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

ONE DOES NOT SIMPLY

PROVE PROGRAM TERMINATION
Goal: Safety

Prove that (some) loops terminate:

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

- Fight against bugs.
Goal: Optimisation

Prove that (some) loops terminate:

```c
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.
$t_2: \frac{0 \leq x \land 0 \leq y}{x := x - 1 \land y := y - 1}$

$k_0$

$t_1: \frac{x \leq 10 \land 0 \leq y}{x := x + 1 \land y := y - 1}$
The Initial position:

\[ x = 5 \text{ and } y = 10 \]
The Initial position:

\[ x = 5 \text{ and } y = 10 \]

The invariant \( I \).
A linear ranking function:

\[ \rho(x, y) = y + 1 \]

- Linear
- Decreasing along the transitions
- Positive on \( \mathcal{I} \)
A linear ranking function:

\[ \rho(x, y) = y + 1 \]

- Linear
- Decreasing along the transitions
- Positive on \( \mathcal{I} \)
- **Strict**: decreasing by \( \geq 1 \).
We want to find a linear ranking function.

- Linear
- Decreasing along the transitions
- Positive on $I$

In programs with one control point, for now.

When transitions are \textit{linear}.

A \textbf{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:

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

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

$\mathcal{G}$: generators of $\mathcal{D}$

$\mathcal{I}$: invariants

Max termination power on $\mathcal{G}$

$\rho$ positive, decreasing on $\mathcal{G}$, and \textbf{strictly decreasing on a maximal subset of $\mathcal{G}$}
Existing techniques: build a system of constraints and solve:

\[ \text{Size} = O(\#\text{vars} \times \#\text{Bblocks} \times \#\text{transitions}) \]

- scalability: all basic blocks \(\leadsto\) big constraint systems
- precision: \(\rho\) 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

- Program
- Initial Guess

Try to find a contradiction.
- Is there a counterexample?

- No
- Yes

Refine with counterexample.

✔️ The program Terminates!
Solving the problem

\( \mathcal{G} \): generators of \( \mathcal{D} \)

\( \mathcal{I} \): invariants

\( \text{Max termination power on } \mathcal{G} \)

\( \mathbf{LP} \)

\( \rho \)

- We construct \( \mathcal{G} \) lazily.
Simple algorithm for one control point

Program
Null ranking function

Is there a path that contradicts the fact $\rho$ is a strict ranking function?

Yes

Compute a new ranking function that “satisfies” all elements of $G$.

No
Add the counterexample in $G$

✔ The program Terminates!

✗ Stop : Fail

Program
Null ranking function

Is there a path that contradicts the fact $\rho$ is a strict ranking function?

No
Add the counterexample in $G$

✔ The program Terminates!

✗ Stop : Fail

Program
Null ranking function

Is there a path that contradicts the fact $\rho$ is a strict ranking function?

Yes
Compute a new ranking function that “satisfies” all elements of $G$.

No
Add the counterexample in $G$

✔ The program Terminates!

✗ Stop : Fail

Program
Null ranking function

Is there a path that contradicts the fact $\rho$ is a strict ranking function?

Yes
Compute a new ranking function that “satisfies” all elements of $G$.

No
Add the counterexample in $G$

✔ The program Terminates!

✗ Stop : Fail

Program
Null ranking function

Is there a path that contradicts the fact $\rho$ is a strict ranking function?

Yes
Compute a new ranking function that “satisfies” all elements of $G$.

No
Add the counterexample in $G$

✔ The program Terminates!

✗ Stop : Fail
Simple algorithm for one control point

- Program $\tau + I$
- Null ranking function $\rho \leftarrow 0$
- $G = \emptyset$

Is there a path that contradicts the fact $\rho$ is a strict ranking function?

- No
  - ✔️ The program Terminates!
- Yes
  - Compute a new ranking function that "satisfies" all elements of $G$.

Add the counterexample in $G$

- ✗ Stop : Fail
Simple algorithm for one control point

- Program $\tau + \mathcal{I}$
- Null ranking function $\rho \leftarrow 0$
- $\mathcal{G} = \emptyset$

**SMT-query:**
Is there any $x, x'$ such that $\mathcal{I} \land \tau \land \rho(x) - \rho(x') \leq 0$?

- No
  - Add the counterexample in $\mathcal{G}$
  - ✓ The program Terminates!

- Yes
  - Compute a new ranking function that “satisfies” all elements of $\mathcal{G}$
  - × Stop : Fail
Simple algorithm for one control point

Program $\tau + I$
Null ranking function $\rho \leftarrow 0$
$G = \emptyset$

SMT-query: Is there any $x, x'$ such that $I \land \tau \land \rho(x) - \rho(x') \leq 0$?

No

✔ The program Terminates!

Yes

Update: $G \leftarrow G \cup (x - x')$

Compute a new ranking function that "satisfies" all elements of $G$.

❌ Stop: Fail

Algorithm to synthesize a ranking function
Simple algorithm for one control point

Program $\tau + I$
Null ranking function $\rho \leftarrow 0$
$G = \emptyset$

SMT-query:
Is there any $x, x'$ such that $I \land \tau \land \rho(x) - \rho(x') \leq 0$?

No
Yes

Update:
$G \leftarrow G \cup (x - x')$

✅ The program Terminates!

Linear Programming:
$\rho \leftarrow \text{LP}(I, G)$

❌ Stop : Fail
**Current State**

\[ \rho(x, y) = 0 \quad \mathcal{G} = \{\} \]
Current State

\[ \rho(x, y) = 0 \quad \mathcal{G} = \{ \} \]

SMT:

\[ I \land \tau \land \rho(x, y) - \rho(x', y') \leq 0 \]
**Motivation and big picture**

Algorithm to synthesize a ranking function

Experimental results

Conclusion

---

**Current State**

\[ \rho(x, y) = 0 \quad G = \{\} \]

**SMT:**

\[
\begin{align*}
\mathcal{I} \land \tau \land \rho(x, y) - \rho(x', y') &\leq 0 \\
x &= -1 \quad x' = 0 \\
y &= 0 \quad y' = -1 \\
G &= \begin{pmatrix} -1 \\ 1 \end{pmatrix}
\end{align*}
\]
**Current State**

\( \rho(x, y) = 0 \)

\[ G = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\} \]

**LP** gives us a potential ranking function:

\[ \rho(x, y) = 11 - x \]
**Current State**

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

**SMT:**

\[ \mathcal{I} \land \tau \land \rho(x, y) - \rho(x', y') \leq 0 \]

\[
\begin{align*}
  x &= 11 & x' &= 10 \\
  y &= 0 & y' &= -1
\end{align*}
\]

\[ \mathcal{G} = \mathcal{G} \cup \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\} \]
Motivation and big picture

Algorithm to synthesize a ranking function

Experimental results

Conclusion

Current State

\[ \rho(x, y) = 11 - x \quad \mathcal{G} = \left\{ \left( \begin{array}{c} \frac{-1}{1} \\ \frac{1}{1} \end{array} \right) \right\} \]

**LP** gives us a potential ranking function:

\[ \rho(x, y) = y + 1 \]
**Current State**

\[ \rho(x, y) = y + 1 \quad \mathcal{G} = \left\{ \left( \begin{array}{c} -1 \\ 1 \end{array} \right), \left( \begin{array}{c} 1 \\ 1 \end{array} \right) \right\} \]

**SMT:**

\[ \mathcal{I} \land \tau \land \rho(x, y) - \rho(x', y') \leq 0 \]

which is **unsat**: There is no counterexample!
Motivation and big picture

Algorithm to synthesize a ranking function

Experimental results

Conclusion

\[ t_2 : \begin{array}{l}
0 \leq x \land 0 \leq y \\
x := x - 1 \\
y := y - 1
\end{array} \]

\[ t_1 : \begin{array}{l}
x \leq 10 \land 0 \leq y \\
x := x + 1 \\
y := y - 1
\end{array} \]

Current State

\[ \rho(x, y) = y + 1 \quad \mathcal{G} = \left\{ \left(\begin{array}{c}-1 \\ 1\end{array}\right), \left(\begin{array}{c}1 \\ 1\end{array}\right) \right\} \]

Output

\[ \rho(x, y) = y + 1 \]

\( \rho \) 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 $u = x - x'$
- impose counterexamples to be in the boundary of $\mathcal{D}$ (max-SMT).
- always *improve* the ranking or quit.
Control flow graph and LLVM representation

```c
void simple_loop_constant() {
    for(unsigned i=0; i<10; i++) {
        // Do nothing
    }
}
```
Control flow graph and LLVM representation

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

![Control flow graph diagram]

- `block %0
  br label %1`
- `block %1
  %i.0 = phi i32 [ 0, %0 ], [ %5, %4 ]
  %2 = icmp ult i32 %i.0, 10
  br i1 %2, label %3, label %6`
- `block %3
  br label %4`
- `block %4
  %5 = add i32 %i.0, 1
  br label %1`
- `block %6
  ret void`
SMT encoding for control-flow-graph

block %0
br label %1

block %1
%i.0 = phi i32 [ %0, %0 ], [ %5, %4 ]
%2 = icmp ult i32 %i.0, 10
br i1 %2, label %3, label %6

block %3
br label %4

block %4
%5 = add i32 %i.0, 1
br label %1

block %6
ret void
SMT encoding for control-flow-graph

```
%0 = phi i32 [ 0, %0 ], [ %5, %4 ]
%2 = icmp ult i32 %i.0, 10
br i1 %2, label %3, label %6
%5 = add i32 %i.0, 1
br label %1
%3 = add i32 %i.0, 1
br label %1
%6 = add i32 %i.0, 1
ret void
```
SMT encoding for control-flow-graph

<table>
<thead>
<tr>
<th>Block %1 Down part</th>
</tr>
</thead>
<tbody>
<tr>
<td>%2 = icmp ult i32 %i.0, 10</td>
</tr>
<tr>
<td>br i1 %2, label %3, label %6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block %3</th>
</tr>
</thead>
<tbody>
<tr>
<td>br label %4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block %6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ret void</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block %0</th>
</tr>
</thead>
<tbody>
<tr>
<td>br label %1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block %4</th>
</tr>
</thead>
<tbody>
<tr>
<td>%5 = add i32 %i.0, 1</td>
</tr>
<tr>
<td>br label %1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block %1 Up part</th>
</tr>
</thead>
<tbody>
<tr>
<td>%i.0 = phi i32 [ 0, %0 ], [ %5, %4 ]</td>
</tr>
</tbody>
</table>
SMT encoding for control-flow-graph

Motivation and big picture

Algorithm to synthesize a ranking function

Experimental results

Conclusion

block %1 Down part

\[ x_2 = i_0 < 10 \]

if \( x_2 \) then \( b_1 = e_9 \) else \( b_1 = e_{10} \)

block %3

\[ e_9 = b_3 \]

\[ b_3 = e_7 \]

\[ e_7 \]

block %6

\[ e_{10} = b_6 \]

block %0

false = \( b_1 \)

\[ b_1 = e_6 \]

block %1 Up part

\[ e_6 \lor e_8 = b_5 \]

\[ i'_0 = \text{ite} \ e_6 \text{ then } 0 \text{ else if } e_8 \text{ then } x_5 \]

\[ e_6 \]

\[ e_8 \]
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):

<table>
<thead>
<tr>
<th>Tool</th>
<th>#lines (constraints)</th>
<th>#columns (variables)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>584</td>
<td>229</td>
</tr>
<tr>
<td>Termite</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

*Rank* is the termination tool from [Alias et al, SAS 2010]
Timing Comparison

Timings exclude the front-end for Termite and Loopus.
Precision Comparison

Precision comparison (higher is better)
In the paper

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

- A complete method to synthesise 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 $I$.
- Conditional termination.
Questions ?
Questions ?
Polyhedrons

Closed convex polyhedron

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

$$\mathcal{P} = \{x \mid \bigwedge_i a_i \cdot x \geq b_i\}$$

Unbounded convex polyhedron

If unbounded, we have to include rays as generators:

$$\mathcal{P} = \left\{ \left( \sum_i \alpha_i v_i \right) + \left( \sum_i \beta_i r_i \right) \mid \alpha_i \geq 0, \beta_i \geq 0, \sum_i \alpha_i = 1 \right\}$$
Transition system

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

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

Set of reachable values

We note

$$R_k = \{x \mid (k, x) \in S\}$$

the set of all values of $x$ when the flow is in the state $k$. 
Invariants

**Invariant**

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

**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 \( k \in W \), there exists a convex polyhedron \( P_k \) such that \( R_k \subseteq P_k \).
### 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 $k \in \mathcal{W}$, $x \mapsto \rho(k, x)$ is affine linear;
- for any transition $(k, x, k', x')$, $\rho(k', x') \leq \rho(k, x) - 1$;
- for any state $(k, x)$ in the invariant $I$, $\rho(k, x) \geq 0$;

### Weak linear ranking function

We replace the second condition by $\rho(k', x') \leq \rho(k, x)$.
**Lexicographic Linear ranking function**

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

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

**Weak Lexicographic linear ranking function**

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

**Lexicographic order**

$\langle x_1, \ldots, x_m \rangle < \langle y_1, \ldots, y_m \rangle$ 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

\[ \rho(x) = A \cdot x + B \]
Details about the LP problem

\[ \rho(x) = \left( \sum_{i=1}^{m} \lambda_i a_i \right) \cdot x + \sum_{i=1}^{m} \lambda_i b_i \quad \text{with} \quad \mathcal{I} = \left\{ x \mid a_i \cdot x + b_i \geq 0 \right\} \]
Details about the LP problem

\[ \rho(x) = \left( \sum_{i=1}^{m} \lambda_i a_i \right) \cdot x + \sum_{i=1}^{m} \lambda_i b_i \quad \text{with } \mathcal{I} = \left\{ x \mid a_i \cdot x + b_i \geq 0 \right\} \]

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

- \( C = (u_1, \ldots, u_N) \) a set of generators of the polyhedron \( P_{\mathcal{I},\tau} \),

\[
LP(C, \mathcal{I}) = \begin{cases} 
\text{Maximize } \sum_i \delta_i \quad \text{s.t.} \\
\lambda_1, \ldots, \lambda_m \geq 0 \\
0 \leq \delta_j \leq 1 \quad \text{for all } 1 \leq j \leq N \\
\sum_{i=1}^{m} \lambda_i (u_j \cdot a_i) \geq \delta_j \quad \text{for all } 1 \leq j \leq N 
\end{cases}
\]

Proposition

- \( \lambda = 0 \) is always a solution.
- The ranking function such defined is “maximally strict” on \( C \).
1: $C \leftarrow \emptyset$, $I \leftarrow 0$, $\ell \leftarrow 0$
2: $finished \leftarrow false$
3: while not($finished$) and ($I \land \tau \land (x - x').I \leq 0$ satisfiable) do
4:     $(x, x') \leftarrow$ a model for the above SMT test
5:     $C \leftarrow C \cup \{x - x'\}$
6:     $(\lambda, \delta) \leftarrow LP(C, Cons_I)$
7:     if $\lambda = 0$ then
8:         $finished \leftarrow true$
9:     else
10:        $l \leftarrow \sum_{i=1}^{m} \lambda_i a_i$
11:        $\ell \leftarrow \sum_{i=1}^{m} \lambda_i b_i$
12:     end if
13: end while
14: Return $(l, \ell, \land_i \delta_i = 1)$. 