} < \text{end} \land \text{end} < 40 \\
\text{end} - \text{start} & \quad \text{if} \enspace \text{start} < \text{end} \land \text{end} > 100 \\
0 & \quad \text{else}
\end{align*}
\]
Presentation Outline

1. Introduction
2. Considered Loops
3. Test-Based Inference Procedure
4. Communication with KeY
5. Prototype
6. Conclusions
Helicopter View

Java source

Test-based inference procedure

Annotated generated method with a chosen loop

External checking tool (KeY)

Verified LBF

Not verifiable automatically
Manual steps

Rejection: repeat testing with a higher degree
Test-Based Approach

- Instrument loop with a counter
- Do test runs for different input values
- Interpolate a polynomial from the results
Test-Based Inference Procedure: Example (1/3)

```java
public void meth(int x, int i, int j) {
    while (x > 0 && i > 0 && i < x && j > 0 && j <= x) {
        if (j==x) { i++; j = 0;}
        j++;
    }
}

public int meth(int x, int i, int j) {
    int count=0;
    while (x > 0 && i > 0 && i < x && j > 0 && j <= x) {
        if (j==x) { i++; j = 0;}
        j++;
        count++;
    }
    return count;
}
```
Test-Based Inference Procedure: Example (2/3)

Test runs

1st group: degree 2 NCA on plane
x=2, i=1, j=1 => count =2
x=3, i=1, j=1 => count=6
x=4, i=1, j=1 => count=12

2nd group: degree 1 NCA on plane
x=3, i=1, j=2 => count=5
x=4, i=1, j=2 => count=11
x=3, i=2, j=2 => count=2

3rd group: degree 0 NCA on plane
x=4 i=1, j=3 => count=10

Degree of a loop bound (e.g. d=2)
Test-Based Inference Procedure: Example (3/3)

Test runs

1\textsuperscript{st} group: degree 2 NCA on plane
\begin{align*}
x=2, \ i=1, \ j=1 & \Rightarrow \text{count} = 2 \\
x=3, \ i=1, \ j=1 & \Rightarrow \text{count} = 6 \\
x=4, \ i=1, \ j=1 & \Rightarrow \text{count} = 12 \\
x=3, \ i=2, \ j=1 & \Rightarrow \text{count} = 3 \\
x=4, \ i=2, \ j=1 & \Rightarrow \text{count} = 8 \\
x=4, \ i=3, \ j=1 & \Rightarrow \text{count} = 4
\end{align*}

2\textsuperscript{nd} group: degree 1 NCA on plane
\begin{align*}
x=3, \ i=1, \ j=2 & \Rightarrow \text{count} = 5 \\
x=4, \ i=1, \ j=2 & \Rightarrow \text{count} = 11 \\
x=3, \ i=2, \ j=2 & \Rightarrow \text{count} = 2
\end{align*}

3\textsuperscript{rd} group: degree 0 NCA on plane
\begin{align*}
x=4, \ i=1, \ j=3 & \Rightarrow \text{count} = 10
\end{align*}

Find the interpolating polynomial and generate the method annotated with the corresponding loop bound:
\[ p(x, i, j) = x^2 - xi - j + 1; \]
A 1-variable polynomial $p(z)$ of degree $d$ can be written as:

$$a_0 + a_1 z + \ldots + a_d z^d = p(z)$$

We need the values of $p(z)$ in $d + 1$ pairwise different points to interpolate.

These form a system of equations with a unique solution.
The system of equations can be written as:

\[
\begin{pmatrix}
1 & z_0 & \cdots & z_0^{d-1} & z_0^d \\
1 & z_1 & \cdots & z_1^{d-1} & z_1^d \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
1 & z_{d-1} & \cdots & z_{d-1}^{d-1} & z_{d-1}^d \\
1 & z_d & \cdots & z_d^{d-1} & z_d^d \\
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{d-1} \\
a_d \\
\end{pmatrix}
=
\begin{pmatrix}
p(z_0) \\
p(z_1) \\
\vdots \\
p(z_{d-1}) \\
p(z_d) \\
\end{pmatrix}
\]

■ Vandermonde determinant

■ Non-zero for pairwise different points \(z_0, \ldots, z_d\)

■ When non-zero, there exists a unique interpolating polynomial
Node Configuration A: 2-dimensional

- The condition under which there exists a unique multivariate polynomial $p(z_1, \ldots, z_k)$ that interpolates multivariate data is not trivial.
- A condition which ensures it is NCA

$N_d^2$ nodes forming a set $W \subset \mathbb{R}^2$ lie in a 2-dimensional NCA if there exist lines $\gamma_1, \ldots, \gamma_{d+1}$ in the space $\mathbb{R}^2$, such that $d + 1$ nodes of $W$ lie on $\gamma_{d+1}$ and $d$ nodes of $W$ lie on $\gamma_d \setminus \gamma_{d+1}$, $\ldots$, and finally 1 node of $W$ lies on $\gamma_1 \setminus (\gamma_2 \cup \ldots \cup \gamma_{d+1})$. 
Typical NCA Instance: Grid
Node Configuration A: k-dimensional

For dimensions $k > 2$ the NCA is defined inductively on $k$.

A set of $N^k_d$ nodes is in NCA in $\mathbb{R}^k$ if and only if

- there is a $(k - 1)$-dimensional hyperplane such that it contains some $N^{k-1}_d$ of the given nodes lying in $(k - 1)$-dimensional NCA for the degree $d$

- For any $0 \leq i \leq d$, there is a $(k - 1)$-dimensional hyperplane such that it contains some $N^{k-1}_{d-i}$ nodes, lying in $(k - 1)$-dimensional NCA for the degree $d - i$, and these nodes do not lie on the previous hyperplanes

- Thus, the remaining 1 node lies on the remaining hyperplane and does not belong to the previous ones
Adding Loop-Conditions

- Test-nodes must not only satisfy NCA-configuration, but also the loop condition.
- When the loop-condition is not satisfied, the loop is executed 0 times.
- This would make any interpolation an incorrect bound.
Example: Grid
Algorithm for Finding Test-Nodes

- Using a global optimisation procedure, find bounding box (within minimal-maximal values)
- Split each dimension into $d + 1$ hyperplanes, i.e. construct grid
- If the nodes on this grid satisfy NCA configuration: done
- Else: increase grid granularity
We consider loop conditions in DNF over arithmetical (in)equalities:
\[
\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} (e_{lij} \ b \ e_{rij}), \text{ with } b \in \{<, >, =, \neq, \leq, \geq\}
\]

For every sub-formula \( \bigwedge_{j=1}^{m_i} (e_{lij} \ b \ e_{rij}) \) we infer and check separately a particular polynomial bound \( p_i(\bar{x}) \)

Altogether they form a piecewise loop-bound function:
\[
\begin{cases}
  p_1(\bar{x}) & \text{if } \bigwedge_{j=1}^{m_1} (e_{l1j} \ b \ e_{r1j})(\bar{x}) \\
  \ldots \\
  p_n(\bar{x}) & \text{if } \bigwedge_{j=1}^{m_n} (e_{lnj} \ b \ e_{rnj})(\bar{x}) \\
  0 & \text{else}
\end{cases}
\]
Dealing with LBFs with Rational or Real Coefficients

1 while (start < end) {
   2      start += 4;
3 }

- The exact number of iterations is $\lceil \frac{end - start}{4} \rceil$
- The bound is a polynomial over rationals
- In general, when the coefficients of the polynomial LBF $p(\bar{z})$ are not naturals, the actual bound should be read as $\lceil p(\bar{z}) \rceil$
Expressing the LBF in JML

```java
public void meth(int x, int i, int j) {
    //assign
    assignable i,j;
    //loop invariant
    loop_invvariant true;
    //decreases
    decreases x*x - x*i - j + 1;
    while (x>0 && i>0 && i<x && j>0 && j<x) {
        if (j==x) { i++; j = 0; }
        j++;
    }
}
```
Prove Using KeY

- Prove that the postcondition holds
  - Therefore we need to prove termination
  - Which means prove that the decreases-clause is correct
  - Because the decreases-clause is $\geq 0$, it forms an upper bound on the number of iterations

- In the simpler (most) cases, KeY can do the proof automatically

- When the decreases-clause is non-linear or contains divisions, manual steps are needed
We have implemented a prototype in Java. It can infer loops and generate annotations. It has been tested on a series of case-studies.
## Case Studies

<table>
<thead>
<tr>
<th></th>
<th>Nr. of loops</th>
<th>Analysable</th>
<th>%</th>
<th>Correct</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hunt et al</td>
<td>2</td>
<td>2</td>
<td>100%</td>
<td>2</td>
<td>100%</td>
</tr>
<tr>
<td>DIANA</td>
<td>4</td>
<td>4</td>
<td>100%</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>CD_\text{x}</td>
<td>38</td>
<td>23</td>
<td>61%</td>
<td>23</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>44</strong></td>
<td><strong>29</strong></td>
<td><strong>66%</strong></td>
<td><strong>29</strong></td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>
Presentation Outline

1. Introduction
2. Considered Loops
3. Test-Based Inference Procedure
4. Communication with KeY
5. Prototype
6. Conclusions
Conclusions

- Novel, general technique for inferring loop-bound functions
- While various other methods for inferring loop-bounds exist, we are not familiar with any other works on generating non-linear loop-bound functions for Java
- Complementary to other methods, since it is more general and can solve certain more complex cases, such as quadratic bounds
- Inferred LBFs are provable!