Synthesis of ranking functions using extremal counterexamples

Laure Gonnord David Monniaux **Gabriel Radanne**

November, 4th 2015

Why ?

Our goal:

- Prove termination of sequential programs.

Why ?

Our goal:

- Prove termination of sequential programs.

Why ?

Our goal:

- Prove termination of **some** sequential programs.

Why ?

Our goal:

- Prove termination of **some** sequential programs.
- Make an algorithm capable of working on big examples.

Why ?

Our goal:

- Prove termination of **some** sequential programs.
- Make an algorithm capable of working on big examples.
- For Safety and Performance

Goal : Safety

Prove that (some) loops terminate:

```
int main () {
    unsigned int i,j ;
    i=42 ; j=1515 ;
    while(i>0)i-- ;  ✔
    while(j>=0)j++ ;  ✘
}
```

- Fight against bugs.

Goal : Optimisation

Prove that (some) loops terminate:

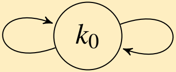
```
int main () {
    unsigned int i,j ;
    i=42 ; j=1515 ;
    while(i>0) i-- ; ✔
    foo(j);
}
```

- Code motion (compiler optimisation).

Contributions

- A technique to prove that (some) loops terminate:
  - Automatic generation of **ranking functions**
  - Based on Linear Programming.
  - Focus on scalability: incremental construction of LP instances.
- Implemented as a standalone tool: TERMITE
  - Capable of proving 119 on 129 programs of TERMCOMP benchmark.
  - Competitive with other state-of-the-art tools.
  - Publicly available on github.

$$\boldsymbol{t_2} : \frac{0 \le x \land 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}}$$

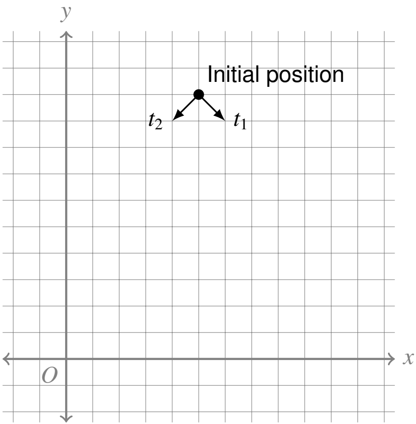


$$\boldsymbol{t_1} : \frac{x \le 10 \land 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

$$t_2 : \frac{0 \leq x \wedge 0 \leq y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \quad \circlearrowleft k_0 \circlearrowright \quad t_1 : \frac{x \leq 10 \wedge 0 \leq y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

The Initial position:

$x = 5$ and $y = 10$

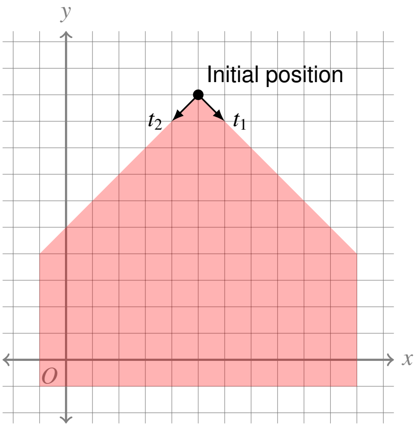


$$t_2 : \frac{0 \leq x \wedge 0 \leq y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \leq 10 \wedge 0 \leq y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

The Initial position:

$x = 5$ and $y = 10$

The invariant $\mathcal{I}$.

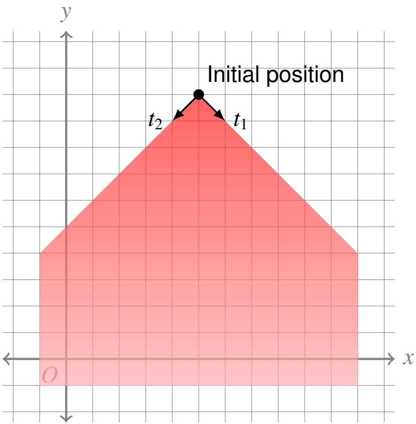


$$t_2 : \frac{0 \le x \wedge 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \le 10 \wedge 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

A linear ranking function:

$$\rho(x, y) = y + 1$$

- Linear
- Decreasing along the transitions
- Positive on $\mathcal{I}$

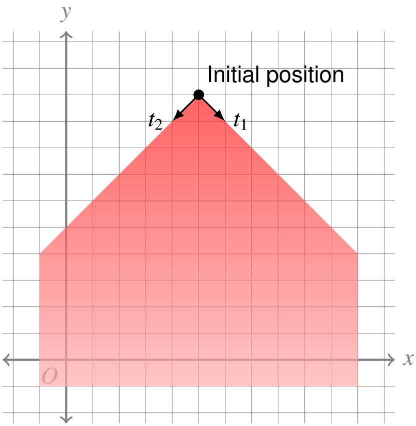


$$t_2 : \frac{0 \le x \wedge 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \le 10 \wedge 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

A linear ranking function:

$$\rho(x, y) = y + 1$$

- Linear
- Decreasing along the transitions
- Positive on $\mathcal{I}$
- **Strict**: decreasing by ≥ 1.



Synthesis of ranking functions

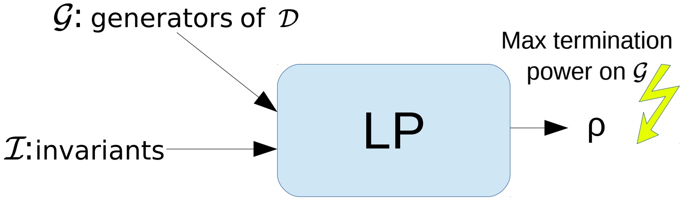
- We want to find a linear ranking function.
  - Linear
  - Decreasing along the transitions
  - Positive on $\mathcal{I}$
- In programs with one control point, for now.
- When transitions are **linear**.
- A **maximally strict** one.
  - Decreases by at least one in as many transitions as possible.

Solving the problem

Let's consider $\mathcal{D}$ the set of reachable one-step differences:

$$\mathcal{D} = \{\boldsymbol{x} - \boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{I}, (\boldsymbol{x}, \boldsymbol{x}') \in \tau\}$$

Thanks to linearity + Farkas' Lemma we are able to define:



- ρ positive, decreasing on $\mathcal{G}$, and **stricly decreasing on a maximal subset of $\mathcal{G}$**

Existing techniques: drawbacks / solutions

Existing techniques: build a system of constraints and solve:

$$Size = O(\#vars \times \#Bblocks \times \#transitions)$$

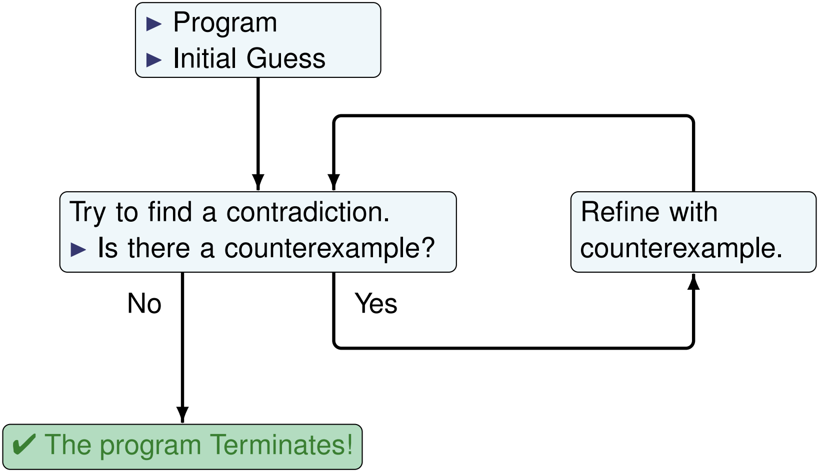
- scalability: all basic blocks $\leadsto$ big constraint systems
- precision: ρ must decrease at **each** transition.

Our technique:

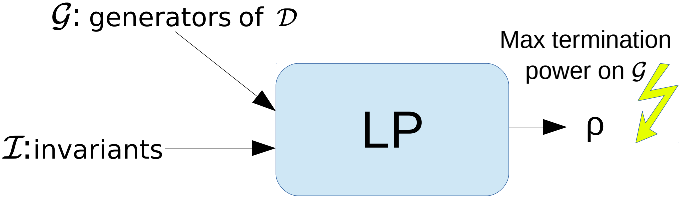
- only considers **a cut-set** of basic blocks.
- considers loops as single transitions.

► **We do not compute all paths** explicitly (CEGAR-based algorithm).

Our key insight : incremental generation of constraints

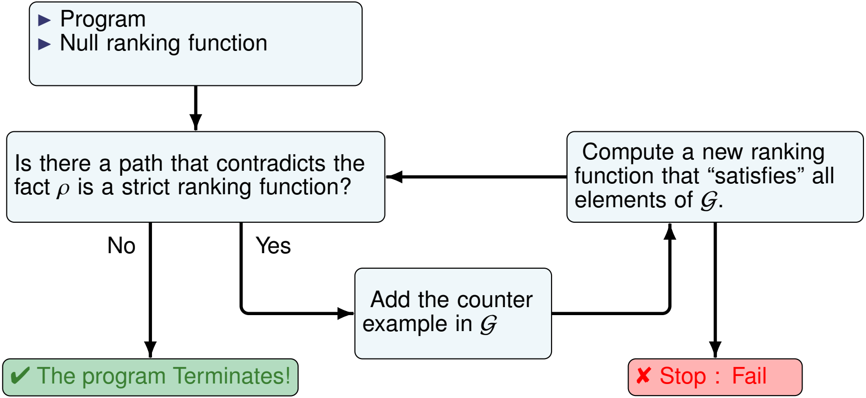


Solving the problem

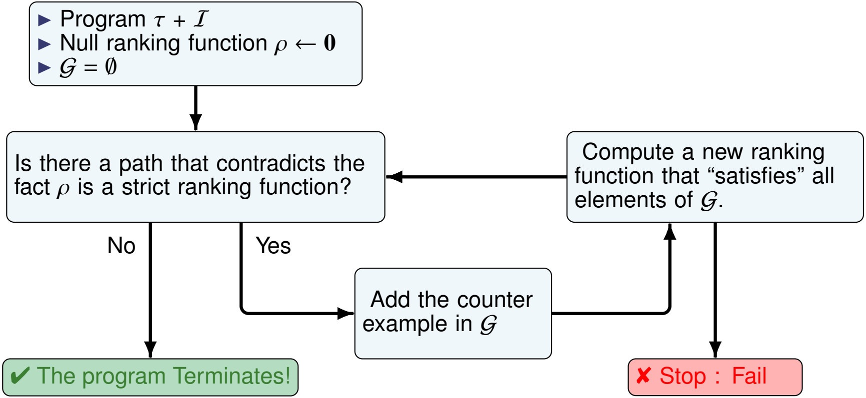


- We construct $\mathcal{G}$ lazily.

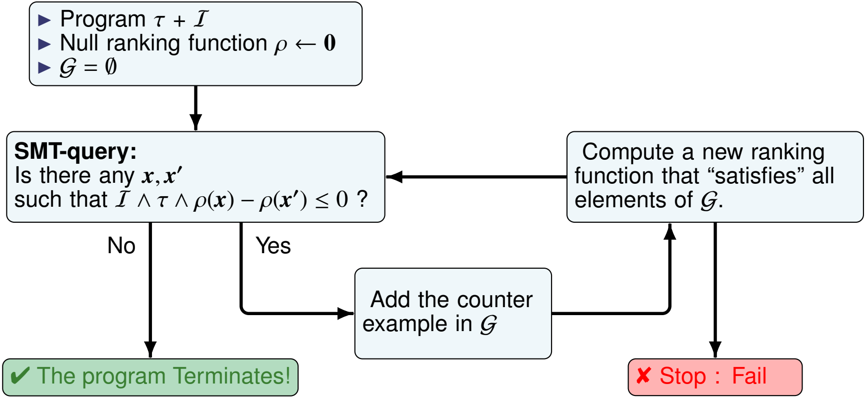
Simple algorithm for one control point



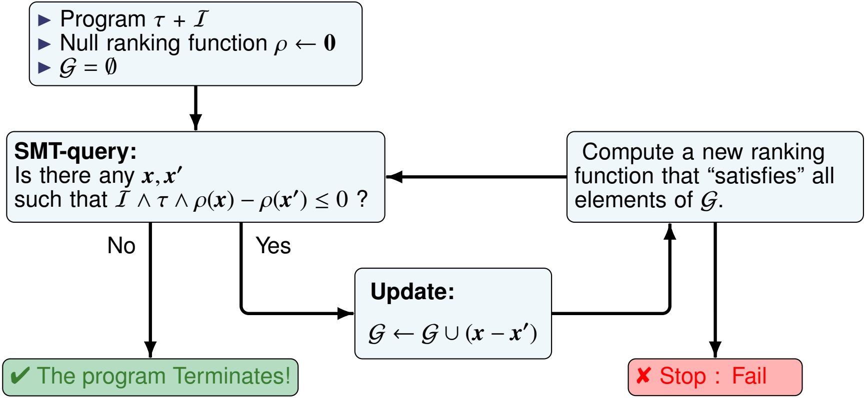
Simple algorithm for one control point



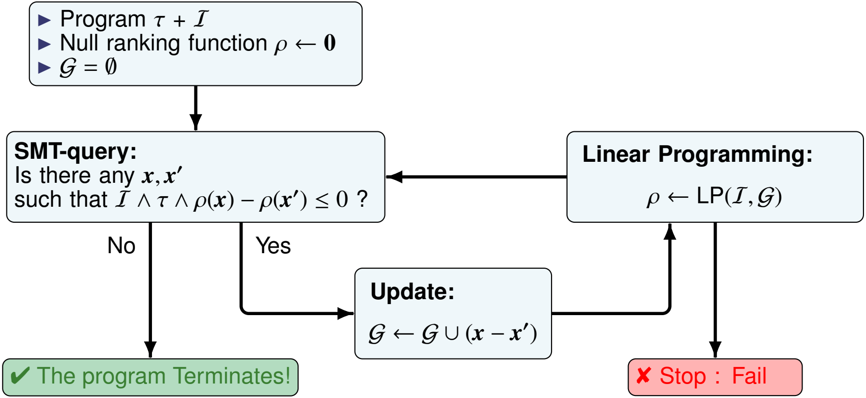
Simple algorithm for one control point



Simple algorithm for one control point



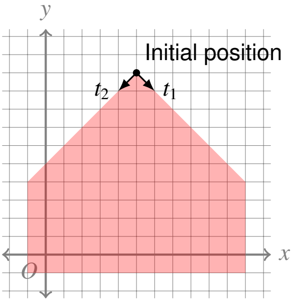
Simple algorithm for one control point



$$t_2 : \frac{0 \le x \wedge 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \le 10 \wedge 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

Current State

$\rho(x, y) = 0 \qquad \mathcal{G} = \{\}$



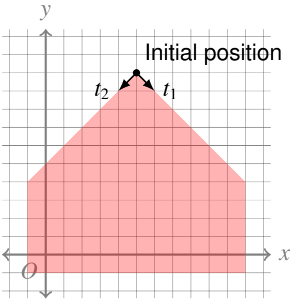
$$t_2 : \frac{0 \leq x \wedge 0 \leq y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \leq 10 \wedge 0 \leq y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

Current State

$\rho(x, y) = 0 \qquad \mathcal{G} = \{\}$

SMT:

$$\mathcal{I} \wedge \tau \wedge \rho(x, y) - \rho(x', y') \leq 0$$



$$t_2 : \frac{0 \le x \wedge 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \le 10 \wedge 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

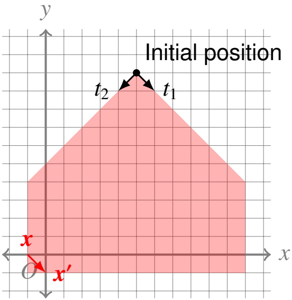
Current State

$\rho(x, y) = 0 \qquad \mathcal{G} = \{\}$

SMT:

$$\mathcal{I} \wedge \tau \wedge \rho(x, y) - \rho(x', y') \le 0$$

$$\begin{array}{ll} x = -1 & x' = 0 \\ y = 0 & y' = -1 \end{array} \qquad \mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$



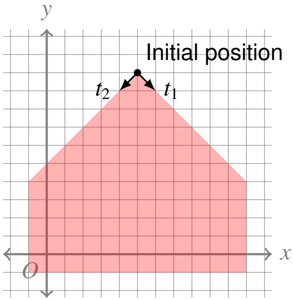
$$t_2 : \frac{0 \leq x \wedge 0 \leq y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \leq 10 \wedge 0 \leq y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

Current State

$\rho(x, y) = 0 \qquad \mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$

LP gives us a potential ranking function:

$$\rho(x, y) = 11 - x$$



$$t_2 : \frac{0 \le x \wedge 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \le 10 \wedge 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

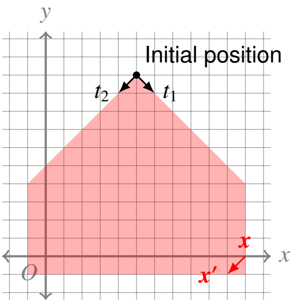
Current State

$$\rho(x, y) = 11 - x \qquad \mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$

SMT:

$$\mathcal{I} \wedge \tau \wedge \rho(x, y) - \rho(x', y') \le 0$$

$$\begin{array}{ll} x = 11 & x' = 10 \\ y = 0 & y' = -1 \end{array} \qquad \mathcal{G} = \mathcal{G} \cup \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$



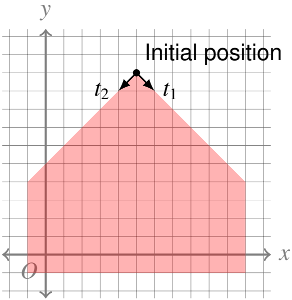
$$t_2 : \frac{0 \le x \wedge 0 \le y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \le 10 \wedge 0 \le y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

Current State

$$\rho(x, y) = 11 - x \qquad \mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$

LP gives us a potential ranking function:

$$\rho(x, y) = y + 1$$



$$t_2 : \frac{0 \leq x \wedge 0 \leq y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}} \qquad k_0 \qquad t_1 : \frac{x \leq 10 \wedge 0 \leq y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

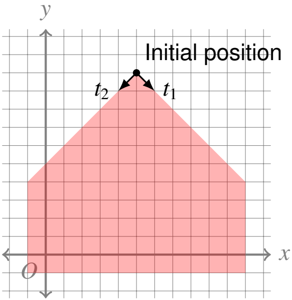
Current State

$$\rho(x, y) = y + 1 \qquad \mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$

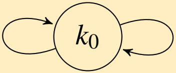
SMT:

$$\mathcal{I} \wedge \tau \wedge \rho(x, y) - \rho(x', y') \leq 0$$

which is **unsat**: There is no counterexample!



$$t_2 : \frac{0 \leq x \land 0 \leq y}{\begin{array}{c} x := x - 1 \\ y := y - 1 \end{array}}$$



$$t_1 : \frac{x \leq 10 \land 0 \leq y}{\begin{array}{c} x := x + 1 \\ y := y - 1 \end{array}}$$

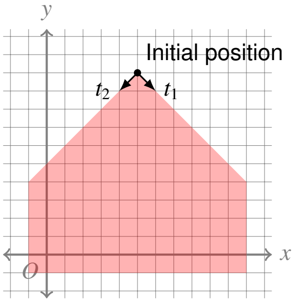
Current State

$$\rho(x, y) = y + 1 \qquad \mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$

Output

$$\rho(x, y) = y + 1$$

ρ is a **strict** ranking function.



A major issue!

This algorithm **doesn't terminate** in general:

- The set of counter examples can be infinite.
- If there is no strict ranking function.

Fix: limit the search area for the counterexample $u = \boldsymbol{x} - \boldsymbol{x}'$

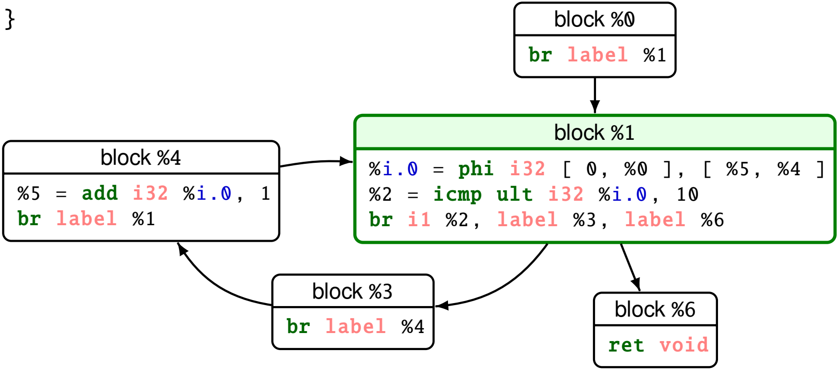
- impose counterexamples to be in the boundary of $\mathcal{D}$ (max-SMT).
- always **improve** the ranking or quit.

Control flow graph and LLVM representation

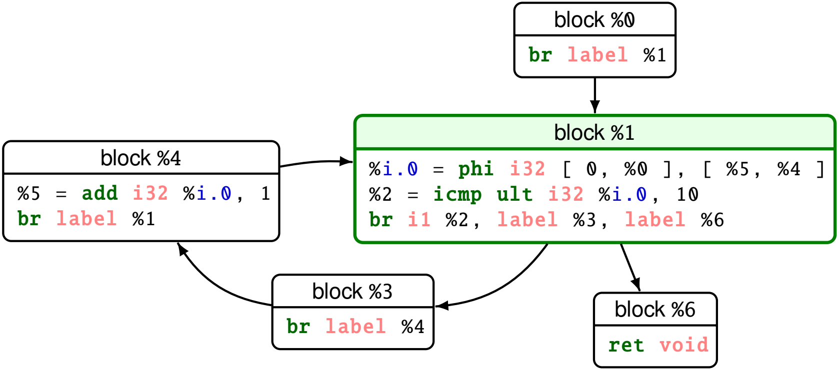
```
void simple_loop_constant() {
  for(unsigned i=0; i<10; i++) {
    // Do nothing
  }
}
```

Control flow graph and LLVM representation

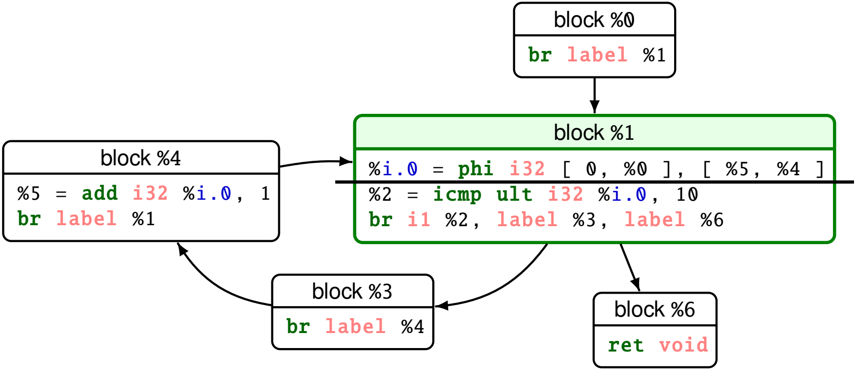
```
void simple_loop_constant() {
  for(unsigned i=0; i<10; i++) {
    // Do nothing
  }
}
```



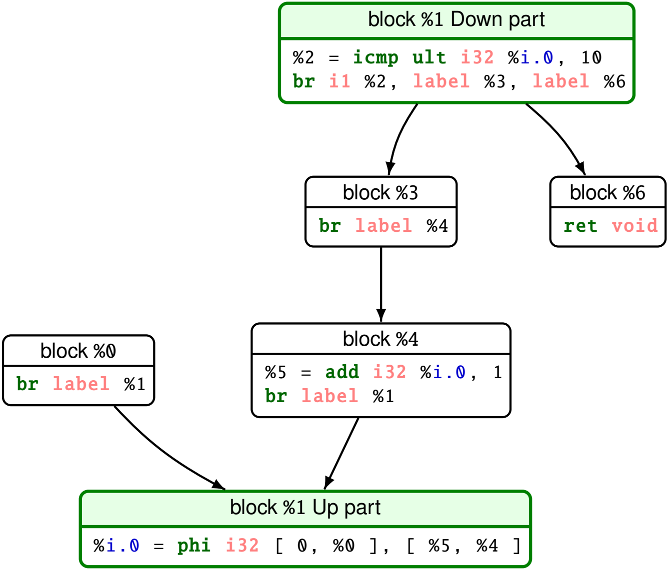
SMT encoding for control-flow-graph



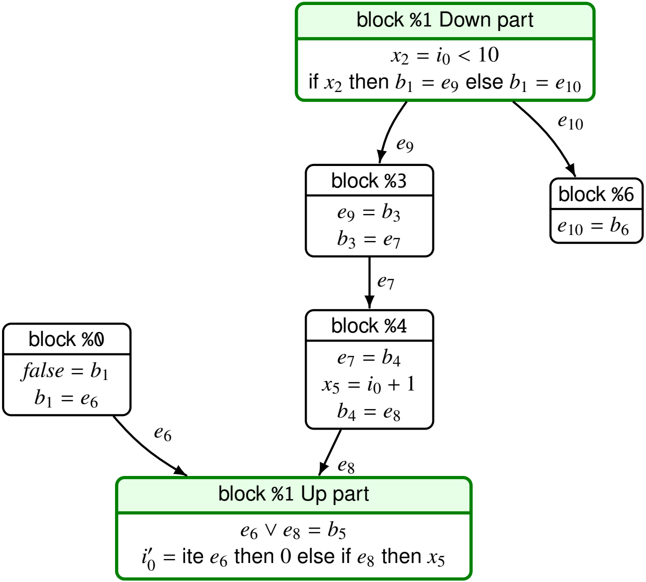
SMT encoding for control-flow-graph



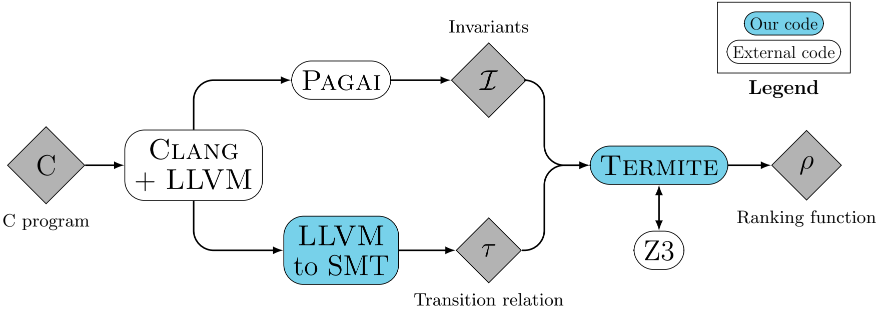
SMT encoding for control-flow-graph



SMT encoding for control-flow-graph



Software architecture



http://termite-analyser.github.io/

Experimental setup

- **Benchmarks**: PolyBench, Some sorts, TermComp, WTC
- **Machine:** Intel(R) Xeon(R) @ 2.00GHz 20MB Cache.
- **Other tools:** (Rank), Aprove, Büchi Ultimate, Loopus.

▶ Issue : various front-ends / invariant generators

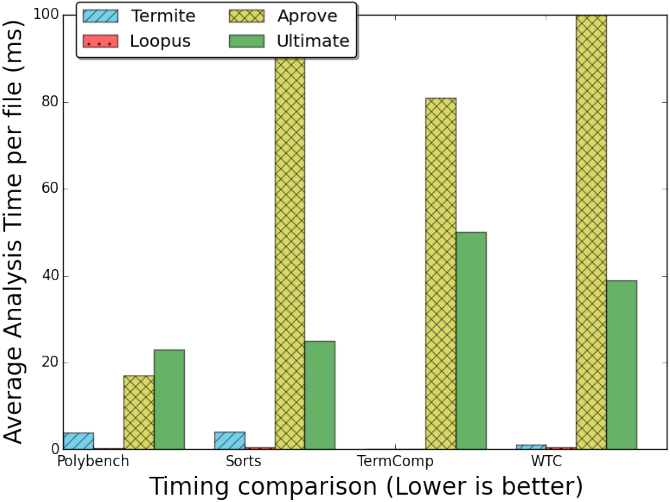
Comparison : Linear Programming instances sizes

On WTC benchmark (average per file):

Tool	#lines (constraints)	#columns (variables)
RANK	584	229
TERMITE	5	2

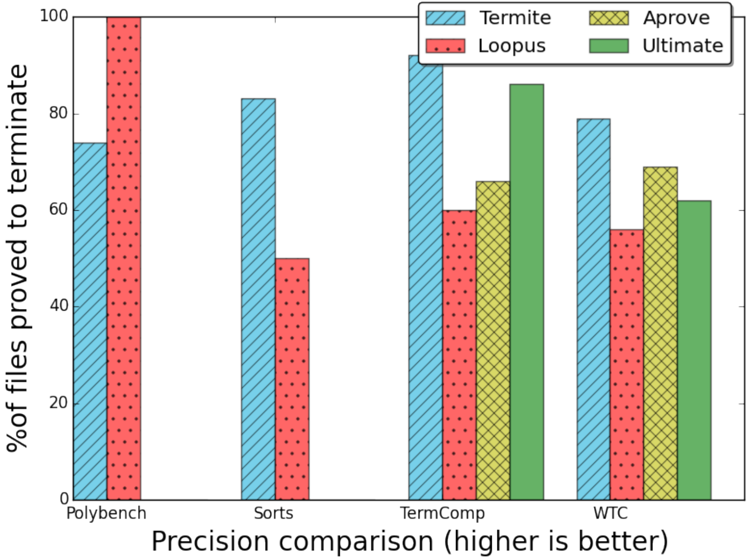
RANK is the termination tool from [Alias et al, SAS 2010]

Timing Comparison



Timings exclude the front-end for Termite and Loopus.

Precision Comparison



In the paper

- The complete method: multidimensional algorithm, multi control points.
- Correctness, Complexity.
- Experimental evaluation.

Summary

- A complete method to synthetise multidimensional ranking functions
- Based on large block encoding + counter-example based linear programming instance generation.
- Experiments show great results!

▶ `http://termite-analyser.github.io/`

Future Work

- Use the technique to also compute $\mathcal{I}$.
- Conditional termination.

Questions ?

Questions ?

Polyhedrons

Closed convex polyhedron

A set $\mathcal{P}$ is a *closed convex polyhedron* iff there exists a set of pairs $(\boldsymbol{a}_i, b_i)$ such that

$$\mathcal{P} = \{\boldsymbol{x} \mid \bigwedge_i \boldsymbol{a}_i.\boldsymbol{x} \geq b_i\}$$

Unbounded convex polyhedron

If unbounded, we have to include rays as generators:

$$\mathcal{P} = \left\{ \left(\sum_i \alpha_i \boldsymbol{v}_i \right) + \left(\sum_i \beta_i \boldsymbol{r}_i \right) \middle| \alpha_i \geq 0, \beta_i \geq 0, \sum_i \alpha_i = 1 \right\}$$

Transition system

Transition system

We consider programs over a state space $\mathcal{S} \subset \mathcal{W} \times \mathbb{Q}^n$, where:

- $\mathcal{W}$ is the finite set of control states, defined by an initial state and a transition relation τ;
- $\mathbb{Q}^n$ is the value of the set of variable considered at the different control points.

Set of reachable values

We note

$$\mathcal{R}_k = \left\{ \boldsymbol{x} \;\middle|\; (k, \boldsymbol{x}) \in \mathcal{S} \right\}$$

the set of all values of $\boldsymbol{x}$ when the flow is in the state k.

Invariants

Invariant

An *invariant* on a control point $k \in \mathcal{W}$ is a formula $\phi_k(\boldsymbol{x})$ that is true for all reachable states $(k, \boldsymbol{x})$.

Affine invariant

An invariant is *affine* if it is a conjunction of a finite number of affine conditions on program variables.
Said in another way, for all $k \in \mathcal{W}$, there exists a convex polyhedron $\mathcal{P}_k$ such that $\mathcal{R}_k \subseteq \mathcal{P}_k$.

Linear ranking function

A *(strict) linear ranking function*
is a function $\rho : \mathcal{W} \times \mathbb{Q}^n \to \mathbb{Q}$ such that:

- for any state $k \in \mathcal{W}$, $\boldsymbol{x} \mapsto \rho(k, \boldsymbol{x})$ is affine linear;
- for any transition $(k, \boldsymbol{x}, k', \boldsymbol{x}')$, $\rho(k', \boldsymbol{x}') \leq \rho(k, \boldsymbol{x}) - 1$;
- for any state $(k, \boldsymbol{x})$ in the invariant $\mathcal{I}$, $\rho(k, \boldsymbol{x}) \geq 0$;

Weak linear ranking function

We replaces the second condition by $\rho(k', \boldsymbol{x}') \leq \rho(k, \boldsymbol{x})$.

Lexicographic *Linear ranking function*

A **Lexicographic** *(strict) linear ranking function* **of dimension** m is a function $\rho : \mathcal{W} \times \mathbb{Q}^n \to \mathbb{Q}^{\mathbf{m}}$ such that:

- for any state $k \in \mathcal{W}$, $\boldsymbol{x} \mapsto \rho(k, \boldsymbol{x})$ is affine linear;
- for any transition $(k, \boldsymbol{x}, k', \boldsymbol{x}')$, $\rho(k', \boldsymbol{x}') \prec \rho(k, \boldsymbol{x})$;
- for any state $(k, \boldsymbol{x})$ in the invariant $\mathcal{I}$,
  all coordinates of $\rho(k, x)$ are nonnegative .

Weak **Lexicographic** *linear ranking function*

We replaces the second condition by $\rho(k', \boldsymbol{x}') \preceq \rho(k, \boldsymbol{x})$.

Lexicographic order

$\langle x_1, \ldots, x_m \rangle \prec \langle y_1, \ldots, y_m \rangle$ if and only if there exists an i such that $x_j = y_j$ for all $j < i$ and $x_i \leq y_i - 1$

Details about the LP problem

$$\rho(\boldsymbol{x}) = \boldsymbol{A} \cdot \boldsymbol{x} + B$$

Details about the LP problem

$$\rho(\boldsymbol{x}) = \left(\sum_{i=1}^{m} \lambda_i \boldsymbol{a}_i\right) \cdot \boldsymbol{x} + \sum_{i=1}^{m} \lambda_i b_i \quad \text{with } \mathcal{I} = \left\{\boldsymbol{x} \;\middle|\; \boldsymbol{a}_i \cdot \boldsymbol{x} + b_i \geq 0\right\}$$

Details about the LP problem

$$\rho(\boldsymbol{x}) = \left(\sum_{i=1}^{m} \lambda_i \boldsymbol{a}_i\right) \cdot \boldsymbol{x} + \sum_{i=1}^{m} \lambda_i b_i \quad \text{with } \mathcal{I} = \left\{\boldsymbol{x} \;\middle|\; \boldsymbol{a}_i \cdot \boldsymbol{x} + b_i \geq 0\right\}$$

Definition: $LP(C, \mathcal{I})$

- $C = (\boldsymbol{u}_1, \ldots \boldsymbol{u}_N)$ a set of generators of the polyhedron $\mathcal{P}_{\mathcal{I},\tau}$,

$$LP(C, \mathcal{I}) = \begin{cases} Maximize \sum_i \delta_i \text{ s.t.} & \\ \lambda_1, \ldots, \lambda_m \geq 0 & \\ 0 \leq \delta_j \leq 1 & \text{for all } 1 \leq j \leq N \\ \sum_{i=1}^{m} \lambda_i(\boldsymbol{u}_j.\boldsymbol{a}_i) \geq \delta_j & \text{for all } 1 \leq j \leq N \end{cases}$$

Proposition

- $\lambda = \mathbf{0}$ is always a solution.
- The ranking function such defined is "maximally strict" on C.

$\mathcal{C} \leftarrow \emptyset$, $\boldsymbol{l} \leftarrow \mathbf{0}$, $\ell \leftarrow 0$

finished $\leftarrow$ *false*

while $not(finished)$ and $(\,\mathcal{I} \wedge \tau \wedge (\boldsymbol{x} - \boldsymbol{x'}).\boldsymbol{l} \leq 0$ satisfiable$)$ **do**

    $(\boldsymbol{x}, \boldsymbol{x'}) \leftarrow$ a model for the above SMT test

    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{x} - \boldsymbol{x'}\}$

    $(\boldsymbol{\lambda}, \boldsymbol{\delta}) \leftarrow LP(\mathcal{C}, \mathrm{Cons}_{\mathcal{I}})$

    if $\lambda = \mathbf{0}$ **then**

        finished $\leftarrow$ *true*

    else

        $\boldsymbol{l} \leftarrow \sum_{i=1}^{m} \lambda_i \boldsymbol{a}_i$

        $\ell \leftarrow \sum_{i=1}^{m} \lambda_i b_i$

    end if

end while

Return $(\boldsymbol{l}, \ell, \bigwedge_i \delta_i = 1)$.