To Infinity. . . and Beyond![1]

Caterina Urban and **Antoine Miné**

École Normale Supérieure & CNRS & INRIA
Paris, France

WST 2014
Vienna, Austria

[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)

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)

Outline

- **ranking functions**[2]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below

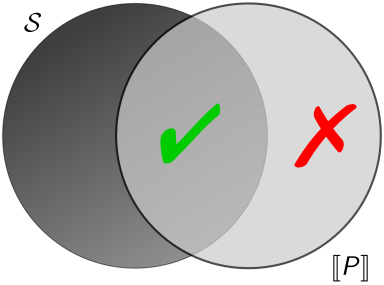
- **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)

- 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)

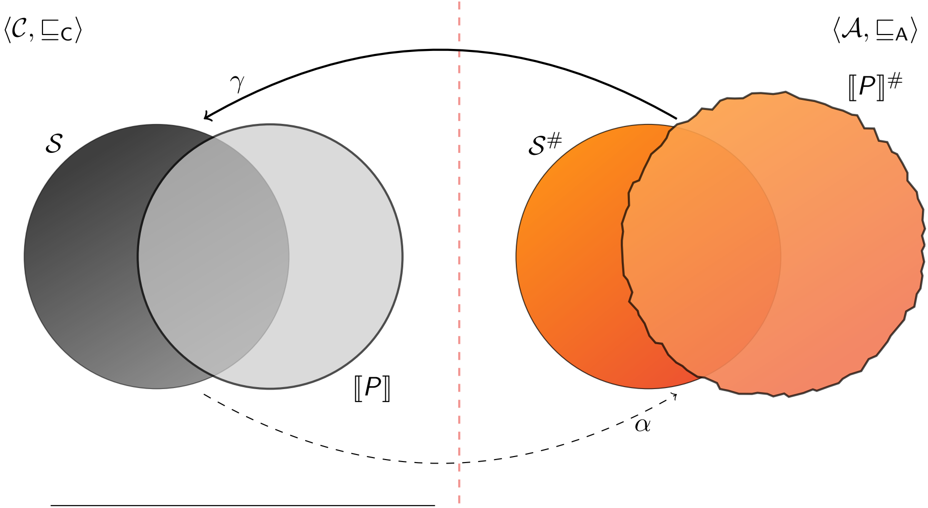
Abstract Interpretation[5]

$\langle \mathcal{C}, \sqsubseteq_{\mathsf{C}} \rangle$



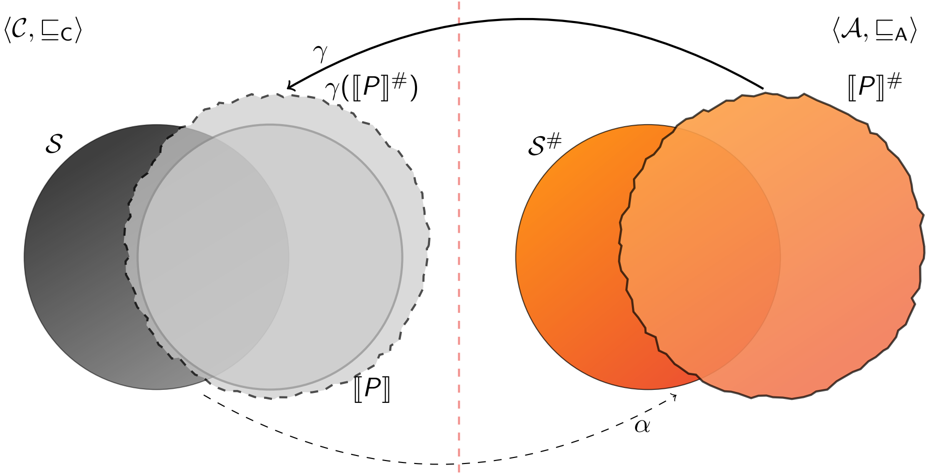
[5] Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)

Abstract Interpretation[5]

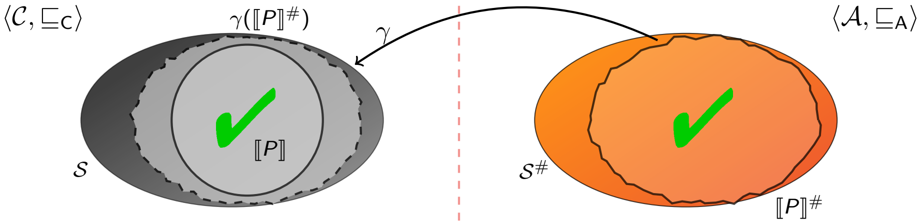


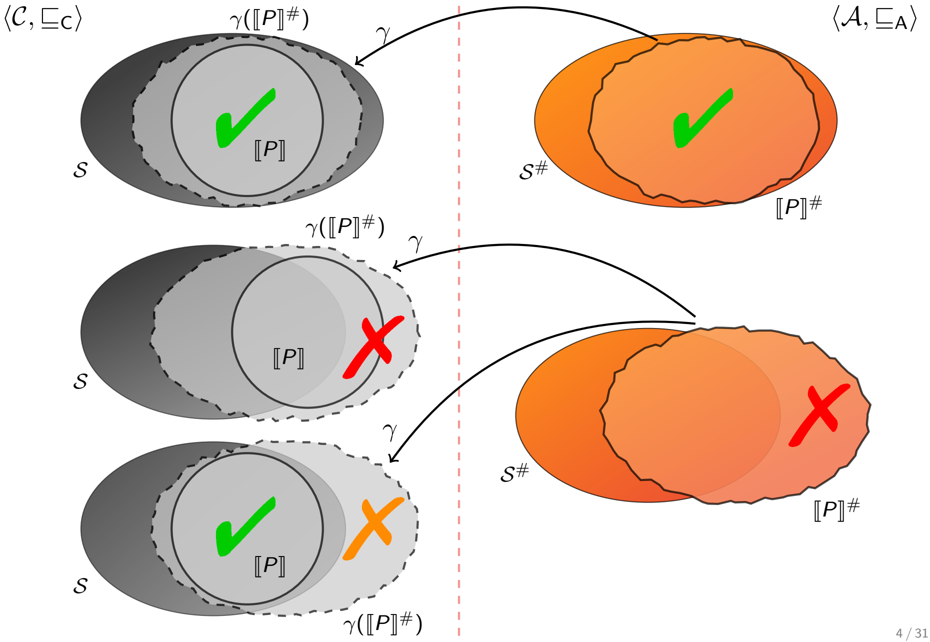
[5] Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)

Abstract Interpretation[5]



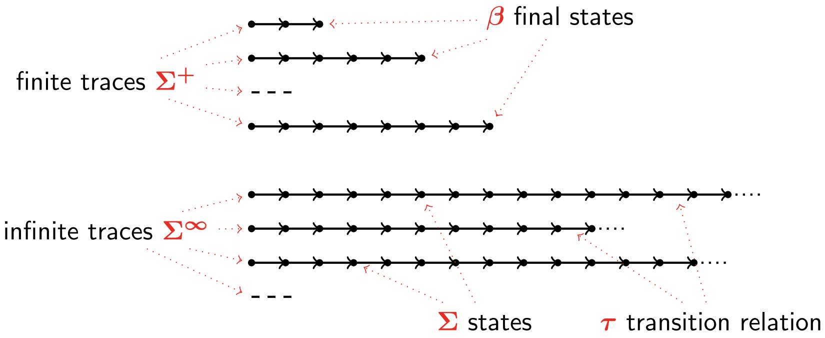
[5] Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)


$\langle \mathcal{C}, \sqsubseteq_C \rangle$
$\gamma(\llbracket P \rrbracket^\#)$
γ
$\langle \mathcal{A}, \sqsubseteq_A \rangle$
$\llbracket P \rrbracket$
$\mathcal{S}$
$\mathcal{S}^\#$
$\llbracket P \rrbracket^\#$


$\langle \mathcal{C}, \sqsubseteq_C \rangle$
$\gamma([\![P]\!]^\#)$
γ
$\langle \mathcal{A}, \sqsubseteq_A \rangle$
$[\![P]\!]$
$\mathcal{S}$
$\mathcal{S}^\#$
$[\![P]\!]^\#$
$\gamma([\![P]\!]^\#)$
γ
$[\![P]\!]$
$\mathcal{S}$
γ
$\mathcal{S}^\#$
$[\![P]\!]^\#$
$[\![P]\!]$
$\mathcal{S}$
$\gamma([\![P]\!]^\#)$

Termination Semantics

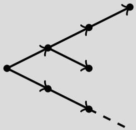
program ↦ **trace semantics**



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)

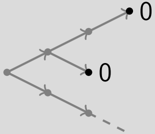
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

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

program $\mapsto$ trace semantics $\mapsto$ **termination semantics**

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)

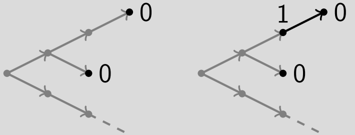
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

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

program $\mapsto$ trace semantics $\mapsto$ **termination semantics**

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)

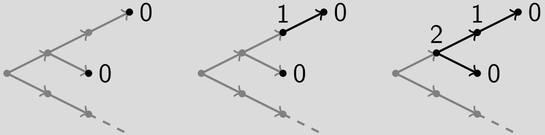
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

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

program ↦ trace semantics ↦ **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)

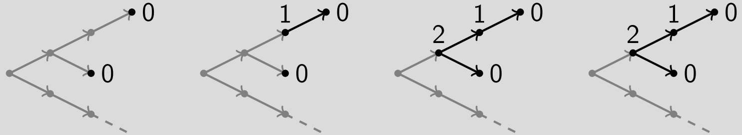
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

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

program $\mapsto$ trace semantics $\mapsto$ **termination semantics**

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)

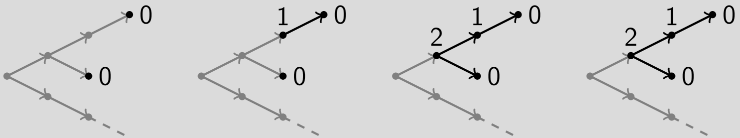
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

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

program $\mapsto$ trace semantics $\mapsto$ **termination semantics**

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)

Example

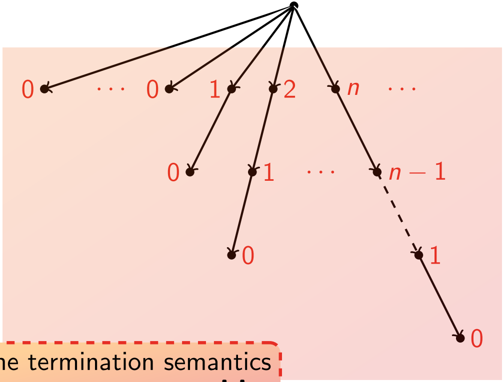
int : x

$x :=\ ?$

while $(x > 0)$ do

  $x := x - 1$

od



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

Example

int : x

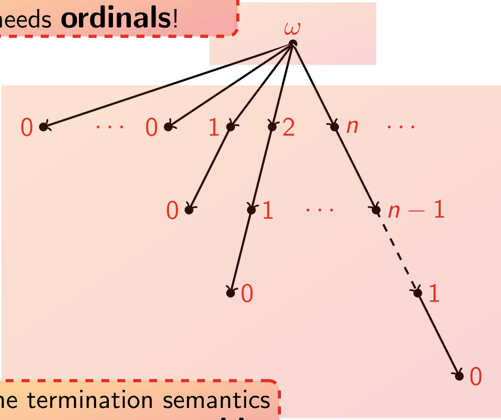
$x :=\ ?$

while $(x > 0)$ do

$\quad x := x - 1$

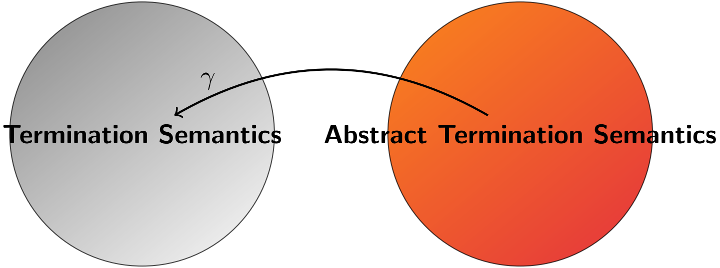
od

the termination semantics needs **ordinals**!



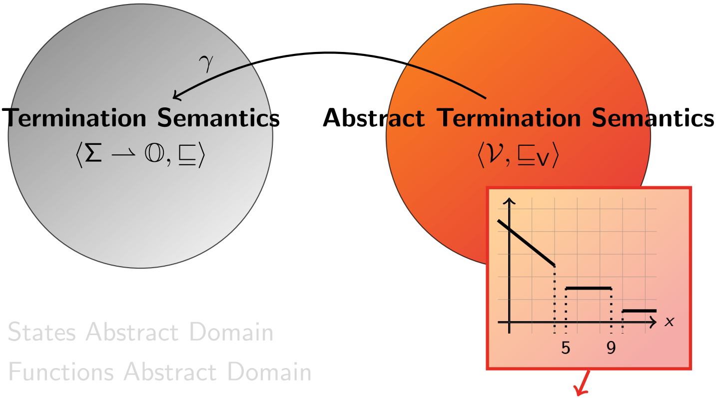
the termination semantics it is **not computable**!

Piecewise-Defined Ranking Functions



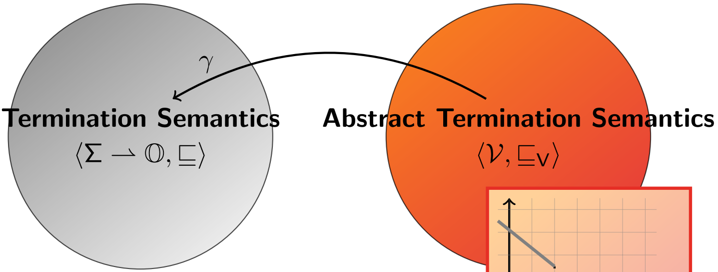
- States Abstract Domain $\mathcal{S}$
- Functions Abstract Domain $\mathcal{F}$
- Piecewise-Defined Ranking Functions Abstract Domain $\mathcal{V}(\mathcal{S}, \mathcal{F})$

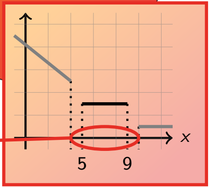
Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)



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

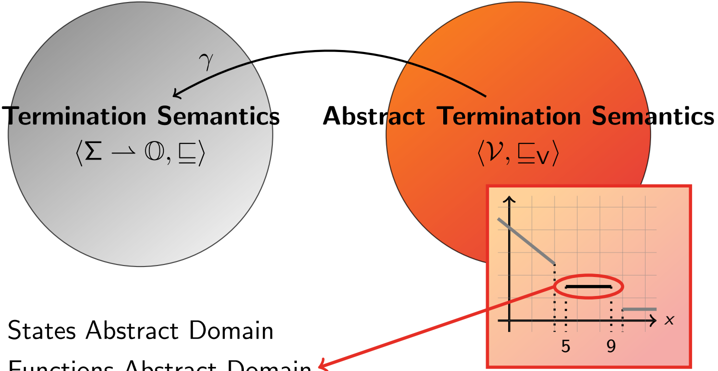
Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)





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

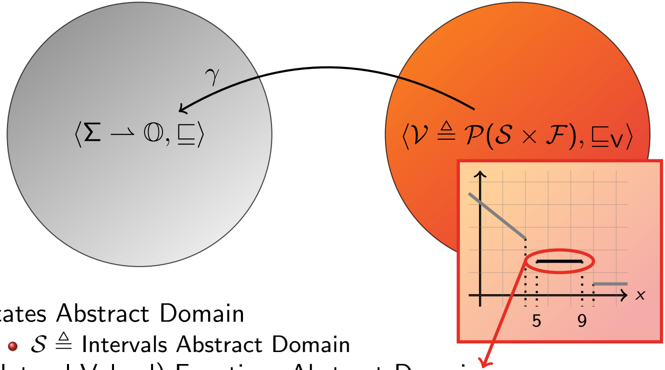
Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)



- 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)

Affine Ranking Functions Abstract Domain



- States Abstract Domain
  - $\mathcal{S} \triangleq$ Intervals Abstract Domain
- (Natural-Valued) Functions Abstract Domain
  - $\mathcal{F} \triangleq \{\bot_F\} \cup \{f \mid f \in \mathbb{Z}^n \to \mathbb{N}\} \cup \{\top_F\}$
    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

int : x

while [1]$(x \leq 10)$ do

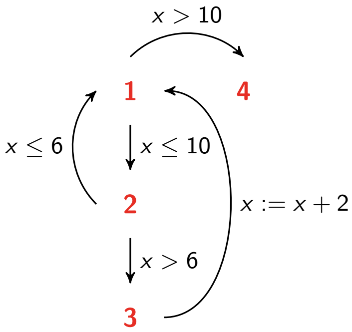
  if [2]$(x > 6)$ then

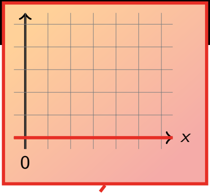
    [3]$x := x + 2$

  fi

od[4]

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





Example

int : x

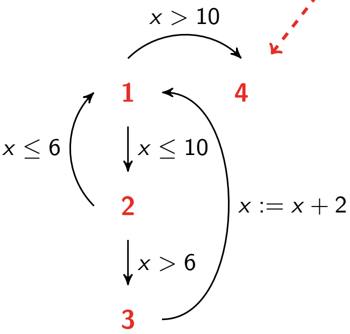
while [1]$(x \leq 10)$ do

if [2]$(x > 6)$ then

[3]$x := x + 2$

fi

od[4]



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

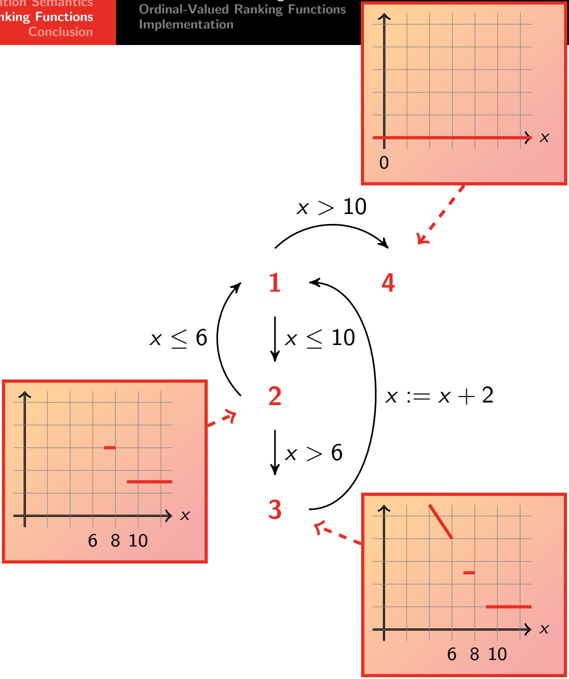
if $^2(x > 6)$ then

$^3x := x + 2$

fi

od^4

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



Example

int : x

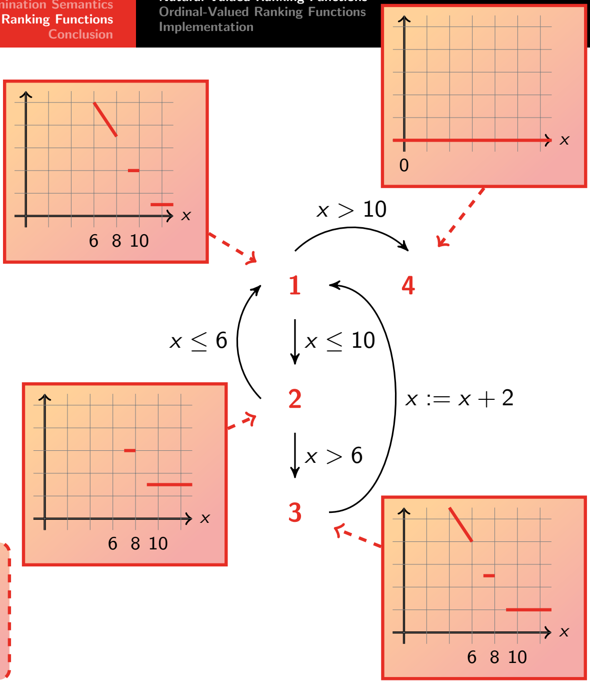
while [1]$(x \leq 10)$ do

  if [2]$(x > 6)$ then

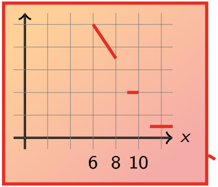
    [3]$x := x + 2$

  fi

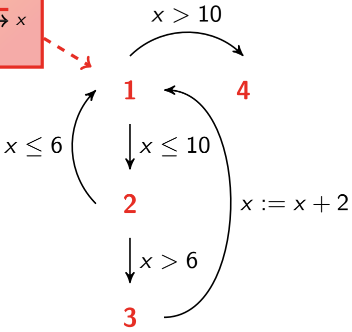
od[4]



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



Example

int : x

while $^{1}(x \leq 10)$ do

if $^{2}(x > 6)$ then

$^{3}x := x + 2$

fi

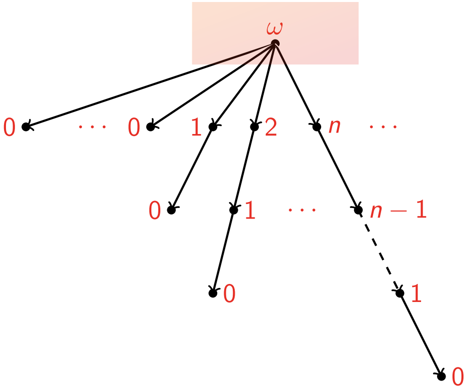
od^{4}

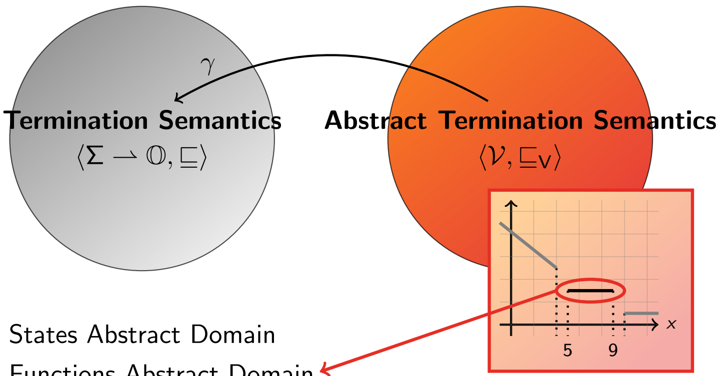
we map each point
to a function of x giving
an **upper bound** on the
steps before termination

- **remark**: natural-valued ranking functions are not sufficient

Example

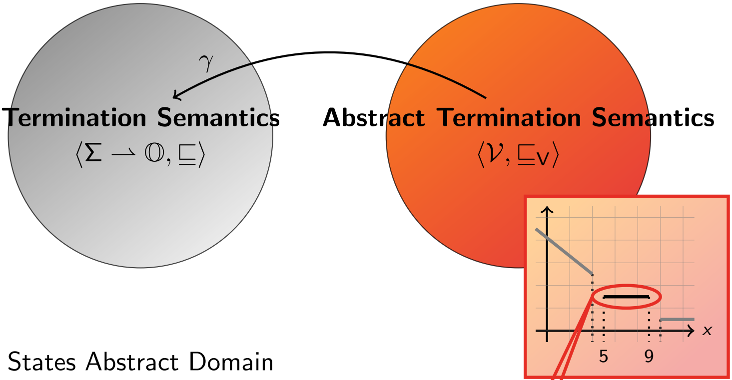
int : x
$x :=\ ?$
while $(x > 0)$ do
  $x := x - 1$
od





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

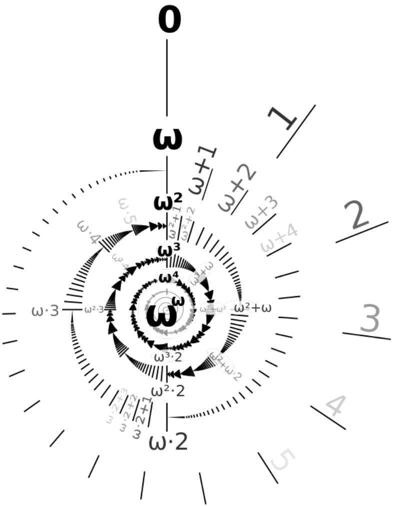
Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)



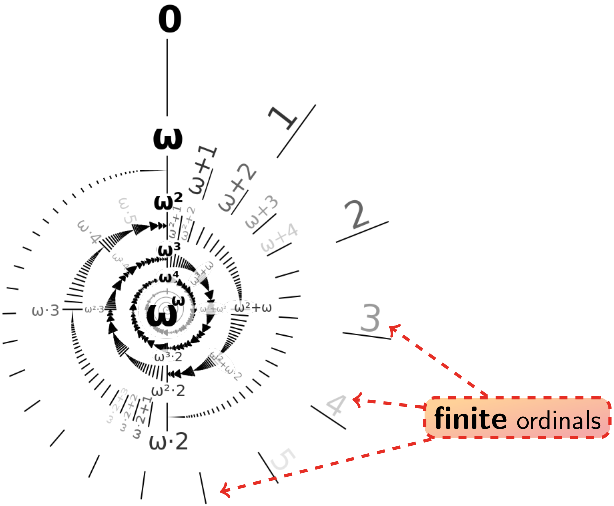
- States Abstract Domain — S
- **Natural-Valued** Functions Abstract Domain — F
- **Ordinal-Valued** Functions Abstract Domain — **O**(F)
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, **O**(F))

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

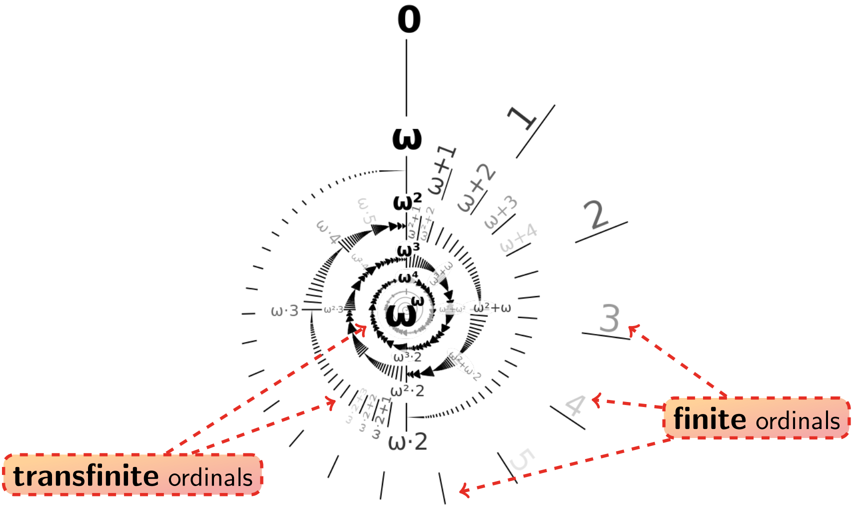
Ordinals



Ordinals

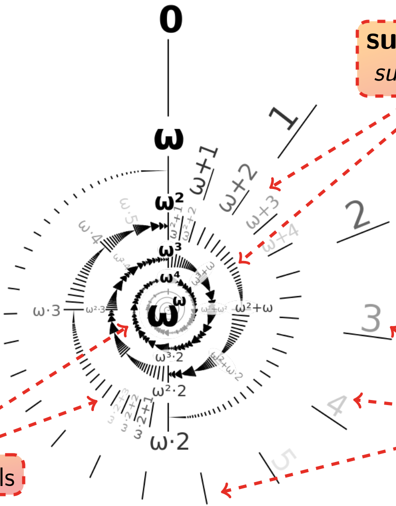

0
ω
1
ω+1
ω+2
2
ω+3
ω+4
ω²
ω³
ω⁴
ω·5
ω·4
ω·3
ω²+ω
3
ω³·2
ω²·2
4
ω·2
5
finite ordinals

Ordinals


0
ω
1
ω+1
ω+2
2
ω+3
ω+4
ω²
ω³
ω⁴
ω·3
ω²+ω
3
ω³·2
ω²·2
4
ω·2
5
finite ordinals
transfinite ordinals

Ordinals

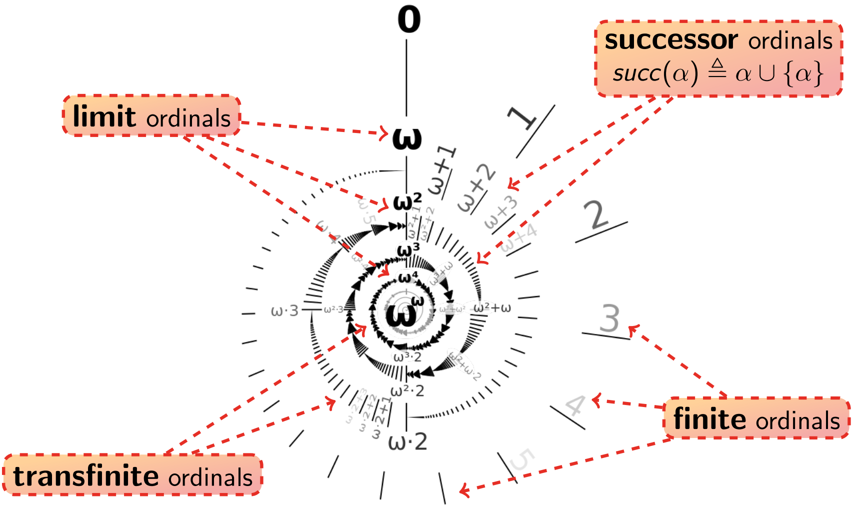
successor ordinals
$succ(\alpha) \triangleq \alpha \cup \{\alpha\}$



finite ordinals

transfinite ordinals

Ordinals


successor ordinals
$succ(\alpha) \triangleq \alpha \cup \{\alpha\}$
limit ordinals
transfinite ordinals
finite ordinals
0
ω
ω+1
ω+2
ω+3
ω+4
1
2
3
4
5
ω²
ω³
ω⁴
ω·2
ω·3
ω²+ω
ω³·2
ω²·2

Ordinal Arithmetic

- **addition**

$$\alpha + 0 = \alpha \qquad \text{(zero case)}$$

$$\alpha + \mathit{succ}(\beta) = \mathit{succ}(\alpha + \beta) \qquad \text{(successor case)}$$

$$\alpha + \beta = \bigcup_{\gamma < \beta} (\alpha + \gamma) \qquad \text{(limit case)}$$

  - associative: $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
  - not commutative: $1 + \omega = \omega \neq \omega + 1$
- **multiplication**

Ordinal Arithmetic

- **addition**
- **multiplication**

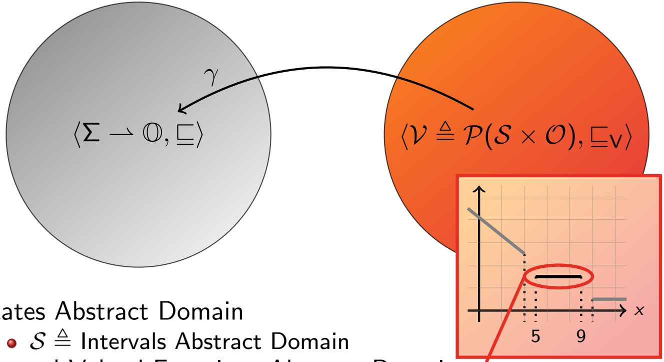
$$\alpha \cdot 0 = 0 \qquad \text{(zero case)}$$

$$\alpha \cdot \mathit{succ}(\beta) = (\alpha \cdot \beta) + \alpha \qquad \text{(successor case)}$$

$$\alpha \cdot \beta = \bigcup_{\gamma < \beta} (\alpha \cdot \gamma) \qquad \text{(limit case)}$$

- 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
  - $\mathcal{S} \triangleq$ Intervals Abstract Domain
- Natural-Valued Functions Abstract Domain
  - $\mathcal{F} \triangleq$ Affine Ranking Functions Abstract Domain
- Ordinal-Valued Functions Abstract Domain
  - $\mathcal{O} \triangleq \{\bot_O\} \cup \{\sum_i \omega^i \cdot f_i \mid f_i \in \mathcal{F} \setminus \{\bot_F, \top_F\}\} \cup \{\top_O\}$

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

Backward Assignments

- assignment transfer functions amount to weakest preconditions

Example

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

$$\Downarrow \quad \mathbf{x_1 := x_1 + x_2}$$

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

- the resulting covering is refined to obtain a partition

Backward Assignments

- assignment transfer functions amount to weakest preconditions

Example

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

$$\Downarrow \quad \mathbf{x_1 := x_1 + x_2}$$

$$[-\infty, +\infty] \mapsto o \triangleq \qquad + \quad 1$$

- the resulting covering is refined to obtain a partition

Backward Assignments

- assignment transfer functions amount to weakest preconditions

Example

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

$$\Downarrow \quad x_1 := x_1 + x_2$$

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot (x_1 + x_2 - x_2) + x_1 + x_2 + 1$$

- the resulting covering is refined to obtain a partition

Backward Assignments

- assignment transfer functions amount to weakest preconditions

Example

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

$$\Downarrow \quad x_1 := x_1 + x_2$$

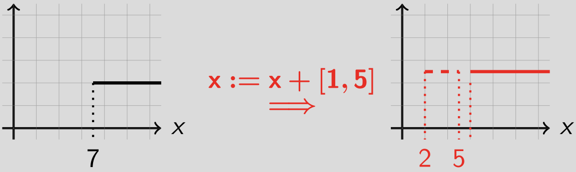
$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_1 + x_2 + 1$$

- 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

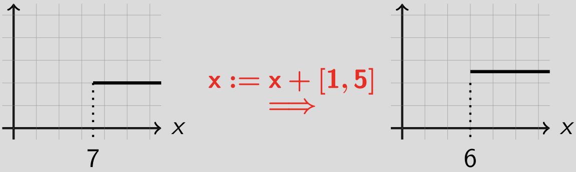
Example



Backward Assignments

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

Example



Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

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

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$[-\infty, +\infty] \mapsto o \triangleq \mathbf{?}$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

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

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$[-\infty, +\infty] \mapsto o \triangleq \quad + \mathbf{1}$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + \mathbf{x_2}$$

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$[-\infty, +\infty] \mapsto o \triangleq + \mathbf{x_2} + 1$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot \mathbf{x_1} + x_2$$

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$[-\infty, +\infty] \mapsto o \triangleq \omega^2 \cdot \mathbf{0} \; {}^{1} + \omega \cdot \mathbf{0} + x_2 + 1$$

$$\omega^k \cdot \omega = \omega^{k+1} \cdot 1 + \omega^k \times 0 = \omega^{k+1}$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

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

$$\Downarrow \quad \mathbf{x_1 := ?}$$

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

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

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

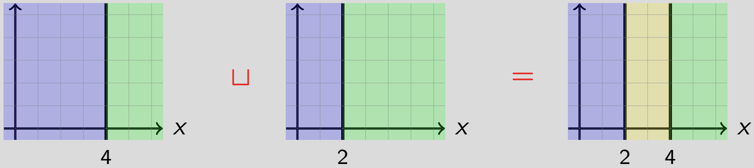
$$\Downarrow \quad \mathbf{x_1 := ?}$$

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

Join

- segmentation unification: $\sqcup$

Example

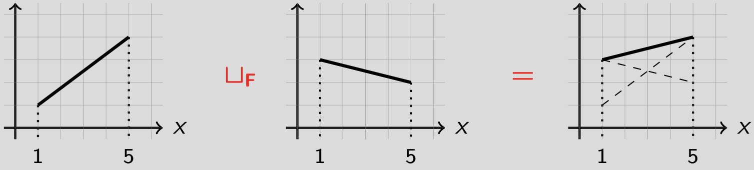


- join: $\sqcup_F$
- join: $\sqcup_O$

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 ω

Example

$$[-\infty, +\infty] \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3$$
$$[-\infty, +\infty] \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$$

$$[-\infty, +\infty] \mapsto o_1 \sqcup_O o_2 \triangleq ?$$

Join

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

Example

$$[-\infty, +\infty] \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3$$
$$[-\infty, +\infty] \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$$
$$[-\infty, +\infty] \mapsto o_1 \sqcup_O o_2 \triangleq + 4$$

Join

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

Example

$$[-\infty, +\infty] \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3$$

$$[-\infty, +\infty] \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$$

$$[-\infty, +\infty] \mapsto o_1 \sqcup_O o_2 \triangleq \quad 1 + \omega \cdot 0 + 4$$

$$\omega^k \cdot \omega = \omega^{k+1} \cdot 1 + \omega^k \times 0 = \omega^{k+1}$$

Join

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

Example

$$[-\infty, +\infty] \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3$$
$$[-\infty, +\infty] \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$$

$$[-\infty, +\infty] \mapsto o_1 \sqcup_O o_2 \triangleq \omega^2 \cdot x_1 \;^{1} + \omega \cdot 0 + 4$$

Join

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

Example

$$[-\infty, +\infty] \mapsto o_1 \triangleq \omega^2 \cdot \mathbf{x_1} + \omega \cdot x_2 + 3$$

$$[-\infty, +\infty] \mapsto o_2 \triangleq \omega^2 \cdot \mathbf{x_1} + \omega \cdot (-x_2) + 4$$

$$[-\infty, +\infty] \mapsto o_1 \sqcup_O o_2 \triangleq \omega^2 \cdot (\mathbf{x_1} + \mathbf{1}) + \omega \cdot 0 + 4$$

Join

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

Example

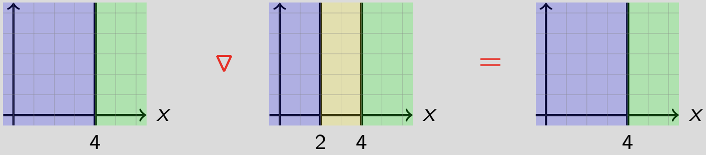
$$[-\infty, +\infty] \mapsto o_1 \triangleq \omega^2 \cdot x_1 + \omega \cdot x_2 + 3$$
$$[-\infty, +\infty] \mapsto o_2 \triangleq \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4$$

$$[-\infty, +\infty] \mapsto o_1 \sqcup_O o_2 \triangleq \omega^2 \cdot (x_1 + 1) + 4$$

Widening

- segmentation left-unification: $\triangledown$

Example

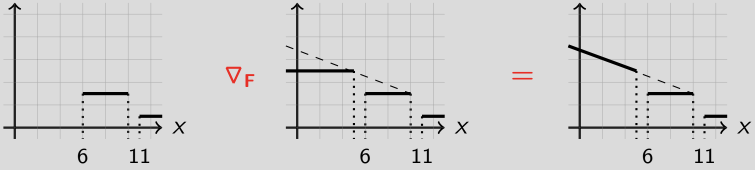


- widening: $\triangledown_F$
- widening: $\triangledown_O$

Widening

- segmentation left-unification: $\triangledown$
- widening: $\triangledown_F$

Example

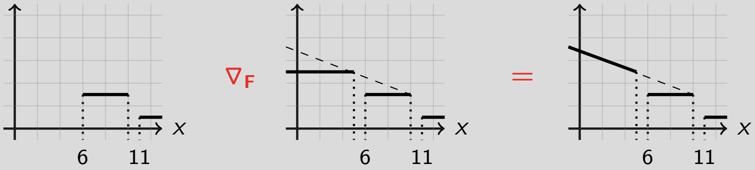


- unstable ranking functions yield $\top_F$
- widening: $\triangledown_O$

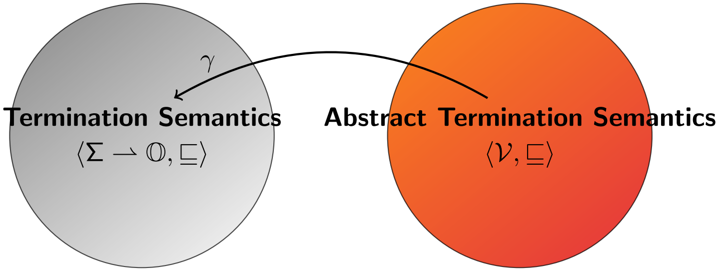
Widening

- segmentation left-unification: $\triangledown$
- widening: $\triangledown_{\mathrm{F}}$

Example



- widening: $\triangledown_{\mathrm{O}}$
  - $\triangledown_{\mathrm{F}}$ in ascending powers of ω
  - unstable ranking functions yield $\top_{\mathrm{O}}$



Theorem (Soundness)

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

Example

int : x_1, x_2

while [1]($x_1 > 0 \land x_2 > 0$) do

  if [2](?) then

    [3]$x_1 := x_1 - 1$

    [4]$x_2 :=$?

  else

    [5]$x_2 := x_2 - 1$

od[6]

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

Example

```
int : x1, x2
while 1(x1 ≠ 0 ∧ x2 > 0) do
    if 2(x1 > 0) then
        if 3( ? ) then
            4x1 := x1 − 1
            5x2 := ?
        else
            6x2 := x2 − 1
    else /* x1 < 0 */
        if 7( ? ) then
            8x1 := x1 + 1
        else
            9x2 := x2 − 1
            10x1 := ?
od11
```

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

Example

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

the coefficients and their **order** are **inferred by the analysis**

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

Non-Linear Ranking Functions

Example

int : N, x_1, x_2
$^{1}x_1 := N$
while $^{2}(x_1 \geq 0)$ do
  $^{3}x_2 := N$
  while $^{4}(x_2 \geq 0)$ do
    $^{5}x_2 := x_2 - 1$
  od^{6}
  $^{7}x_1 := x_1 - 1$
od^{8}

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

Non-Linear Ranking Functions

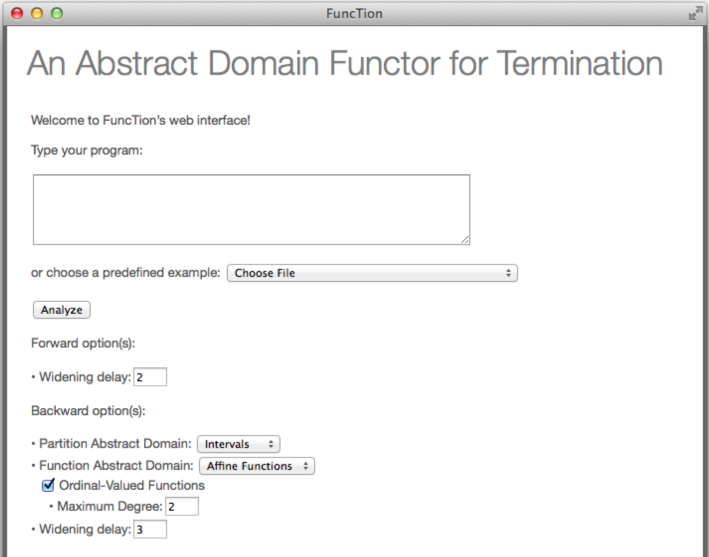
Example

int : N, x_1, x_2
${}^{1}x_1 := N$
while ${}^{2}(x_1 \geq 0)$ do
  ${}^{3}x_2 := N$
  while ${}^{4}(x_2 \geq 0)$ do
    ${}^{5}x_2 := x_2 - 1$
  od^{6}
  ${}^{7}x_1 := x_1 - 1$
od^{8}

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

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

To Infinity. . .

- 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

. . . and Beyond!

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

Conclusions — To Infinity. . .

- 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 — . . . and Beyond!

- **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
¹x := 0, b := true
while ²(b) do
    if ³(?) then
        ⁴x := x + 1
    else
        ⁵b := false
od⁶
```

Fair Termination

Example (Dijkstra's Random Number Generator)

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 [6]

$$f_2(x, b, z_1, z_2) = \begin{cases} 1 & \neg b \\ 4 & b \wedge z_1 > z_2 \\ 5z_2 + 9 & b \wedge z_1 \leq z_2 \end{cases}$$

Thank You!