Rank: A Tool to Check Program Termination and Computational Complexity

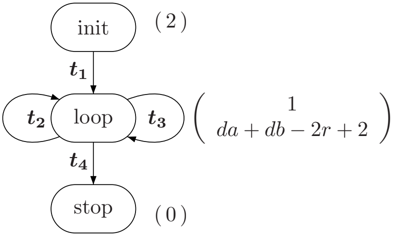
Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord

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

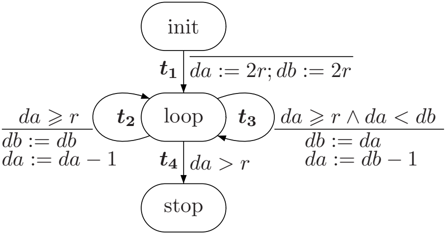
- Linear relation analysis (abstract interpretation)

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 ➠non-determinism. ▸ Details
- Refine automaton with path compression.

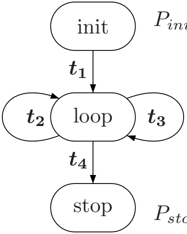
☛ Tool C2fsm.

Step 2: Invariants

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

System of equations:

$$\begin{cases} P_{init} & = & \{1 \leq r\} \\ \boldsymbol{P_{loop}} & = & P_{init} \cup t_2(\boldsymbol{P_{loop}}) \\ & & \quad \cup t_3(\boldsymbol{P_{loop}}) \\ P_{stop} & = & t_4(P_{loop}) \end{cases}$$



$$P_{init} = 1 \leqslant r$$

$$P_{loop} = \begin{cases} r - 1 \leqslant da \leqslant 2r \\ r \leqslant db \leqslant 2r \\ 1 \leqslant r \end{cases}$$

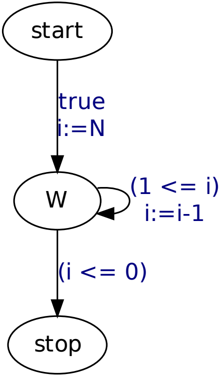
$$P_{stop} = \begin{cases} 1 \leqslant r \\ r \leqslant db \leqslant 2r \\ r - 1 \leqslant da < r \end{cases}$$

- Each invariant over-approximate 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.

Step 3: Ranking

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



Map each state s to a rank $\rho(s) \in (\mathcal{W}, <)$:

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

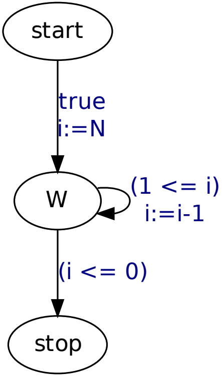
Multidimensionnal affine ranking functions:

- $\mathcal{W} = \mathbb{N}^n$ with lexico. 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.

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



We find :

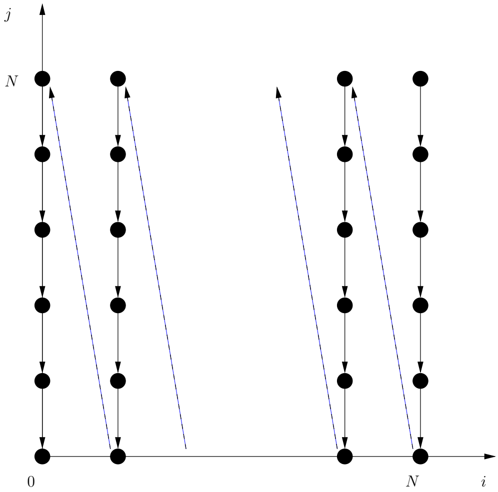
```
state start:
2+N__o

state W:
1+i

state stop:
0
```

Finding a ranking function - nD case (n>1)

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



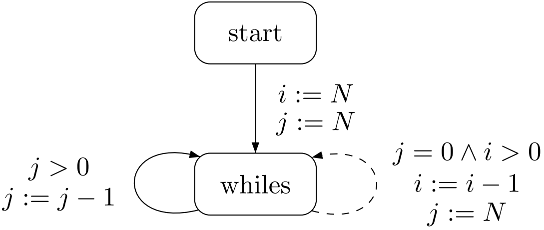
Greedy algorithm:

For each dimension: solve ($\rho \downarrow$) as much transitions as possible.

ρ constant on unsolved transitions.

Example

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

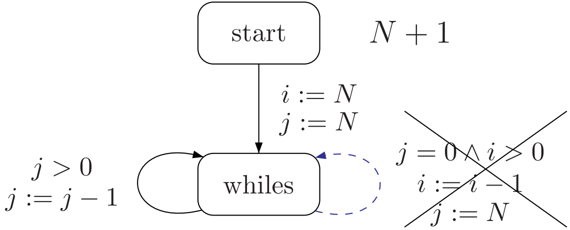


Invariant for `whiles` :

$$-1 < i \leq N, -1 < j \leq N, N > 0$$

Example

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



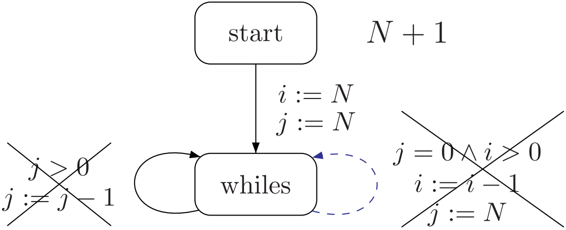
$\varepsilon = \mathbf{0}$ $\quad (i,$ $\quad \varepsilon > \mathbf{0}$

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--;
}
```



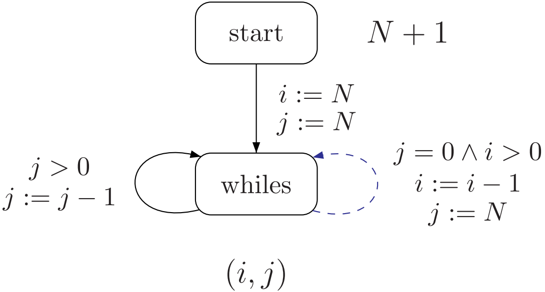
$\varepsilon > \mathbf{0}$ $\quad (i, j)$

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--;
}
```

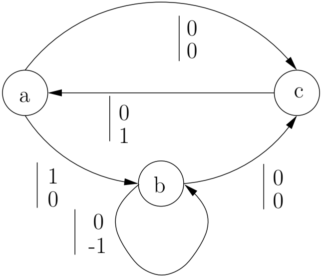


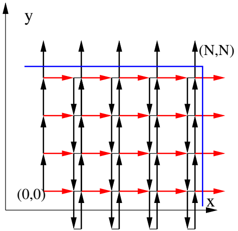
Invariant for `whiles` :

$$-1 < i \leq N, -1 < j \leq N, N > 0$$

A non-deterministic example

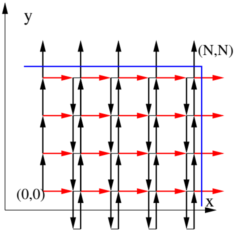
```
y = 0; x = 0;
while (x <= N && y <= N) {
  if (?) {
    x=x+1;
    while (y >= 0 && ?) y=y-1;
  }
  y=y+1;
}
```

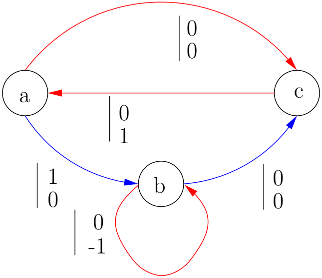




A non-deterministic example

```
y = 0; x = 0;
while (x <= N && y <= N) {
  if (?) {
    x=x+1;
    while (y >= 0 && ?) y=y-1;
  }
  y=y+1;
}
```



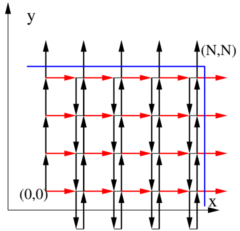


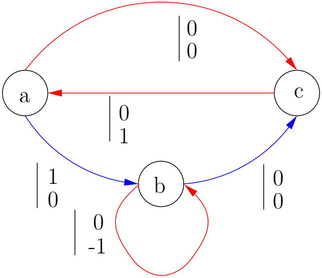
$$\begin{cases} \sigma_1(a,x,y) = -2x + 2N \\ \sigma_1(b,x,y) = -2x + 1 + 2N \\ \sigma_1(c,x,y = -2x + 2N \end{cases}$$

$$\begin{cases} \sigma_2(a,x,y) = -2y + 1 + 2N \\ \sigma_2(b,x,y) = y \\ \sigma_2(c,x,y = -2y + 2N \end{cases}$$

A non-deterministic example

```
y = 0; x = 0;
while (x <= N && y <= N) {
  if (?) {
    x=x+1;
    while (y >= 0 && ?) y=y-1;
  }
  y=y+1;
}
```





$$\begin{cases} \sigma_1(a, x, y) = -2x + 2N \\ \sigma_1(b, x, y) = -2x + 1 + 2N \\ \sigma_1(c, x, y = -2x + 2N \end{cases}$$

$$\begin{cases} \sigma_2(a, x, y) = -2y + 1 + 2N \\ \sigma_2(b, x, y) = y \\ \sigma_2(c, x, y = -2y + 2N \end{cases}$$

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

Step 4: Worst-case computational complexity

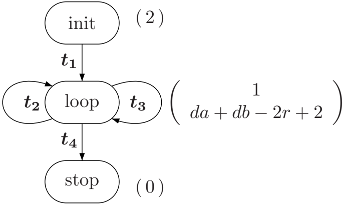
Worst-case computational complexity (WCCC): maximum number of transitions fired by the automaton:

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

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

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

$$= 2 + \#\{(1, i) \mid 1 \leq i \leq 2r + 2\}$$
$$= 2r + 4$$

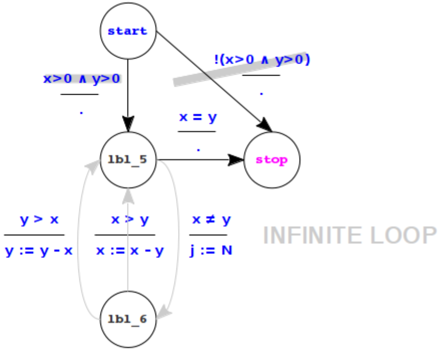


Step 4: Detection of infinite loops

```
int x,y;

if(x<=0 || y<=0)
  return;

while(x != y)
  if(x<y)
    y = y-x;
  else
    x = x-y;
```



For each cycle with (affine) relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$, solve:

$$(\vec{x}_0\ \mathcal{R}_1\ \vec{x}_1)\ \wedge (\vec{x}_1\ \mathcal{R}_2\ \vec{x}_2)\ \wedge\ \ \ldots\ \ \wedge (\vec{x}_{n-1}\ \mathcal{R}_n\ \vec{x}_n)\ \wedge\ \vec{x}_0 = \vec{x}_n$$

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`

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.

Perspectives

Flow graph construction

- Better approximation
  - Handling mods and divs
  - Handling context-dependent constraints (e.g. $\perp^2 \geq 0$).
- 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 ?

Benchmarks 1/2

Toy examples :

Name	Result	Caracteristics
terminate	Terminates	non constant loop bound
nestedLoop	Terminates	3 nested loops
random2d	Terminates	Non deterministic
maccarthy91	Terminates	complex loop
wise	Terminates	disjunctive form for complexity
speedFails4	Terminates	Speed tool fails on it
pgcd	DK	rank shows an infinite loop

Benchmarks 2/2

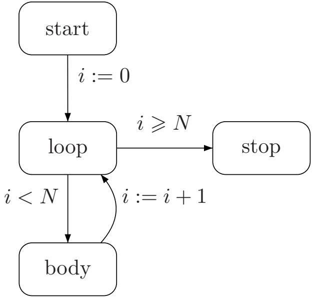
Sorting arrays :

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

▶More examples on :
http://compsys-tools.ens-lyon.fr/rank

Improvements, I

```
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.
➠ A piece-wise affine ranking is required:

$$\rho(loop, i, N) = \begin{cases} N \geq 0 : & (1, 2(N-i)) \\ N < 0 : & (1) \end{cases}$$
$$\rho(body, i, N) = (1, 2(N-i)-1)$$

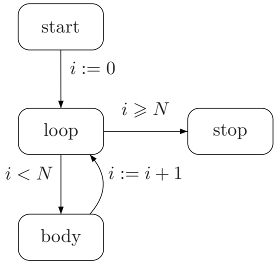
Sometimes, this situation can be avoided thanks to cut points.

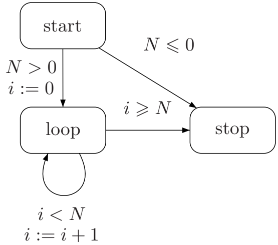
Definition 1

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.

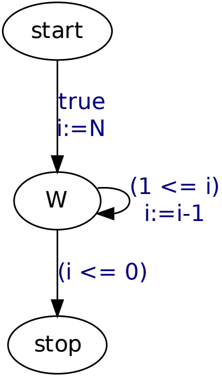
Direct approach: Restrict the flow graph to the cut-points.
Then, compute the ranking functions.





Finding a ranking function - 1D case (n=1)

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



Searching for :

$$\begin{aligned}\rho(start, i, N) &= \alpha_{start}.i + \beta_{start}.N + \gamma_{start}\\ \rho(W, i, N) &= \alpha_W.i + \beta_W.N + \gamma_W\\ \rho(stop, i, N) &= \alpha_{stop}.i + \beta_{stop}.N + \gamma_{stop}\end{aligned}$$

The constraints are :

- For each pc : $\rho(pc, \vec{x}) \geq 0$ on P_{pc}
- For each transition $(\vec{x'} - \vec{x}) \in t \Rightarrow \rho(dest, \vec{x'}) - \rho(src, \vec{x'}) > 0$

Encoding into a linear programming problem !

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

$$\rho(W, i, N) \quad = \quad \lambda_W^0 + \lambda_W^1.(N-1) + \lambda_W^2.i + \lambda_W^3.(N-i)$$

   +affine form for $\rho(W, \vec{x})$:

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

   ▶Identifying i : $\alpha_W^1 = \lambda_W^4 - \lambda_W^3$, etc
2. Constraints for decreasing transitions : similar, as: $\rho(x) - \rho(x') - 1 \geq 0, \ \forall (x, x') \in t$

Back

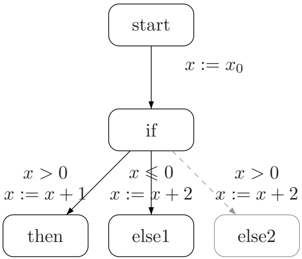
Approximations

Replace non-affine constructions by $\perp$, then normalize.

Non-affine expressions: $\perp \rightsquigarrow$ random value.

Non-affine conditions: $\perp \rightsquigarrow$ 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;
  }
```



Back