Bound Analysis of Imperative Programs

Seminar Report

Michael Schaper

Computational Logic
Institute of Computer Science
University of Innsbruck

June 13, 2012

Bibliography

S. Gulwani, K. Mehra and T. Chilimbi
SPEED: Precise and Efficient Static Estimation of Program Computational Complexity
In *Proc. of POPL*, 2009

Motivation

Example

```
void simpleLoop(int n){
  int x = 0;
  int y = n;
  while(x < n){
    x = x + 2;
    y = y * x;
    }
  printf("%d",y);
}
```

Motivation

Complexity

Number of loop iterations of a procedure P, in terms of the size of its input.

Motivation

Complexity

Number of loop iterations of a procedure P, in terms of the size of its input.

precise: precise computational complexity + precise constants

efficient: quadratic (modulo invariant generation) with respect to the number of back-edges

static: based on static analysis

Motivation

Example

```
void simpleLoop(int n){
  int x = 0;
  int y = n;
  while(x < n){
    x = x + 2;
    y = y * x;
  }
    print("%d",y);
}
```

Motivation

Example

```
void simpleLoop(int n){
  int x = 0;

  while(x < n){
    x = x + 2;

  }

}
```

- ignore condition-irrelevant statements (slicing)

Motivation

Example

```
simpleLoop(n)
  x = 0;

  while(x < n)
    x = x + 2;
```

- ignore condition-irrelevant statements (slicing)
- convention: omit types and parenthesis

Motivation

Example

```
simpleLoop(n)
  x = 0;
  c = 0;
  while(x < n)
    x = x + 2;
    c++;
```

- instrument back-edges with counter

Motivation

Example

```
simpleLoop(n)
  x = 0;
  c = 0;
  while(x < n)
    x = x + 2;
    c++;
```

$c \leq \max(0, 1/2 * n)$

- instrument back-edges with counter
- generate loop-invariant

Motivation

But

There is no almighty invariant generator.

Invariant Generation

- abstract interpretation
  - iterative fixed point analysis over abstract domain
- linear arithmetic abstract domain over $\mathbb{R}$
- convex domain (constraints over conjuncts)

Limitation

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0;
  y = y0;
  while(x < n)
    if(y > x)
      x++;
    else
      y++;
```

Limitation

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0;
  y = y0;
  while(x < n)
    if(y > x)
      x++;
    else
      y++;
```

Example (non-linear bound)

```
nonLinear(n,m)
  x = 0;
  while(x < n)
    y = 0;
    x++;
    while(y < m)
      y++;
```

Limitation

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0;
  y = y0;
  while(x < n)
    if(y > x)
      x++;
    else
      y++;
```

Example (non-linear bound)

```
nonLinear(n,m)
  x = 0;
  while(x < n)
    y = 0;
    x++;
    while(y < m)
      y++;
```

Example (data-structures)

```
iterate(List e,f)
  for(e=f;e!=null;e=L.GetNext(e));
```

Key Ideas

- instrumentation of multiple counters
- generation of a linear (invariant) bound for each counter
- composition of generated (linear) bounds

- quantitative functions + effects

Outline

- Proof Structure
- User Defined Data-Structures
- Further Approaches
- Summary

Outline

- Proof Structure
- User Defined Data-Structures
- Further Approaches
- Summary

Instrumentation

Definition

- S set of counter variables
- M mapping from back-edges to counter variables from S
- G *DAG* over $S \cup \{r\}$, where r is the root symbol
- B mapping from back-edges to bounds

Instrumentation

Definition

- S set of counter variables
- M mapping from back-edges to counter variables from S
- G *DAG* over $S \cup \{r\}$, where r is the root symbol
- B mapping from back-edges to bounds

Definition (Instrumentation ($P(S, M, G)$))

- each back-edge q is instrumented with an increment (c++), where $M(q) = c$
- if $(r, c) \in G$, then c is initialized (c=0) at entry point
- if $(c, c') \in G$, then c' is initialized at q, where $M(q) = c$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0;
  y = y0;
  while(x < n)
    if(y > x)
      x++;
```
$q1$
```
    else
      y++;
```
$q2$

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0; c1 = 0;
  y = y0; c2 = 0;
  while(x < n)
    if(y > x)
      x++;
q1    c1++;
    else
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$

Proof Structure

Definition (Proof Structure)

Let P be a procedure then (S, M, G, B) is a proof structure for P, if for all back-edges q in P, the invariant generation tool can establish bound $B(q)$ on counter variable $M(q)$ at q in instrumentation $(P(S, M, G))$.

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0; c1 = 0;
  y = y0; c2 = 0;
  while(x < n)
    if(y > x)
      x++;
q1    c1++;
    else
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0; c1 = 0;
  y = y0; c2 = 0;
  while(x < n)
    if(y > x)
      x++;
q1    c1++;
    else
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$
$$B = \{q1 \mapsto n - x0, q2 \mapsto n - y0\}$$

Computing Bounds

Theorem

Let (S, M, G, B) be a proof structure for P, then U defines an upper bound on the total number of iterations of all loops in P.

$$U = \sum_{c \in S} TotalBound(c)$$

$$TotalBound(r) = 0$$

$$TotalBound(c) = \max(\{0\} \bigcup \{B(q) \mid M(q) = c\}) \times \left(1 + \sum_{(c',c) \in G} TotalBound(c')\right)$$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0; c1 = 0;
  y = y0; c2 = 0;
  while(x < n)
    if(y > x)
      x++;
q1    c1++;
    else
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$
$$B = \{q1 \mapsto n - x0, q2 \mapsto n - y0\}$$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0; c1 = 0;
  y = y0; c2 = 0;
  while(x < n)
    if(y > x)
      x++;
q1    c1++;
    else
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$
$$B = \{q1 \mapsto n - x0, q2 \mapsto n - y0\}$$

$$U = \mathit{TotalBound}(c1) + \mathit{TotalBound}(c2)$$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
  x = x0; c1 = 0;
  y = y0; c2 = 0;
  while(x < n)
    if(y > x)
      x++;
q1    c1++;
    else
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$
$$B = \{q1 \mapsto n - x0, q2 \mapsto n - y0\}$$

$U = \max(0, n - x0) \times (1 + TotalBound(r)) + \max(0, n - y0) \times (1 + TotalBound(r))$

Continued Example

Example (disjunctive bound)

```
disjunctive(x0,y0,n)
   x = x0; c1 = 0;
   y = y0; c2 = 0;
   while(x < n)
      if(y > x)
         x++;
q1       c1++;
      else
         y++;
q2       c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2)\}$$
$$B = \{q1 \mapsto n - x0, q2 \mapsto n - y0\}$$

$$U = \max(0, n - x0) + \max(0, n - y0)$$

Continued Example

Example (non-linear bound)

```
nonLinear(n,m)
  x = 0; c1 = 0; c2 = 0;
  while(x < n)
    y = 0;
    x++;
q1  c1++; c2 = 0;
    while(y < m)
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2), (c1, c2)\}$$
$$B = \{q1 \mapsto n, q2 \mapsto m\}$$

Continued Example

Example (non-linear bound)

```
nonLinear(n,m)
  x = 0; c1 = 0; c2 = 0;
  while(x < n)
    y = 0;
    x++;
q1  c1++; c2 = 0;
    while(y < m)
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2), (c1, c2)\}$$
$$B = \{q1 \mapsto n, q2 \mapsto m\}$$

$$U = \mathit{TotalBound}(c1) + \mathit{TotalBound}(c2)$$

Continued Example

Example (non-linear bound)

```
nonLinear(n,m)
  x = 0; c1 = 0; c2 = 0;
  while(x < n)
    y = 0;
    x++;
q1  c1++; c2 = 0;
    while(y < m)
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2), (c1, c2)\}$$
$$B = \{q1 \mapsto n, q2 \mapsto m\}$$

$$U = \max(0, n) \times (1 + \mathit{TotalBound}(r)) + \max(0, m) \times (1 + \mathit{TotalBound}(r) + \mathit{TotalBound}(c1))$$

Continued Example

Example (non-linear bound)

```
nonLinear(n,m)
  x = 0; c1 = 0; c2 = 0;
  while(x < n)
    y = 0;
    x++;
q1  c1++; c2 = 0;
    while(y < m)
      y++;
q2    c2++;
```

$$S = \{c1, c2\}$$
$$M = \{q1 \mapsto c1, q2 \mapsto c2\}$$
$$G = \{(r, c1), (r, c2), (c1, c2)\}$$
$$B = \{q1 \mapsto n, q2 \mapsto m\}$$

$$U = \max(0, n) + \max(0, m) \times (1 + \max(0, n))$$

Outline

Properties

- iteration over abstract data-structures
- quantitative functions + effects

Properties

- iteration over abstract data-structures
- quantitative functions + effects

- no analysis of heap properties (shape, size, ...)
- reflects user's idea of complexity
- implementation independent
- semi-automatic
- requires support for uninterpreted functions
  - combination of abstract interpretation of linear arithmetic + abstract interpretation of uninterpreted functions

Singly Linked List

- quantitative functions:

$$\begin{aligned} Len(L) &:= \text{length of list } L \\ Pos(e, L) &:= \text{position of element } e \text{ of list } L \end{aligned}$$

- effects:

$$e = \texttt{L.GetNext}(f) := \quad \begin{aligned} & Pos(e, L) = Pos(f, L) + 1; \\ & Assume(0 \leq Pos(f, L) < Len(L)) \end{aligned}$$

Singly Linked List

- quantitative functions:

$$\begin{aligned} Len(L) &:= \text{length of list } L \\ Pos(e, L) &:= \text{position of element } e \text{ of list } L \end{aligned}$$

- effects:

$$e = \texttt{L.GetNext}(f) := \quad \begin{aligned} &Pos(e, L) = Pos(f, L) + 1; \\ &Assume(0 \leq Pos(f, L) < Len(L)) \end{aligned}$$

Example

```
iterate(List e,f)
  for(e=f; e!=null;
      e=L.GetNext(e));
```

$$\begin{aligned} c = &Pos(e, L) - Pos(f, L) \wedge \\ &Pos(e, L) \leq Len(L) \\ U = &Len(L) - Pos(f, L) \end{aligned}$$

Outline

- Proof Structure
- User Defined Data-Structures
- Further Approaches
- Summary

S. Falke and D. Kapur
A Term Rewriting Approach to the Automated Termination Analysis of Imperative Programs
In *Proc. of CADE*, 2009

Example

```
simpleloop(n)
  x = 0;
  while(x < n)
    x++;
```

$$\mathsf{eval}_1(x, y) \rightarrow \mathsf{eval}_2(0, y)$$

$$\mathsf{eval}_2(x, y) \rightarrow \mathsf{eval}_2(x + 1, y)[x < y]$$

- transformation (in a natural way) to $\mathcal{PA}$-based TRSs
- used for proving termination

- notion of complexity for $\mathcal{PA}$-based TRSs
- adaption of existing techniques
- expandable to heap analysis

Outline

Summary

- Microsoft product code + C++ Standard Template Library
- claim: bounds for more than the half of encountered examples

Summary

- Microsoft product code + `C++` Standard Template Library
- claim: bounds for more than the half of encountered examples

- multiple counters + composing of linear invariants
- maximal polynomial bounds
- quantitative functions + effects for abstract data-structures

End

Thank you for your attention!