ABC: Algebraic Bound Computation for Loops

Régis Blanc
EPFL

Thomas A. Henzinger
IST Austria

Thibaud Hottelier
UC Berkeley

Laura Kovács
TU Vienna

What Is ABC?

- Automatically computes an algebraic relation among iteration variables of a loop

What Is ABC?

- Automatically computes an algebraic relation among iteration variables of a loop
- Can then derive iteration bound of the loop

What Is ABC?

- Automatically computes an algebraic relation among iteration variables of a loop
- Can then derive iteration bound of the loop
- Works for any level of nested loops

What Is ABC?

- Automatically computes an algebraic relation among iteration variables of a loop
- Can then derive iteration bound of the loop
- Works for any level of nested loops
- Only works on specific shape of loops

Motivations

- Some real-time systems need to prove an upper bound time of their computation
- Can be used to prove termination

Application: Matrix Linearization

for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $M[i,j] = i + j$
  end for
end for

$\longrightarrow$

for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $A[(i - 1)n + j] = i + j$
  end for
end for

An Example

```
z = 1
for i = 1 to n do
   for j = 1 to n do
      z = z + 1
   end for
end for
```

An Example

$z = 1$
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $z = z + 1$
  end for
end for

- Z-relation:

$$z = (i-1)n + j$$

An Example

$z = 1$
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $z = z + 1$
  end for
end for

- Z-relation:

$$z = (i - 1)n + j$$

- Iteration bound:

$$n^2$$

Another Example

$z = 1$
for $i = 0$; $i \le n$; $i = i + 1$ **do**
  for $j = 0$; $j \le m$; $j = j + 2$ **do**
    $z = z + 1$
  end for
end for

Another Example

$z = 1$
for $i = 0$; $i \leq n$; $i = i + 1$ **do**
  for $j = 0$; $j \leq m$; $j = j + 2$ **do**
    $z = z + 1$
  end for
end for

- Z-relation:

$$z = 1 + \left\lfloor \frac{j}{2} \right\rfloor + i \left(\left\lfloor \frac{m}{2} \right\rfloor + 1 \right)$$

Another Example

$z = 1$
for $i = 0$; $i \leq n$; $i = i + 1$ **do**
  for $j = 0$; $j \leq m$; $j = j + 2$ **do**
    $z = z + 1$
  end for
end for

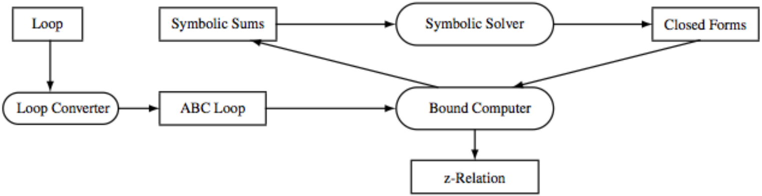
- Z-relation:
$$z = 1 + \left\lfloor \frac{j}{2} \right\rfloor + i\left(\left\lfloor \frac{m}{2} \right\rfloor + 1\right)$$
- Iteration bound:
$$1 + (1 + n)\left\lfloor \frac{m}{2} \right\rfloor + n$$

ABC Structure

ABC relies on three modules:

- Bound Computer
- Loop Converter
- Symbolic Solver

ABC Overall Workflow



Bound Computer

- Works on a special class of loop: ABC-loops
- Computes a polynomial relation over loop variables: z-relation
- Core part of ABC

Bound Computer On An Example

$z = 1$
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    $z = z + 1$
  end for
end for

$\longrightarrow$

$z = 1$
for $i = 1$ to n **do**
  $z = z + \sum_{k=1}^{m} 1$
end for

Bound Computer On An Example

Solving the sum:

```
z = 1
for i = 1 to n do
   for j = 1 to m do
      z = z + 1
   end for
end for
```

$\longrightarrow$

```
z = 1
for i = 1 to n do
   z = z + m
end for
```

Recurrence relation for the value z_i of z at iteration i:

$$z_i = z_{i-1} + m$$

With $z_1 = 1$, we get:

$$z_i = 1 + \sum_{k=1}^{i-1} m = 1 + (i-1)m$$

Bound Computer On An Example

Apply recursively the method on the loop:

$z = 1$
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    $z = z + 1$
  end for
end for

$\longrightarrow$

$z = 1 + (i-1)m$
for $j = 1$ to m **do**
  $z = z + 1$
end for

Recurrence relation of z_j:

$$z_j = z_{j-1} + 1$$

With $z_1 = 1 + (i-1)m$, we get:

$$z_j = 1 + (i-1)m + \sum_{k=1}^{j-1} 1 = (i-1)m + j$$

Bound Computer On An Example

$z = 1$
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    $z = z + 1$
  end for
end for

Thus we obtain the z-relation:

$$z = (i-1)m + j$$

Substituting i by n and j by m, we get the iteration bound:

$$nm$$

Why A Loop Converter?

- The preceding algorithm only works on the ABC-loops.

Why A Loop Converter?

- The preceding algorithm only works on the ABC-loops.
- Transforms, whenever possible, a loop into its equivalent ABC-loop.

Why A Loop Converter?

- The preceding algorithm only works on the ABC-loops.
- Transforms, whenever possible, a loop into its equivalent ABC-loop.
  - Loop that starts from an arbitrary expression

Why A Loop Converter?

- The preceding algorithm only works on the ABC-loops.
- Transforms, whenever possible, a loop into its equivalent ABC-loop.
  - Loop that starts from an arbitrary expression
  - Loop with each of its iteration variables incremented by one

Yet Another Example

$z = 1$
for $i = 0;\ i \leq n;\ i = i + 1$ **do**
  for $j = 0;\ j \leq m;\ j = j + 2$ **do**
    $z = z + 1$
  end for
end for

Yet Another Example

$z = 1$
for $i = 0;\ i \leq n;\ i = i + 1$ **do**
  for $j' = 0;\ j' \leq \left\lfloor \frac{m}{2} \right\rfloor;\ j' = j' + 1$ **do**
    $z = z + 1$
  end for
end for

With:

$$j = 2j'$$

Yet Another Example

$z = 1$
for $i' = 1$; $i' \leq n + 1$; $i' = i' + 1$ **do**
  for $j'' = 1$; $j'' \leq \left\lfloor \frac{m}{2} \right\rfloor + 1$; $j'' = j'' + 1$ **do**
    $z = z + 1$
  end for
end for

With:

$$j = 2j' \quad \wedge \quad j' = j'' - 1 \quad \wedge \quad i = i' - 1$$

Symbolic Solver

- We built our own symbolic solver:
  - Simplifies symbolic expressions
  - Derives closed forms for symbolic sums
- We rely on a simplified version of the Gosper Algorithm

The ABC-Loops

for $(i_1 = 1; i_1 \leq f_1; i_1 = i_1 + 1)$ **do**
  for $(i_2 = 1; i_2 \leq f_2(i_1); i_2 = i_2 + 1)$ **do**
    $\ldots$
    for $(i_n = 1; i_n \leq f_n(i_1, \ldots, i_{n-1}); i_n = i_n + 1)$ **do**
      skip
    end for
    $\ldots$
  end for
end for

Where
$i_1, i_2, \ldots, i_n$ are iteration variables
$f_1, f_2, \ldots, f_n$ are polynomials functions with symbolic constants

Bound Computer Algorithm

Require: ABC-loop F, initial value z_0 of z
Ensure: z-relation $zrel$
$inner :=$ loop_body(F)
$incr :=$ z_reduce_loop($inner$)
$\langle ovar,\ oubound \rangle :=$
$\langle$outer_iteration_variable(F),outer_iteration_upperbound(F)$\rangle$
$nvar :=$ fresh_variable()
$z_i := z_0 +$ solve_sum($nvar$, 1, $ovar - 1$, $incr\left[ovar \mapsto nvar\right]$)
if isloop(inner) **then**
$zrel := z =$Bound Computer($inner$, z_i)
else
$zrel := z = z_i$
end if

Loop Converter Algorithm

Require: For-loop F and $conversion\_list = \{\}$
Ensure: ABC-loop F' and $conversion\_list$

$\langle ovar,\ oincr\rangle := \langle \text{outer\_iteration\_variable}(F),\ \text{outer\_iteration\_increment}(F)\rangle$

$\langle olbound,\ oubound\rangle := \langle \text{outer\_iteration\_lowerbound}(F),$
$\text{outer\_iteration\_upperbound}(F)\rangle$

$nvar := \text{fresh\_variable}()$

$F_0 := loop\_body(F)\big[ovar \mapsto oincr \cdot \big(nvar + olbound - 1\big)\big]$

$conversion\_list := conversion\_list \cup \big\{ovar = oincr \cdot \big(nvar + olbound - 1\big)\big\}$

if isloop(F_0) **then**

$F' := \text{for-loop}(nvar,\ 1,\ \lfloor \frac{oubound - olbound}{oincr} \rfloor + 1,\ 1,\ \text{Loop Converter}(F_0))$

else

$F' := \text{for-loop}(nvar,\ 1,\ \lfloor \frac{oubound - olbound}{oincr} \rfloor + 1,\ 1,\ F_0)$

end if

More General Loops

for $(i_1 = g_1; i_1 \diamond f_1; i_1 = i_1 + r_1)$ **do**
  for $(i_2 = g_2(i_1); i_2 \diamond f_2(i_1); i_2 = i_2 + r_2)$ **do**
    $\ldots$
    for $(i_n = g_n(i_1, \ldots, i_{n-1}); i_n \diamond f_n(i_1, \ldots, i_{n-1}); i_n = i_n + r_n)$ **do**
      skip
    end for
    $\ldots$
  end for
end for

Where
$i_1, \ldots, i_n$ are iteration variables
$r_1, \ldots, r_n$ are symbolics constants
$f_1, \ldots, f_n, g_1, \ldots, g_n$ are polynomials functions with symbolic constants
$\diamond \in \{<, \leq, >, \geq\}$

Symbolic Solver Capabilities

The handled sums are of the following form:

$$\sum_{x=e_1}^{e_2} c_1 \cdot n_1^x \cdot x^{d_1} + \ldots + c_r \cdot n_r^x \cdot x^{d_r}$$

where e_1, e_2 are integer valued symbolic constants, $n_i, d_i \in \mathbb{N}$ and $c_i \in \mathbb{Q}$.

JAMA Package

- We extracted 90 loops from the JAMA package

JAMA Package

- We extracted 90 loops from the JAMA package
- ABC derived the z-relation for 87 of them
- ABC was able to compute complexity for all of them

JAMA Package

- We extracted 90 loops from the JAMA package
- ABC derived the z-relation for 87 of them
- ABC was able to compute complexity for all of them
- All in less than one second

Some Results

Loop	z-relation	Iteration bound	Time [s]
for $(i = 1; i \leq n; i = i + 1)$ **for** $(j = 1; j \leq i; j = j + 1)$ **skip** **end do** **end do**	$z = \frac{i^2 - i}{2} + j$	$\frac{n^2+n}{2}$	0.203
for $(i = 1; i \leq m; i = i + 1)$ **for** $(j = 1; j \leq i; j = j + 1)$ **for** $(k = i + 1; k \leq m; k = k + 1)$ **for** $(l = 1; l \leq k; l = l + 1)$ **skip** **end do** **end do** **end do** **end do**	$z =$ $\frac{i^2m^2 - im^2 + i^2m - im}{4} +$ $\frac{i^2 - i^4}{8} + \frac{i^3 - i}{12} +$ $\frac{jm + jm^2 + k^2}{2} -$ $\frac{m^2 + ji^2 + ji + m + k}{2} + 1$	$\frac{3m^4 + 2m^3 - 3m^2 - 2m}{24}$	1.281
for $(i = 0; i \leq (\frac{n*n*n}{2} - 1); i = i + 1)$ **for** $(j = 0; j \leq n - 1; j = j + 1)$ **for** $(k = 0; k \leq j - 1; k = k + 1)$ **skip** **end do** **end do** **end do**	$z =$ $1 + k + \frac{in^2 - in + j^2 - j}{2}$	$\frac{n^5 - n^4}{4}$	0.234

Some Other Results

Loop	z-relation	Iteration bound	Time [s]
for $(i = n; i \geq 1; i = i - 1)$ **for** $(j = m; j \geq 1; j = j - 1)$ **skip** **end do** **end do**	$z = 1 - j + (n - i + 1)m$	nm	0.187
for $(i = a; i \leq b; i = i + 1)$ **for** $(j = c; j \leq d; j = j + 1)$ **for** $(k = i - j; k \leq i + j; k = k + 1)$ **skip** **end do** **end do** **end do**	$z = 1 - 2ad + 2id - ad^2 + id^2 + ac^2 - ic^2 + j^2 - c^2 + j - a + k$	$(c^2 - (d+1)^2)(a - b - 1)$	0.328
for $(i = n; i \geq 1; i = i - 1)$ **for** $(j = 1; j \leq m; j = j + 1)$ **for** $(k = i; k \leq i + j; k = k + 1)$ **for** $(l = 1; l \leq k; l = l + 1)$ **skip** **end do** **end do** **end do** **end do**	$z = -\frac{m^2+3m+2}{4} i^2 + (\frac{j^2+j-1}{2} - \frac{2m^3+9m^2+13}{12})i + \frac{k^2-k}{2} + \frac{j^3-j}{6} + 1 + \frac{2m^2+3mn+9m+9n+13}{12} mn$	$\frac{2m^2+3mn+9m+9n+13}{12} mn$	0.625

Related Work

- User-defined invariant templates [Seidl04]

  → invariants: constraint solving over template coefficients;

- User-defined atomic predicates and loop patterns [Gulwani10]

  → bounds: control-flow refinement and abstract interpretation;

- Recurrence Solving [van Egelen00, Albert08, Valigator08]

  → bounds: unfolding loops with simple but non-deterministic recurrences;

  → bounds: pattern matching simple class of recurrences;

  → bounds: quantifier elimination for unnested loops and non-initializing assignments;

- WCET [aiT04, TUBound09]

  → bounds: interval-based abstract interpretation with unrollings of simple-loops;

  → bounds: solving constraints over variables from linear loop tests.

Conclusion And Future Work

- ABC automatically computes algebraic loop bounds
- ABC is available at http://mtc.epfl.ch/software-tools/ABC

Conclusion And Future Work

- ABC automatically computes algebraic loop bounds
- ABC is available at http://mtc.epfl.ch/software-tools/ABC
- Extend ABC to handle more complex loops and symbolic sums