Automatic Inference of Ranking Functions by Abstract Interpretation

Caterina Urban

Département d'Informatique
École Normale Supérieure

Project-Team ANTIQUE
INRIA Paris-Rocquencourt

Dagstuhl Seminar 14352
“Next Generation Static Software Analysis Tools”
Schloss Dagstuhl, Wadern, Germany
Proving Program Termination? Why?

30GB Zunes all over the world fail en masse

posted Dec 31, 2008 by Matt Burns

It seems that a random bug is affecting a bunch, if not every, 30GB Zunes. Real early this morning, a bunch of Zune 30s just stopped working. No official word from Redmond on this one yet but we might have a gadget Y2K going on here. Fan boards and support forums all have the same mantra saying that at 2:00 AM this morning, the Zune 30s reset on their own and doesn’t fully reboot. We’re sure Microsoft will get flooded with angry Zune owners as soon as the phone lines open up for the last time in 2008. More as we get it.

Update 2: The solution is ... kind of weak: let your Zune run out of battery and it'll be fixed when you wake up tomorrow and charge it.
Proving Program Termination? Why?

Zune bug explained in detail

Posted Dec 31, 2008 by Devin Coldewey

Earlier today, the sound of thousands of Zune owners crying out in terror made ripples across the blogosphere. The response from Microsoft is to wait until tomorrow and all will be well. You're probably wondering, what kind of bug fixes itself?

Well, I've got the code here and it's very simple, really; if you've taken an introductory programming class, you'll see the error right away.

```c
while (days > 365)
{
   if (IsLeapYear(year))
   {
      if (days > 366)
      {
         days -= 366;
         year += 1;
      }
   } else
   {
      days -= 355;
      year += 1;
   }
}
```

You can see the details here, but the important bit is that today, the day count is 366. As you can
Outline

- **ranking functions**
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below

- idea: computation of ranking functions by abstract interpretation

- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
  - instances based on ordinal-valued ranking functions
  - instances based on decision trees

---

1. Floyd - *Assigning Meanings to Programs* (1967)
5. Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)
Outline

- **ranking functions**\(^1\)
  - functions that strictly **decrease** at each program step...
  - ... and that are **bounded** from below

- **idea**: computation of ranking functions by abstract interpretation\(^2\)

- family of **abstract domains** for program termination\(^3\)
  - piecewise-defined ranking functions
  - **backward invariance analysis**
  - **sufficient conditions** for termination

- instances based on **ordinal-valued ranking functions**\(^4\)
- instances based on **decision trees**\(^5\)

---

\(^1\) Floyd - *Assigning Meanings to Programs* (1967)

\(^2\) Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

\(^3\) Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)

\(^4\) Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)

\(^5\) Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)
Outline

- **ranking functions**: functions that strictly decrease at each program step... and that are bounded from below.

- **idea**: computation of ranking functions by abstract interpretation

- **family of abstract domains** for program termination
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination

- instances based on **ordinal-valued ranking functions**
- instances based on **decision trees**

---

1. Floyd - *Assigning Meanings to Programs* (1967)
5. Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)
Abstract Interpretation\(^6\)

\[ \langle C, \sqsubseteq_C \rangle \]

---

\(^6\)Cousot&Cousot - Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. (POPL 1977)
Abstract Interpretation\(^6\)

\[ \langle C, \sqsubseteq_C \rangle \quad \gamma \quad \langle A, \sqsubseteq_A \rangle \]

\[ S \quad \gamma \quad [P]^\alpha \]

\[ S^\alpha \]

---

\(^6\)Cousot&Cousot - Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. (POPL 1977)
Abstract Interpretation\(^6\)

\[ \langle C, \sqsubseteq_C \rangle \]

\[ \gamma(\llbracket P \rrbracket^\alpha) \]

\[ \langle A, \sqsubseteq_A \rangle \]

\[ \llbracket P \rrbracket^\alpha \]

---

\(^6\)Cousot & Cousot - Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. (POPL 1977)
\[ \langle C, \sqsubseteq_C \rangle \quad \gamma([P]^\alpha) \quad \gamma \quad \langle A, \sqsubseteq_A \rangle \]

\[
\begin{align*}
S & \quad \checkmark \quad [P] \\
S^\alpha & \quad \checkmark \quad [P]^\alpha
\end{align*}
\]
\[ \langle C, \sqsubseteq_C \rangle \]

\[ \langle A, \sqsubseteq_A \rangle \]

\[ S \]

\[ \gamma([P]^\alpha) \]

\[ [P] \]

\[ S^\alpha \]

\[ [P]^\alpha \]

\[ S \]

\[ \gamma([P]^\alpha) \]

\[ [P] \]

\[ S^\alpha \]

\[ [P]^\alpha \]
Termination Semantics
program $\mapsto$ trace semantics

finite traces $\Sigma^+$

\[\begin{array}{c}
\text{β final states}
\end{array}\]

infinite traces $\Sigma^\infty$

\[\begin{array}{c}
\Sigma \text{ states}
\end{array}\]

transition relation $\tau$
program $\mapsto$ trace semantics $\mapsto$ termination semantics

**Example**

idea = define a ranking function counting the number of program steps from the end of the program and extracting the well-founded part of the program transition relation

**Theorem (Soundness and Completeness)**

the termination semantics is **sound** and **complete** to prove the termination of programs

---

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
program $\mapsto$ trace semantics $\mapsto$ termination semantics

**Example**

idea = define a ranking function counting the number of program steps from the end of the program and extracting the well-founded part of the program transition relation

**Theorem (Soundness and Completeness)**

the termination semantics is sound and complete to prove the termination of programs

Cousot&Cousot - An Abstract Interpretation Framework for Termination (POPL 2012)
program $\mapsto$ trace semantics $\mapsto$ termination semantics

**Example**

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

**Theorem (Soundness and Completeness)**

the termination semantics is **sound and complete** to prove the termination of programs

---

Cousot& Cousot - An Abstract Interpretation Framework for Termination (POPL 2012)
The idea is to define a ranking function counting the number of program steps from the end of the program and extracting the well-founded part of the program transition relation.

Theorem (Soundness and Completeness)

the termination semantics is sound and complete to prove the termination of programs

Cousot & Cousot - An Abstract Interpretation Framework for Termination (POPL 2012)
**Termination Semantics**

**Piecewise-Defined Ranking Functions**

**Conclusion**

---

**Example**

- **idea**: define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

- **Theorem (Soundness and Completeness)**
  
  *the termination semantics is **sound** and **complete** to prove the termination of programs*

---

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
**Example**

**program** \(\mapsto\) **trace semantics** \(\mapsto\) **termination semantics**

**Idea** = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation.

**Example**

Theorem (Soundness and Completeness)

the **termination semantics** is **sound** and **complete** to prove the termination of programs

---

Cousot & Cousot - An Abstract Interpretation Framework for Termination (POPL 2012)
remark: the termination semantics is not computable!

Example

```latex
\begin{verbatim}
int : x
x := ?
while (x > 0) do
  x := x - 1
od
\end{verbatim}
```
Piecewise-Defined Ranking Functions
Termination Semantics

Abstract Termination Semantics

- States Abstract Domain \( S \)
- Functions Abstract Domain \( F \)
- Piecewise-Defined Ranking Functions Abstract Domain \( V(S, F) \)

Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
Termination Semantics
\[⟨Σ → ⊘, ⊑⟩\]

Abstract Termination Semantics
\[⟨V, ⊑^V⟩\]

- States Abstract Domain
- Functions Abstract Domain
- Piecewise-Defined Ranking Functions Abstract Domain

Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
Termination Semantics
\(\langle \Sigma \rightarrow \emptyset, \sqsubseteq \rangle\)

Abstract Termination Semantics
\(\langle V, \sqsubseteq V \rangle\)

- States Abstract Domain
- Functions Abstract Domain
- Piecewise-Defined Ranking Functions Abstract Domain

Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
Termination Semantics
\[ \langle \Sigma \rightarrow \varnothing, \sqsubseteq \rangle \]

Abstract Termination Semantics
\[ \langle V, \sqsubseteq V \rangle \]

- States Abstract Domain
- Functions Abstract Domain
- Piecewise-Defined Ranking Functions Abstract Domain

Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
Affine Ranking Functions Abstract Domain

- States Abstract Domain
  - \( \mathcal{I} \triangleq \text{Intervals Abstract Domain} \)
- Functions Abstract Domain
  - \( \mathcal{A} \triangleq \{ \bot_A \} \cup \{ f \mid f \in \mathbb{Z}^n \rightarrow \mathbb{N} \} \cup \{ \top_A \} \)
  - where \( f \equiv f(x_1, \ldots, x_n) = m_1x_1 + \cdots + m_nx_n + q \)

Urban - The Abstract Domain of Segmented Ranking Functions (SAS 2013)
**Example**

```plaintext
int : x
while \( x \leq 10 \) do
  if \( x > 4 \) then
    \( x := x + 2 \)
  fi
od
```

![Decision Tree Diagram](image)
we map each point to a function of $x$ giving an upper bound on the steps before termination.

Example

```
int : $x$
while $1(x \leq 10)$ do
    if $2(x > 4)$ then
        $3x := x + 2$
    fi
od
```

```
1
\downarrow
x \leq 4
\downarrow
x \leq 10
\downarrow
x > 4
\downarrow
x := x + 2
3
4
x > 10
```
we start at the end with 0 steps before termination

Example

```
int : x
while 1(x ≤ 10) do
  if 2(x > 4) then
    3x := x + 2
  fi
od
```
we take into account $x > 10$ and we have now 1 step to termination

Example

```plaintext
int : x
while $1(x \leq 10)$ do
  if $2(x > 4)$ then
    $3x := x + 2$
  fi
od
```
Example

\[
\text{int : } x \\
\text{while } 1(x \leq 10) \text{ do} \\
\quad \text{if } 2(x > 4) \text{ then} \\
\quad \quad 3x := x + 2 \\
\quad \text{fi} \\
\text{od}^4
\]

we consider the assignment \( x := x + 2 \) or the test \( x \leq 4 \) and we are now at 2 steps to termination
Example

int : x
while \(1(x \leq 10)\) do
  if \(2(x > 4)\) then
    \(3x := x + 2\)
  fi
od

we consider \(x > 4\) and we do the join
we consider $x \leq 10$ and we do the join

**Example**

```plaintext
int : x
while $1(x \leq 10)$ do
  if $2(x > 4)$ then
    $3x := x + 2$
  fi
od
```

---

**Graphical Representation**

- **Node 1**: $x \leq 4$
- **Node 2**: $x \leq 10$
- **Node 3**: $x > 4$
- **Node 4**: $x := x + 2$

**Edges**:
- From Node 1 to Node 2: $x > 10$
- From Node 2 to Node 3: $x \leq 10$
- From Node 3 to Node 4: $x := x + 2$
- From Node 4 to Node 1: $x > 4$

---

**Explanation**

- **Node 1**: $x \leq 4$
- **Node 2**: $x \leq 10$
- **Node 3**: $x > 4$
- **Node 4**: $x := x + 2$

---

**Termination Analysis**

- We start with $0$ steps before termination.
- We consider $x > 10$ and now have $1$ step to termination.
- We consider $x \leq 10$ and we do the join.
- We consider $x > 4$ and we do the join.
- We consider $x \leq 4$ and we do the join.
- We consider $x \leq 10$ and we do the join.
- We map each point to a function of $x$ giving an upper bound on the steps before termination.
- We take into account $x > 10$ and we have now $1$ step to termination.
- We consider the assignment $x := x + 2$ or the test $x \leq 4$ and we are now at $2$ steps to termination.
- We consider $x > 4$ and we do the join.
- We consider $x \leq 10$ and we do the join.
- We do the widening.
- The analysis provides $x > 4$ as sufficient precondition for termination.
Example

\[
\text{int : } x \\
\text{while } 1(x \leq 10) \text{ do} \\
\quad \text{if } 2(x > 4) \text{ then} \\
\quad \quad 3x := x + 2 \\
\quad \quad fi \\
\quad od 4
\]

\[
\begin{array}{cccccc}
4 & 6 & 8 & 10 & x \\
\hline
\end{array}
\]

\[
\begin{array}{cccc}
1 & x \leq 4 & 4 & x > 10 \\
\hline
2 & x \leq 10 & 3 & x > 4 \\
\hline
3 & x := x + 2
\end{array}
\]
Example

int : x
while \(1(x \leq 10)\) do
  if \(2(x > 4)\) then
    \(3x := x + 2\)
  fi
od

we do the widening
Example

\begin{verbatim}
int : x
while 1(x ≤ 10) do
  if 2(x > 4) then
    3x := x + 2
  fi
od
\end{verbatim}
the analysis provides $x > 4$ as **sufficient precondition** for termination

**Example**

```latex
\begin{align*}
\text{int : } & x \\
\text{while } & 1(x \leq 10) \text{ do} \\
\text{if } & 2(x > 4) \text{ then} \\
\text{ } & 3x := x + 2 \\
\text{fi} \\
\text{od}
\end{align*}
```

$0$ $4$ $8$ $10$ $x$ $x > 10$

$1$ $4$

$x \leq 4$

$2$

$x \leq 10$

$3$

$x > 4$

$4$

$x := x + 2$

$2$ $6$ $8$ $x$
Theorem (Soundness)

the abstract termination semantics is sound
to prove the termination of programs
**remark**: natural-valued ranking functions are **not sufficient**!

**Example**

```plaintext
int : x
x := ?
while (x > 0) do
  x := x − 1
od
```
Termination Semantics
\( \langle \Sigma \rightarrow \emptyset, \subseteq \rangle \)

Abstract Termination Semantics
\( \langle V, \subseteq V \rangle \)

- States Abstract Domain
- Functions Abstract Domain
- Piecewise-Defined Ranking Functions Abstract Domain

\( V(S, F) \)

Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
States Abstract Domain $S$

Natural-Valued Functions Abstract Domain $F$

Ordinal-Valued Functions Abstract Domain $O(F)$

Piecewise-Defined Ranking Functions Abstract Domain $V(S, O(F))$
States Abstract Domain
- $\mathcal{I} \triangleq$ Intervals Abstract Domain

Natural-Valued Functions Abstract Domain
- $\mathcal{A} \triangleq$ Affine Ranking Functions Abstract Domain

Ordinal-Valued Functions Abstract Domain
- $\mathcal{O} \triangleq \{\bot_\mathcal{O}\} \cup \{\sum_{i} \omega^i \cdot f_i \mid f_i \in \mathcal{A} \setminus \{\bot_\mathcal{A}, \top_\mathcal{A}\}\} \cup \{\top_\mathcal{O}\}$
non-deterministic assignments are carried out in ascending powers of $\omega$

**Example**

\[
\begin{align*}
[-\infty, +\infty] & \mapsto o \triangleq \omega \cdot x_1 + x_2 \\
& \downarrow x_1 := ?
\end{align*}
\]

\[
\begin{align*}
[-\infty, +\infty] & \mapsto o \triangleq ?
\end{align*}
\]
non-deterministic assignments are carried out in ascending powers of $\omega$

**Example**

\[
\begin{align*}
\mathbb{R} \ni o & \triangleq \omega \cdot x_1 + x_2 \\
\downarrow & \quad \quad \quad \quad \quad x_1 := ? \\
\mathbb{R} \ni o & \triangleq \quad + \quad 1
\end{align*}
\]
Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of $\omega$

Example

\[
\begin{align*}
\textstyle\left[ -\infty, +\infty \right] & \mapsto \quad o \quad \triangleq \\
& \quad \quad \omega \cdot x_1 + x_2 \\
& \quad \downarrow \quad x_1 : = \ ?
\end{align*}
\]

\[
\begin{align*}
\textstyle\left[ -\infty, +\infty \right] & \mapsto \quad o \quad \triangleq \\
& \quad + x_2 + 1
\end{align*}
\]
non-deterministic assignments are carried out in ascending powers of $\omega$

**Example**

\[
[\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_2
\]

\[
\downarrow \quad x_1 := ?
\]

\[
[\infty, +\infty] \mapsto o \triangleq \omega^2 \cdot 1 + \omega \cdot 0 + x_2 + 1
\]

\[
\omega \cdot \omega = \omega^2 \cdot 1 + \omega \cdot 0
\]
Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of $\omega$

Example

$$\left[ -\infty, +\infty \right] \mapsto o \triangleq \omega \cdot x_1 + x_2$$

\[ \downarrow \]

$$x_1 := ?$$

$$\left[ -\infty, +\infty \right] \mapsto o \triangleq \omega^2 + x_2 + 1$$
Join

- join: $\sqcup_A$

Example

- join: $\sqcup_O$
Join

- $\join: \sqcup_A$
- $\join: \sqcup_O$
  - $\sqcup_A$ in ascending powers of $\omega$

Example

\[
\begin{align*}
[\!-\infty, +\infty] & \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 \\
[\!-\infty, +\infty] & \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 \\
[\!-\infty, +\infty] & \mapsto o_1 \sqcup_0 o_2 \triangleq ?
\end{align*}
\]
Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of $\omega$

### Example

<table>
<thead>
<tr>
<th>Interval</th>
<th>$o_1$</th>
<th>$o_2$</th>
<th>$\sqcup_A o_2$</th>
<th>$\omega^2 \cdot x_1 + \omega \cdot x_2 + 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[-\infty, +\infty]$</td>
<td>$\Rightarrow$</td>
<td>$\Rightarrow$</td>
<td>$\sqcup_A$</td>
<td>$\omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$</td>
</tr>
<tr>
<td></td>
<td>$\omega^2 \cdot x_1 + \omega \cdot x_2 + 3$</td>
<td>$\Rightarrow$</td>
<td>$\sqcup_A o_2$</td>
<td>$\omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$</td>
</tr>
<tr>
<td></td>
<td>$\Rightarrow$</td>
<td>$\Rightarrow$</td>
<td></td>
<td>$+ 4$</td>
</tr>
</tbody>
</table>
Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of $\omega$

**Example**

\[
\begin{align*}
\mathbb{[}-\infty, +\infty]\, \mapsto \quad & o_1 \quad \triangleq \quad \omega^2 \cdot x_1 \quad + \quad \omega \cdot x_2 \quad + \quad 3 \\
\mathbb{[}-\infty, +\infty]\, \mapsto \quad & o_2 \quad \triangleq \quad \omega^2 \cdot x_1 \quad + \quad \omega \cdot (-x_2) \quad + \quad 4 \\
\mathbb{[}-\infty, +\infty]\, \mapsto \quad & o_1 \, \sqcup_{O} \, o_2 \quad \triangleq \quad \omega^{2 \cdot 1} \quad + \quad \omega \cdot 0 \quad + \quad 4
\end{align*}
\]

$\omega \cdot \omega = \omega^2 \cdot 1 + \omega \cdot 0$
Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of $\omega$

| Example |
|------------------|------------------|------------------|------------------|
| $[-\infty, +\infty]$ $\mapsto$ $o_1$ $\triangleq$ $\omega^2 \cdot x_1$ $+$ $\omega \cdot x_2$ $+$ $3$ |
| $[-\infty, +\infty]$ $\mapsto$ $o_2$ $\triangleq$ $\omega^2 \cdot x_1$ $+$ $\omega \cdot (\neg x_2)$ $+$ $4$ |
| $[-\infty, +\infty]$ $\mapsto$ $o_1 \sqcup_0 o_2$ $\triangleq$ $\omega^2 \cdot x_1$ $\omega^2 \cdot 1$ $+$ $4$ |
Join

- $\text{join: } \sqcup_A$
- $\text{join: } \sqcup_O$
- $\sqcup_A$ in ascending powers of $\omega$

**Example**

\[
\begin{align*}
[\infty, +\infty] & \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 \\
[\infty, +\infty] & \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 \\
[\infty, +\infty] & \mapsto o_1 \sqcup_o o_2 \triangleq \omega^2 \cdot (x_1 + 1) + 4
\end{align*}
\]
Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of $\omega$

**Example**

\[
\begin{align*}
[\infty, +\infty] & \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 \\
[\infty, +\infty] & \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 \\
[\infty, +\infty] & \mapsto o_1 \sqcup o_2 \triangleq \omega^2 \cdot (x_1 + 1) + 4
\end{align*}
\]
Theorem (Soundness)

the abstract termination semantics is sound
to prove the termination of programs
Example

\[ f_1(x_1, x_2) = \begin{cases} 
1 & x_1 \leq 0 \lor x_2 \leq 0 \\
\omega \cdot (x_1 - 1) + 7x_1 + 3x_2 - 5 & x_1 > 0 \land x_2 > 0 
\end{cases} \]
Example

\[
\begin{align*}
\text{int} : x_1, x_2 \\
\text{while} & \ (x_1 \neq 0 \land x_2 > 0) \ \text{do} \\
\ & \ 	ext{if} \ (x_1 > 0) \ \text{then} \\
\ & \quad \ 	ext{if} \ (\ ? ) \ \text{then} \\
\ & \quad \quad x_1 := x_1 - 1 \\
\ & \quad \ x_2 := \ ? \\
\ & \ 	ext{else} \\
\ & \quad x_2 := x_2 - 1 \\
\ & \ 	ext{else} \\
\ & \ 	ext{if} \ (\ ? ) \ \text{then} \\
\ & \quad x_1 := x_1 + 1 \\
\ & \ 	ext{else} \\
\ & \quad x_2 := x_2 - 1 \\
\ & \ 	ext{od}
\end{align*}
\]

f_1(x_1, x_2) = \begin{cases} 
\omega^2 + \omega \cdot (x_2 - 1) - 4x_1 + 9x_2 - 2 & x_1 < 0 \land x_2 > 0 \\
1 & x_1 = 0 \lor x_2 \leq 0 \\
\omega \cdot (x_1 - 1) + 9x_1 + 4x_2 - 7 & x_1 > 0 \land x_2 > 0 
\end{cases}
Example

int : $x_1, x_2$
while $(x_1 \neq 0 \land x_2 > 0)$ do
  if $(x_1 > 0)$ then
    if $(?)$ then
      $x_1 := x_1 - 1$
      $x_2 := ?$
    else
      $x_2 := x_2 - 1$
  else /* $x_1 < 0$ */
    if $(?)$ then
      $x_1 := x_1 + 1$
    else
      $x_2 := x_2 - 1$
      $x_1 := ?$

the coefficients and their order are inferred by the analysis

$$f_1(x_1, x_2) = \begin{cases} 
  \omega^2 + \omega \cdot (x_2 - 1) - 4x_1 + 9x_2 - 2 & x_1 < 0 \land x_2 > 0 \\
  1 & x_1 = 0 \lor x_2 \leq 0 \\
  \omega \cdot (x_1 - 1) + 9x_1 + 4x_2 - 7 & x_1 > 0 \land x_2 > 0 
\end{cases}$$
Non-Linear Ranking Functions

Example

\[
\begin{align*}
\text{int : } & N, \; x_1, \; x_2 \\
& 1 \; x_1 := N \\
\text{while } & 2 (x_1 \geq 0) \text{ do} \\
& 3 \; x_2 := N \\
\text{while } & 4 (x_2 \geq 0) \text{ do} \\
& 5 \; x_2 := x_2 - 1 \\
\text{od} \\
& 6 \; \text{while } \; x_1 \geq 0 \text{ do} \\
& 7 \; x_1 := x_1 - 1 \\
\text{od} \\
\end{align*}
\]

\[
f_1(x_1, x_2, N) = \begin{cases} 
1 & x_1 < 0 \\
\omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0
\end{cases}
\]

the loop terminates in a finite number of iterations.
Non-Linear Ranking Functions

Example

\begin{align*}
\text{int : } & N, \ x_1, \ x_2 \\
1 \ x_1 & := \ N \\
\text{while } & 2 (x_1 \geq 0) \text{ do} \\
3 \ x_2 & := \ N \\
\text{while } & 4 (x_2 \geq 0) \text{ do} \\
5 \ x_2 & := x_2 - 1 \\
\text{od} & 6 \\
7 \ x_1 & := x_1 - 1 \\
\text{od} & 8
\end{align*}

\[ f_1(x_1, x_2, N) = \begin{cases} 
1 & x_1 < 0 \\
\omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0
\end{cases} \]

the loop terminates in a finite number of iterations
\textbf{remark}: the intervals abstract domain is not sufficient!

\begin{align*}
\text{Example} \quad \text{int : } x, \ y, \ r \\
\text{while } (r > 0) \text{ do} \\
\quad r := r + x \\
\quad r := r - y \\
\text{od}
\end{align*}

\[ f(x, y, r) = \begin{cases} 
1 & r \leq 0 \\
3r + 1 & r > 0 \land x \leq y \\
\text{undefined} & r > 0 \land x \geq y
\end{cases} \]

\[ \forall \triangleq \mathcal{P}(S \times \mathcal{F}) \text{ when } S \triangleq \text{Octagons/Polyhedra \ does not scale!} \]
States Abstract Domain
- $S \triangleq$ Intervals/Octagons/Polyhedra Abstract Domain

Functions Abstract Domain
- $\mathcal{F} \triangleq$ Natural/Ordinal-Valued Ranking Functions Abstract Domain

Piecewise-Defined Ranking Functions Abstract Domain
- $\mathcal{T} \triangleq \{ \text{LEAF}: f \mid f \in \mathcal{F} \} \cup \{ \text{NODE}\{s\}: t_1, t_2 \mid s \in S \wedge t_1, t_2 \in \mathcal{T} \}$

Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)
Example

\[
\text{int} : x \\
\text{while } 1(x > 0 \land y > 0) \text{ do} \\
\quad 2x := x - y \\
\text{od} 3
\]

we map each point to a function of \(x\) and \(y\) giving an upper bound on the steps before termination.
**Example**

\[
\text{int : } x \\
\text{while } 1(x > 0 \land y > 0) \text{ do} \\
\hspace{1em} 2x := x - y \\
\text{od}^3
\]

1. \(x \leq 0 \lor y \leq 0\)
2. \(x > 0 \land y > 0\)
3. \(x := x - y\)

We start at the end with 0 steps before termination.
we take into account $x \leq 0$ and we have 1 step to termination

Example

```plaintext
int : x
while $1(x > 0 \land y > 0)$ do
  $2x := x - y$
  $od^3$
```
Example

\[ \text{int : } x \]
\[ \text{while } 1(x > 0 \land y > 0) \text{ do} \]
\[ 2 \quad x := x - y \]
\[ \text{od}^3 \]

we consider the assignment \[ x := x - 1 \] and we are at 2 steps to termination
Example

\textbf{int : } x
\textbf{while } 1(x > 0 \land y > 0) \textbf{ do}
  \textbf{2 } x := x - y
\textbf{od}

\vspace{1cm}

we consider \( x > 0 \) and we do the join
Example

\[
\begin{align*}
\text{int} & : x \\
\text{while } & (x > 0 \land y > 0) \text{ do} \\
\text{2} & x := x - y \\
\text{od} & \text{3}
\end{align*}
\]
Example

\[
\begin{align*}
\text{int} : x & \\
\text{while } (x > 0 \land y > 0) \text{ do} & \\
2 & x := x - y \\
\text{od} &
\end{align*}
\]
Example

\[ \text{int: } x \]
\[ \text{while } 1(x > 0 \land y > 0) \text{ do} \]
\[ 2x := x - y \]
\[ \text{od} 3 \]

\[ x \leq 0 \lor y \leq 0 \]
\[ x > 0 \land y > 0 \]
\[ x := x - y \]
the analysis gives **true** as **sufficient precondition** for termination

**Example**

```plaintext
int : x
while \(1(x > 0 \land y > 0)\) do
  \(x := x - y\)
  \(x > 0 \land y > 0\)
  \(x := x - y\)
  \(x \leq 0 \lor y \leq 0\)
```

```
0
0
0
0
```

```
0
0
0
0
```

```
0
0
0
0
```

```
0
0
0
0
```
Theorem (Soundness)

_the abstract termination semantics is **sound**
to prove the termination of programs_
An Abstract Domain Functor for Termination

Welcome to FuncTion's web interface!

Type your program:

or choose a predefined example: Choose File

and choose an entry point: main

Analyze

Forward option(s):
- Widening delay: 2

Backward option(s):
- Partition Abstract Domain: Intervals
- Function Abstract Domain: Affine Functions
  - Ordinal-Valued Functions
  - Maximum Degree: 2
  - Widening delay: 2
SV-COMP 2014

3rd Intl. Competition on Software Verification held at TACAS 2014 in Grenoble, France.

The results of the 2014 competition are available in the competition report.

New: SV-COMP 2014 is presented at the FLoC Olympic Games.

Motivation

Competition is a driving force for the invention of new methods, technologies, and tools. This web page describes the competition of software-verification tools, which will take place at TACAS.

There are several new and powerful software-verification tools around, but they are very difficult to compare. The reason is that so far no widely distributed benchmark suite of verification tasks was available and most concepts are only validated in research prototypes. This competition has changed this: Now there is an established set of verification tasks for comparing software verifiers, and the tools are publicized on the SV-COMP web site.

Only few projects aim at producing stable tools that can be used by people outside the respective development groups, and the development of such tools is not continuous. Also, PhD students and PostDocs do not adequately benefit from tool development because theoretical papers count more than papers that present technical contributions, like tool papers. Through its visibility, this competition wants to change this, showing off the latest implementation of the research results in our community, and give credits and benefits to researchers and students who spend considerable amounts of time developing verification algorithms and software packages.

Goals of the Competition

- Provide a snapshot of the state-of-the-art in software verification to the community. That means to compare, independently from particular paper projects and specific techniques, different software verification algorithms.

Example:

```c
int x;
while (x <= 1000) do
  if (?)
    x := 2 * x + 2;
  else
    x := 3 * x^2;
  fi;
od;
```
SV-COMP 2014
Example

```c
int : x
while (x ≤ 1000) do
    if ( ? ) then
        x := −2 · x + 2
    else
        x := −3 · x − 2
    fi
od
```
Experiments

**Benchmark:** 87 terminating C programs collected from the literature

**Tools:**
- FuncTion
- AProVE
- T2
- Ultimate Büchi Automizer

**Result:**

<table>
<thead>
<tr>
<th></th>
<th>Tot</th>
<th>FuncTion</th>
<th>AProVE</th>
<th>T2</th>
<th>Ultimate</th>
<th>Time</th>
<th>Timeouts</th>
</tr>
</thead>
<tbody>
<tr>
<td>FuncTion</td>
<td>51</td>
<td>–</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>6s</td>
<td>5</td>
</tr>
<tr>
<td>AProVE</td>
<td>60</td>
<td>17</td>
<td>–</td>
<td>7</td>
<td>2</td>
<td>35s</td>
<td>19</td>
</tr>
<tr>
<td>T2</td>
<td>73</td>
<td>30</td>
<td>20</td>
<td>–</td>
<td>3</td>
<td>2s</td>
<td>0</td>
</tr>
<tr>
<td>Ultimate</td>
<td>79</td>
<td>31</td>
<td>21</td>
<td>9</td>
<td>–</td>
<td>9s</td>
<td>1</td>
</tr>
</tbody>
</table>
Conclusions

- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - sufficient preconditions for termination
- instances based on **ordinal-valued functions**
  - lexicographic orders automatically inferred by the analysis
  - analysis not limited to programs with linear ranking functions
- instances based on **decision trees**

Future Work

- more abstract domains
  - non-linear ranking functions
  - better widening
- **fair termination**
- other **liveness** properties
Conclusions

- family of \textbf{abstract domains} for program termination
  - piecewise-defined ranking functions
  - sufficient preconditions for termination
- instances based on \textbf{ordinal-valued functions}
  - lexicographic orders automatically inferred by the analysis
  - analysis not limited to programs with linear ranking functions
- instances based on \textbf{decision trees}

Future Work

- \textbf{more abstract domains}
  - non-linear ranking functions
  - better widening
- \textbf{fair termination}
- other \textbf{liveness} properties
Thank You!