Rank: A Tool to Check Program Termination and Computational Complexity

Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord

ENS-Lyon, CNRS, Inria, UCBL
Laboratoire de l’Informatique du Parallélisme

CSTVA 2013, March 22, 2013
Motivation: scheduling irregular programs

```
// expression expr,
// array A,
// r>0 integer.
da = 2r; db = 2r;
while (da >= r) {
    cond = ( da >= db ||
        A[expr] == 0 );
    if (!cond) {
        tmp = db;
        db = da;
        da = tmp - 1;
    } else da = da - 1;
}
```

- Important analysis for compiler optimizations
- Resolved for regular programs since 90s (polyhedral model).
- By-product: program termination and time complexity.
Contributions

Program termination with multi-dimensional affine ranking (schedule)

- Proven to be complete, fully implemented.

Success: worst-case computational complexity (WCCC).

Failure: non-termination counter example.

Any program can fit (with a proper preprocessing...):
  - Multiple loops/ifs of arbitrary nesting patterns,
  - Premature termination and gotos
  - Nondeterministic choices and values
Outline

Step 1: Integer interpreted automaton

- Abstractions, non-determinism handling

Step 2: Invariants

- Linear relation analysis (abstract interpretation)

Step 3: Ranking (+ WCCC/non-termination)

- Scheduling with Farkas lemma + counting points in polyhedra
Step 1: Integer interpreted automaton

// expression expr,
// array A,
// r>0 integer.
da = 2r; db = 2r;
while (da >= r) {
    cond = ( da >= db ||  
             A[expr] == 0 );
    if (!cond) {
        tmp = db;
        db = da;
        da = tmp - 1;
    }
    else da = da - 1;
}

- Build (and invert) the control-flow graph.
- Select the control-impacting part with program slicing.
- Abstract non-affine conditions \( \Rightarrow \) non-determinism.
- Refine automaton with path compression.

Tool C2fsm.
Step 2: Invariants

A la Cousot-Halbwachs linear relation analysis, with abstract interpretation:

System of equations:

\[
\begin{align*}
P_{\text{init}} &= \{1 \leq r\} \\
P_{\text{loop}} &= P_{\text{init}} \cup t_2(P_{\text{loop}}) \cup t_3(P_{\text{loop}}) \\
P_{\text{stop}} &= t_4(P_{\text{loop}})
\end{align*}
\]

- Each invariant over-approximates the reachable values on a control point \( k \) by a polyhedron \( P_k \subset \mathbb{Z}^n \).
- Possibly infinite, parameterized by program inputs.

Tool Aspic.
Map each state $s$ to a rank $\rho(s) \in (\mathcal{W}, <)$:
- $(\mathcal{W}, <)$ is well-founded.
- The rank decreases along each transition $t$: $(s, s') \in t \rightarrow \rho(s') < \rho(s)$.

**Multidimensional affine ranking functions:**
- $\mathcal{W} = \mathbb{N}^n$ with lexicographic order $\ll$.
- $\rho(K, i, N) = A_K (i \ N)^T + b_K$

$n>1$ needed when the complexity is superlinear.

Found thanks to Farkas lemma and ILP.
Finding a ranking function - 1D case (n=1)

```
assume(N>0);
i=N;
while(i>0) --i;
```

We find:

- state start: $2+N\_o$
- state W: $1+i$
- state stop: $0$
Finding a ranking function - nD case (n>1)

```c
//N>0
i = N;
while(i>0)
{
    j = N;
    while(j>0) j--;
    i--;
}
```

**Greedy algorithm:**
For each dimension: solve $(\rho \downarrow)$ as much transitions as possible. 
\(\rho\) constant on unsolved transitions.
Example

//N>0
i = N;
while(i>0){
    j = N;
    while(j>0) j--;
    i--;
}

Invariant for whiles:

\[-1 < i \leq N, -1 < j \leq N, N > 0\]
Example

//N>0
i = N;
while(i>0){
    j = N;
    while(j>0) j--;
    i--;
}

Invariant for whiles:

\[-1 < i \leq N, -1 < j \leq N, N > 0\]
Example

//N>0
i = N;
while(i>0){
    j = N;
    while(j>0) j--;
    i--;
}

Invariant for whiles:

\[ -1 < i \leq N, -1 < j \leq N, N > 0 \]
// N > 0
i = N;
while (i > 0) {
    j = N;
    while (j > 0) j--;
    i--;
}

Invariant for whiles:

-1 < i ≤ N, -1 < j ≤ N, N > 0
A non-deterministic example

\[ y = 0; \; x = 0; \]
\[ \text{while } (x \leq N \&\& y \leq N) \{ \]
\[ \quad \text{if (?) \{} \]
\[ \quad \quad x=x+1; \]
\[ \quad \quad \text{while } (y \geq 0 \&\& ?) \; y=y-1; \]
\[ \quad \}\]
\[ y=y+1; \]
\[ \}\]

y
x(0,0)
(N,N)

Terminates in \( O(N^2) \) steps.
A non-deterministic example

\[
y = 0; \ x = 0;
while (x <= N && y <= N) {
\quad \text{if (?) {}
\quad \quad x = x+1;
\quad \quad \text{while (y >= 0 && ?) y=y-1;}
\quad }
\quad y = y+1;
}\]

\[
\begin{align*}
\sigma_1(a, x, y) &= -2x + 2N \\
\sigma_1(b, x, y) &= -2x + 1 + 2N \\
\sigma_1(c, x, y) &= -2x + 2N \\
\sigma_2(a, x, y) &= -2y + 1 + 2N \\
\sigma_2(b, x, y) &= y \\
\sigma_2(c, x, y) &= -2y + 2N
\end{align*}
\]
A non-deterministic example

\[ y = 0; \ x = 0; \]

\[
\text{while} \ (x \leq N \ \&\& \ y \leq N) \ \{ \\
\quad \text{if} \ (?) \ { \\
\quad \quad x=x+1; \\
\quad \quad \text{while} \ (y \geq 0 \ \&\& \ ?) \ y=y-1; \\
\quad \} \\
\quad y=y+1; \\
\}
\]

Terminates in \( O(N^2) \) steps.

\[
\begin{align*}
\sigma_1(a, x, y) &= -2x + 2N \\
\sigma_1(b, x, y) &= -2x + 1 + 2N \\
\sigma_1(c, x, y) &= -2x + 2N \\
\sigma_2(a, x, y) &= -2y + 1 + 2N \\
\sigma_2(b, x, y) &= y \\
\sigma_2(c, x, y) &= -2y + 2N
\end{align*}
\]
Step 4: Worst-case computational complexity

Worst-case computational complexity (WCCC): maximum number of transitions fired by the automaton:

$$WCCC \leq \# \bigcup \sigma(k, P_k) \leq \sum_k \# \sigma(k, P_k)$$

Counting points in (images of) polyhedra: Ehrhart polynomials, projections, Smith form, union of polyhedra, etc.

$$WCCC \leq \# \sigma(\text{init}, P_{\text{init}}) + \# \sigma(\text{loop}, P_{\text{loop}}) + \# \sigma(\text{end}, P_{\text{end}})$$

$$= 2 + \# \{(1, i) \mid 1 \leq i \leq 2r + 2\} = 2r + 4$$
Step 4: Detection of infinite loops

For each cycle with (affine) relations $R_1, \ldots, R_n$, solve:

$$(\vec{x}_0 R_1 \vec{x}_1) \land (\vec{x}_1 R_2 \vec{x}_2) \land \ldots \land (\vec{x}_{n-1} R_n \vec{x}_n) \land \vec{x}_0 = \vec{x}_n$$

Possible false positive because of invariant approximation.
**Tools**

**C2fsm:** C program $\rightarrow$ integer interpreted automaton.

**Aspic:** invariants for each control point.

**Rank:** multidimensional affine ranking function.
- **Success:** (symbolic) worst-case computational complexity
- **Failure:** non-terminating counter example (sometimes).

Online tool demo: [http://compsys-tools.ens-lyon.fr/rank](http://compsys-tools.ens-lyon.fr/rank)
Conclusions

- Automatic construction of multi-dimensional ranking functions — greedy, but complete.
- Any program can fit with a proper preprocessing.
- Automatic derivation of worst-case computational complexity. — no additional restrictions on the program are necessary.
- Strong link with computation models, theoretical results and tools developed by automatic parallelization and HPC community.
- Method fully implemented, promising experimental results.
Flow graph construction

- Better approximation
  - Handling mods and divs
  - Handling context-dependent constraints (e.g. $\bot^2 \geq 0$).
- State refinement (booleans and enums)
  - trade-off state increasing / more affine rankings.

Ranking functions

- Decrease on cut points only (to improve the WCCC)
- More complex shapes are needed (e.g. piece-wise affine)
  - What states to split, and how?
Toy examples:

<table>
<thead>
<tr>
<th>Name</th>
<th>Result</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminate</td>
<td>Terminates</td>
<td>non constant loop bound</td>
</tr>
<tr>
<td>nestedLoop</td>
<td>Terminates</td>
<td>3 nested loops</td>
</tr>
<tr>
<td>random2d</td>
<td>Terminates</td>
<td>Non deterministic</td>
</tr>
<tr>
<td>maccarthy91</td>
<td>Terminates</td>
<td>complex loop</td>
</tr>
<tr>
<td>wise</td>
<td>Terminates</td>
<td>disjunctive form for complexity</td>
</tr>
<tr>
<td>speedFails4</td>
<td>Terminates</td>
<td>Speed tool fails on it</td>
</tr>
<tr>
<td>pgcd</td>
<td>DK</td>
<td>rank shows an infinite loop</td>
</tr>
</tbody>
</table>
## Sorting arrays:

<table>
<thead>
<tr>
<th>Name</th>
<th>LOCs</th>
<th>Time(c2fsm/analysis)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>selection</td>
<td>20</td>
<td>1.0/0.4</td>
<td>$\frac{N^2}{2} + \frac{3N}{2} + 1$</td>
</tr>
<tr>
<td>insertion</td>
<td>12</td>
<td>0.6/0.22</td>
<td>$\frac{N^2}{2} + \frac{3N}{2} + 1$</td>
</tr>
<tr>
<td>bubble</td>
<td>22</td>
<td>1.2/0.4</td>
<td>$N^2 + 2$</td>
</tr>
<tr>
<td>shell</td>
<td>23</td>
<td>1.0/1.1</td>
<td>$\frac{N^3}{6} - \frac{N}{6}$</td>
</tr>
<tr>
<td>heap</td>
<td>45</td>
<td>3.0/2.8</td>
<td>$4N^2 - 11N + 9$</td>
</tr>
</tbody>
</table>

▶ More examples on:  
http://compsys-tools.ens-lyon.fr/rank
i=0;
while(i<N)
{
    i = i + 1;
}

We expect loop \( \mapsto (1, 2(N - i)) \), body \( \mapsto (1, 2(N - i) - 1) \).
But... the unknown sign of \( N \) prevents to conclude.

\[\begin{align*}
\rho(\text{loop}, i, N) &= \begin{cases} 
N \geq 0 : & (1, 2(N - i)) \\
N < 0 : & (1) \end{cases} \\
\rho(\text{body}, i, N) &= (1, 2(N - i) - 1)
\end{align*}\]

Sometimes, this situation can be avoided thanks to cut points.
Definition 1

A set of cut points is a minimum subset of control points such that:
If we remove the cut points, the flow-graph becomes acyclic.

- Each cut-point “represents” a loop.
- It suffices to compute the rankings functions on the cut points.

Direct approach: Restrict the flow graph to the cut-points. Then, compute the ranking functions.
assume(N>0);
i=N;
while(i>0) --i;

Searching for:
\[
\rho(\text{start}, i, N) = \alpha_{\text{start}} \cdot i + \beta_{\text{start}} \cdot N + \gamma_{\text{start}}
\]
\[
\rho(W, i, N) = \alpha_W \cdot i + \beta_W \cdot N + \gamma_W
\]
\[
\rho(\text{stop}, i, N) = \alpha_{\text{stop}} \cdot i + \beta_{\text{stop}} \cdot N + \gamma_{\text{stop}}
\]

The constraints are:
- For each pc: \( \rho(pc, \vec{x}) \geq 0 \) on \( P_{pc} \)
- For each transition
  \( (\vec{x}', \vec{x}) \in t \Rightarrow \rho(dest, \vec{x}') - \rho(src, \vec{x}') > 0 \)
Finding a ranking function - 1D case (n=1)

Encoding into a linear programming problem!

1. Constraints $\rho(pc, \bar{x}) \geq 0 \ \forall \bar{x} \in P_{pc}$:
   - Invariant for $W = \{N - 1 \geq 0, i \geq 0, N - i \geq 0\}$
   - Farkas lemma: $\exists \lambda_W \geq 0$ s.t.:

     $$
     \rho(W, i, N) = \lambda^0_W + \lambda^1_W(N - 1) + \lambda^2_W i + \lambda^3_W(N - i)
     $$

   + Affine form for $\rho(W, \bar{x})$:

     $$
     \rho(W, i, N) = \alpha_W i + \beta_W N + \gamma
     $$

   ▶ Identifying $i$: $\alpha^1_W = \lambda^4_W - \lambda^3_W$, etc

2. Constraints for decreasing transitions: similar, as:

   $$
   \rho(x) - \rho(x') - 1 \geq 0, \ \forall (x, x') \in t
   $$
Replace non-affine constructions by $\bot$, then normalize.

Non-affine expressions: $\bot \leadsto \text{random value}$.

Non-affine conditions: $\bot \leadsto \text{non-deterministic condition}$.

// integer x, x0
// array A
x = x0;
if(x>0 && x*x < 2 && A[x] == 0)
  { x = x + 1; }
else
  { x = x + 2; }