To Infinity... and Beyond!\textsuperscript{1}

Caterina Urban and Antoine Miné

École Normale Supérieure & CNRS & INRIA
Paris, France

WST 2014
Vienna, Austria

\textsuperscript{1}Urban\&Miné - An Abstract Domain to Infer Ordinal-Valued Ranking Functions (ESOP 2014)
**Outline**

- **ranking functions**\(^2\)
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below

- **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)

- family of **abstract domains** for program termination\(^3\)
  - piecewise-defined ranking functions
  - instances based on ordinal-valued ranking functions\(^4\)

---

\(^2\)Floyd - *Assigning Meanings to Programs* (1967)

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

\(^4\)Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)
● **ranking functions**$^2$
  ● functions that strictly **decrease** at each program step...
  ● ...and that are **bounded** from below

● **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)

● family of **abstract domains** for program termination$^3$
  ● piecewise-defined ranking functions
  ● instances based on **ordinal-valued ranking functions**$^4$

---

$^2$Floyd - *Assigning Meanings to Programs* (1967)

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

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

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

- **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)

- family of abstract domains for program termination
  - piecewise-defined ranking functions
  - instances based on ordinal-valued ranking functions

---

2 Floyd - *Assigning Meanings to Programs* (1967)
3 Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
4 Urban&Míné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)
Abstract Interpretation\(^5\)

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

\[ S \]

\[ \checkmark \]

\[ \times \]

\[ [P] \]

\(^5\)Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)
Abstract Interpretation\(^5\)

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

\[ \gamma \]

\[ S \]

\[ [P] \]

\[ S^\# \]

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

\[ [P]^\# \]

\[^5\] Cousot & Cousot - Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. (POPL 1977)
Abstract Interpretation\(^5\)

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

\[ \gamma \]

\[ \gamma([P]#) \]

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

\[ [P]# \]

\[ S \]

\[ S# \]

---

\(^5\) Cousot&Cousot - Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. (POPL 1977)
\[ \langle C, \sqsubseteq_C \rangle \quad \gamma([P]#) \quad \langle A, \sqsubseteq_A \rangle \]

\[ S \quad \checkmark \quad [P] \quad S# \quad \checkmark \quad [P]# \]
\[ \langle C, \sqsubseteq_C \rangle \quad \gamma([P]\#) \quad \gamma \quad \langle A, \sqsubseteq_A \rangle \]

\[ S \quad [P] \quad \checkmark \]

\[ \gamma([P]\#) \quad \gamma \quad \gamma([P]\#) \quad \gamma \quad S\# \quad [P]\# \quad \checkmark \]

\[ S \quad [P] \quad \times \]

\[ S \quad [P] \quad \checkmark \quad \times \quad S\# \quad [P]\# \quad \times \]
Termination Semantics
program $\mapsto$ trace semantics

finite traces $\Sigma^+$

infinite traces $\Sigma^\infty$

$\beta$ final states

$\Sigma$ states

$\tau$ transition relation
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)
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**

```
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)
Example

\begin{verbatim}
int : x
x := ?
while (x > 0) do
  x := x - 1
od
\end{verbatim}

The termination semantics it is not computable!
Example

```plaintext
int : x
x := ?
while (x > 0) do
  x := x - 1
od
```

the termination semantics needs **ordinals**!

the termination semantics it is **not computable**!
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

Abstract Termination Semantics

- 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

\[ \langle \mathcal{V}, \sqsubseteq \mathcal{V} \rangle \]

- 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)
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
- \( V(S, F) \)
Affine Ranking Functions Abstract Domain

- States Abstract Domain
  - $S \triangleq$ Intervals Abstract Domain
- (Natural-Valued) Functions Abstract Domain
  - $\mathcal{F} \triangleq \{ \bot_{F} \} \cup \{ f \mid f \in \mathbb{Z}^{n} \rightarrow \mathbb{N} \} \cup \{ \top_{F} \}$
    where $f \equiv f(x_1, \ldots, x_n) = m_1 x_1 + \cdots + m_n x_n + q$

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

\[
\text{int : } x \\
\text{while } 1(x \leq 10) \text{ do} \\
\quad \text{if } 2(x > 6) \text{ then} \\
\quad \quad 3x := x + 2 \\
\quad \text{fi} \\
\text{od}
\]

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

\[
\begin{align*}
\text{int} : & x \\
\text{while } & (x \leq 10) \text{ do} \\
\text{if } & (x > 6) \text{ then} \\
& x := x + 2 \\
\text{fi} \\
\text{od}
\end{align*}
\]

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

int : x
while 1(x ≤ 10) do
  if 2(x > 6) then
    3x := x + 2
  fi
od

we map each point to a function of x giving an upper bound on the steps before termination
Example

```plaintext
int : x
while 1 (x ≤ 10) do
  if 2 (x > 6) then
    3 x := x + 2
  fi
od
```

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

\begin{verbatim}
int : x
while 1(x ≤ 10) do
  if 2(x > 6) then
    3x := x + 2
  fi
od
\end{verbatim}

we map each point to a function of x giving an upper bound on the steps before termination

The analysis provides \( x > 6 \) as sufficient precondition for termination.
**remark**: natural-valued ranking functions are not sufficient

Example

```
int : x
x := ?
while (x > 0) do
  x := x - 1
od
```
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
- \( V(S, F) \)

---

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

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

- States Abstract Domain
- **Natural-Valued** Functions Abstract Domain
- **Ordinal-Valued** Functions Abstract Domain
- Piecewise-Defined Ranking Functions Abstract Domain

\[ V(S, O(F)) \]

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

Finite ordinals

Transfinite ordinals

Successor ordinals

\[ \text{succ}(\alpha) \equiv \alpha \cup \{\alpha\} \]

Limit ordinals
Ordinals

Finite ordinals

Transfinite ordinals
Ordinals

\[ \text{successor ordinals } \quad \text{succ}(\alpha) \triangleq \alpha \cup \{\alpha\} \]

\textit{finite} ordinals

\textit{transfinite} ordinals
Ordinals

\[ \text{succ}(\alpha) \triangleq \alpha \cup \{\alpha\} \]

- **Finite ordinals**
- **Limit ordinals**
- **Transfinite ordinals**
Ordinal Arithmetic

- **addition**

\[\alpha + 0 = \alpha\]  
\[\alpha + \text{succ}(\beta) = \text{succ}(\alpha + \beta)\]  
\[\alpha + \beta = \bigcup_{\gamma<\beta}(\alpha + \gamma)\]  

- Associative: \((\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)\)
- Not commutative: \(1 + \omega = \omega \neq \omega + 1\)

- **multiplication**
Ordinal Arithmetic

- **addition**

- **multiplication**

\[
\alpha \cdot 0 = 0 \quad \text{ (zero case)}
\]

\[
\alpha \cdot \text{succ}(\beta) = (\alpha \cdot \beta) + \alpha \quad \text{ (successor case)}
\]

\[
\alpha \cdot \beta = \bigcup_{\gamma < \beta} (\alpha \cdot \gamma) \quad \text{ (limit case)}
\]

- associative: \((\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)\)
- left distributive: \(\alpha \cdot (\beta + \gamma) = (\alpha \cdot \beta) + (\alpha \cdot \gamma)\)
- not commutative: \(2 \cdot \omega = \omega \neq \omega \cdot 2\)
- not right distributive: \((\omega + 1) \cdot \omega = \omega \cdot \omega \neq \omega \cdot \omega + \omega\)
Ordinal-Valued Ranking Functions Domain

- States Abstract Domain
  - $S \triangleq$ Intervals Abstract Domain
- Natural-Valued Functions Abstract Domain
  - $\mathcal{F} \triangleq$ Affine Ranking Functions Abstract Domain
- Ordinal-Valued Functions Abstract Domain
  - $\mathcal{O} \triangleq \{\bot_\mathcal{O}\} \cup \{\sum \omega^i \cdot f_i \mid f_i \in \mathcal{F} \setminus \{\bot_{\mathcal{F}}, \top_{\mathcal{F}}\}\} \cup \{\top_{\mathcal{O}}\}$
Backward Assignments

- assignment transfer functions amount to weakest preconditions

**Example**

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

\[
\downarrow \quad x_1 := x_1 + x_2
\]

\[
[\infty, +\infty] \mapsto o \triangleq ?
\]

- the resulting covering is refined to obtain a partition
Backward Assignments

- assignment transfer functions amount to weakest preconditions

**Example**

\[
(\mathbb{R}, +) \quad o \quad \triangleq \quad \omega \cdot (x_1 - x_2) \quad + \quad x_1
\]

\[
\downarrow \quad x_1 := x_1 + x_2
\]

\[
[\mathbb{R}, +) \quad o \quad \triangleq \quad \omega \cdot (x_1 - x_2) \quad + \quad 1
\]

- the resulting covering is refined to obtain a partition
Backward Assignments

- assignment transfer functions amount to weakest preconditions

Example

\[
\begin{align*}
[\infty, +\infty] & \mapsto o \triangleq \omega \cdot (x_1 - x_2) + x_1 \\
\downarrow & \quad x_1 := x_1 + x_2 \\
[\infty, +\infty] & \mapsto o \triangleq \omega \cdot (x_1 + x_2 - x_2) + x_1 + x_2 + 1
\end{align*}
\]

- the resulting covering is refined to obtain a partition
Backward Assignments

- assignment transfer functions amount to weakest preconditions

**Example**

\[
\begin{align*}
[-\infty, +\infty] \mapsto o & \triangleq \omega \cdot (x_1 - x_2) + x_1 \\
\downarrow & \quad x_1 := x_1 + x_2 \\
[-\infty, +\infty] \mapsto o & \triangleq \omega \cdot x_1 + x_1 + x_2 + 1
\end{align*}
\]

- the resulting covering is refined to obtain a partition
Backward Assignments

- assignment transfer functions amount to weakest preconditions
- the resulting covering is refined to obtain a partition

**Example**

\[ x := x + [1, 5] \]
Backward Assignments

- assignment transfer functions amount to weakest preconditions
- the resulting covering is refined to obtain a partition

Example

\[ x := x + [1, 5] \]

\[ x \rightarrow [7, 12] \]

\[ x \rightarrow [6, 11] \]
non-deterministic assignments are carried out in ascending powers of $\omega$

Example

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

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

Example

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

$\Downarrow$

\[
x_1 := ?
\]

\[
[\infty, +\infty] \mapsto o \triangleq + 1
\]
Backward Non-Deterministic Assignments

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

Example

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

$$\Downarrow x_1 := ?$$

$$\left[ -\infty, +\infty \right] \mapsto o \triangleq + x_2 + 1$$
non-deterministic assignments are carried out in ascending powers of $\omega$

**Example**

\[
\begin{align*}
[\infty, +\infty] & \mapsto o \triangleq \omega \cdot x_1 + x_2 \\
\downarrow & \quad x_1 := ? \\
[\infty, +\infty] & \mapsto o \triangleq \omega^2 \cdot 0^{1} + \omega \cdot 0 + x_2 + 1
\end{align*}
\]

\[
\omega^k \cdot \omega = \omega^{k+1} \cdot 1 + \omega^k \cdot 0 = \omega^{k+1}
\]
non-deterministic assignments are carried out in ascending powers of $\omega$

**Example**

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

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

\[
[-\infty, +\infty] \mapsto o \triangleq \omega^2 \cdot 1 + \omega \cdot 0 + x_2 + 1
\]
non-deterministic assignments are carried out in ascending powers of $\omega$

Example

\[
\begin{align*}
[\infty, +\infty] & \mapsto o \triangleq \omega \cdot x_1 + x_2 \\
\downarrow & \quad x_1 := ? \\
[\infty, +\infty] & \mapsto o \triangleq \omega^2 + x_2 + 1
\end{align*}
\]
Join

- segmentation unification: □

**Example**

- join: □_F
- join: □_O
Join

- segmentation unification: \( \sqcup \)
- join: \( \sqcup_F \)

Example

- join: \( \sqcup_O \)
Join

- segmentation unification: \( \sqcup \)
- join: \( \sqcup_F \)
- join: \( \sqcup_O \)
  - \( \sqcup_F \) in ascending powers of \( \omega \)

Example

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

- segmentation unification: \( \sqcup \)
- join: \( \sqcup_F \)
- join: \( \sqcup_O \)
  - \( \sqcup_F \) in ascending powers of \( \omega \)

### Example

<table>
<thead>
<tr>
<th>Interval</th>
<th>Formula 1</th>
<th>Formula 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>( [-\infty, +\infty] )</td>
<td>( \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 )</td>
<td>( \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 )</td>
<td>4</td>
</tr>
<tr>
<td>( [-\infty, +\infty] )</td>
<td>( \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 )</td>
<td>( \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 )</td>
<td>4</td>
</tr>
<tr>
<td>( [-\infty, +\infty] )</td>
<td>( \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 )</td>
<td>( \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 )</td>
<td>4</td>
</tr>
</tbody>
</table>
Join

- segmentation unification: $\sqcup$
- join: $\sqcup_F$
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of $\omega$

**Example**

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

\[\omega^k \cdot \omega = \omega^{k+1} \cdot 1 + \omega^k \cdot 0 = \omega^{k+1}\]
Join

- segmentation unification: ⊔
- join: ⊔_F
- join: ⊔_O
  - ⊔_F in ascending powers of ω

Example

\[
\begin{align*}
[\infty, +\infty] & \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 \\
[\infty, +\infty] & \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 \\
[\infty, +\infty] & \mapsto o_1 \sqcup_0 o_2 \triangleq \omega^2 \cdot x_1^1 + \omega \cdot 0 + 4
\end{align*}
\]
Join

- segmentation unification: ⊔
- join: ⊔_F
- join: ⊔_O
  - ⊔_F in ascending powers of ω

Example

\[
\begin{align*}
[-\infty, +\infty] & \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 \\
[-\infty, +\infty] & \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 \\
[-\infty, +\infty] & \mapsto o_1 \sqcup_0 o_2 \triangleq \omega^2 \cdot (x_1 + 1) + \omega \cdot 0 + 4
\end{align*}
\]
Join

- segmentation unification: $\sqcup$
- join: $\sqcup_F$
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of $\omega$

**Example**

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

- segmentation left-unification: $\nabla$

Example

- widening: $\nabla_F$
- widening: $\nabla_O$
Widening

- segmentation left-unification: $\nabla$
- widening: $\nabla_F$

Example

- unstable ranking functions yield $\top_F$
- widening: $\nabla_O$
Widening

- segmentation left-unification: $\nabla$
- widening: $\nabla_F$

**Example**

- widening: $\nabla_O$
  - $\nabla_F$ in ascending powers of $\omega$
  - unstable ranking functions yield $\top_O$
Termination Semantics
\[ \langle \Sigma \rightarrow \emptyset, \sqsubseteq \rangle \]

Abstract Termination Semantics
\[ \langle \mathcal{V}, \sqsubseteq \rangle \]

Theorem (Soundness)

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

\[
\text{int : } x_1, x_2 \\
\text{while } 1(x_1 > 0 \land x_2 > 0) \text{ do} \\
\quad \text{if } 2(\ ? ) \text{ then} \\
\qquad 3 x_1 := x_1 - 1 \\
\qquad 4 x_2 := ? \\
\quad \text{else} \\
\qquad 5 x_2 := x_2 - 1 \\
\text{od}
\]

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

24 / 31
### Example

```plaintext
int : x1, x2
while 1(x1 ≠ 0 ∧ x2 > 0) do
  if 2(x1 > 0) then
    if 3(?) then
      4 x1 := x1 - 1
      5 x2 := ?
    else
      6 x2 := x2 - 1
  else /* x1 < 0 */
    if 7(?) then
      8 x1 := x1 + 1
    else
      9 x2 := x2 - 1
  10 x1 := ?
end
```

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

\[
\text{int : } x_1, x_2 \\
\text{while } (x_1 \neq 0 \land x_2 > 0) \text{ do} \\
\quad \text{if } (x_1 > 0) \text{ then} \\
\quad \quad \text{if } ( ? ) \text{ then} \\
\quad \quad \quad x_1 := x_1 - 1 \\
\quad \quad x_2 := ? \\
\quad \text{else} \\
\quad x_2 := x_2 - 1 \\
\text{else} / \ast \ast x_1 < 0 \ast / \\
\quad \text{if } ( ? ) \text{ then} \\
\quad \quad x_1 := x_1 + 1 \\
\quad \text{else} \\
\quad x_2 := x_2 - 1 \\
\quad x_1 := ?
\]

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

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

\[
f_2(x_1, x_2, N) = \begin{cases} 
  1 & x_1 < 0 \\
  \omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0
\end{cases}
\]
Non-Linear Ranking Functions

Example

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

\[
f_2(x_1, x_2, N) = \begin{cases} 
1 & x_1 < 0 \\
\omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0
\end{cases}
\]

the loop terminates in a finite number of iterations
FuncTion: http://www.di.ens.fr/~urban/FuncTion.html
Experiments

**Benchmark:** 38 programs collected from the literature

- 25 always terminating programs
- 13 conditionally terminating programs
- 9 simple loops
- 7 nested loops
- 13 non-deterministic programs

**Result:** proved 30 out of 38 programs

- proved 8 out of 9 simple loops
- proved 4 out of 7 nested loops
  - **ordinals** required for 2 out of 4
- proved 10 out of 13 non-deterministic programs
  - **ordinals** required for 5 out of 10
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

Future Work

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

Future Work

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

Example (Dijkstra’s Random Number Generator)

int : x, b
1 \ x := 0, b := true
while 2 (b) do
  if 3 (?) then
    4 \ x := x + 1
  else
    5 b := false
od 6
Fair Termination

Example (Dijkstra’s Random Number Generator)

```plaintext
int : x, b, z_1, z_2

1 x := 0, b := true, z_1 := [0, +\infty], z_2 := [0, +\infty]

while 2(b) do

  if 3(z_1 \leq z_2) then
    4 x := x + 1, z_1 := [0, +\infty], z_2 := z_2 - 1
  else
    5 b := false, z_2 := [0, +\infty], z_1 := z_1 - 1

od

f_2(x, b, z_1, z_2) = \begin{cases} 
1 & \neg b \\
4 & b \land z_1 > z_2 \\
5z_2 + 9 & b \land z_1 \leq z_2 
\end{cases}
```
Thank You!