The Abstract Domain of Piecewise-Defined Ranking Functions

Caterina Urban

Département d'Informatique
École Normale Supérieure

28th November 2013
East China Normal University
Shanghai, China

Introduction

- **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
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on ranking functions **over natural numbers**[3]
- instances based on ranking functions **over ordinal numbers**

---

[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)

Introduction

- **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
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on ranking functions **over natural numbers**[3]
- instances based on ranking functions **over ordinal numbers**

[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)

Introduction

- **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
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on ranking functions **over natural numbers**[3]
- instances based on ranking functions **over ordinal numbers**

[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)

Introduction

- **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
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on ranking functions **over natural numbers**[3]
- instances based on ranking functions **over ordinal numbers**

[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)

Introduction

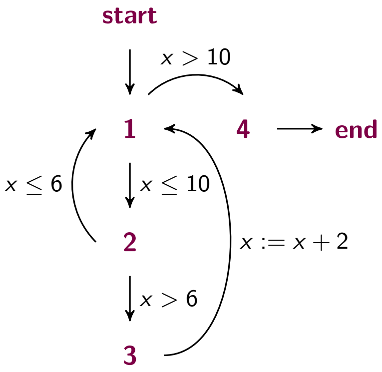
- **ranking functions**[1]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below
- **idea**: computation of ranking functions by abstract interpretation[2]
- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on ranking functions **over natural numbers**[3]
- instances based on ranking functions **over ordinal numbers**

[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)

Program Partial Correctness

Example

int : x
while [1]$(x \le 10)$ do
  if [2]$(x > 6)$ then
    [3]$x := x + 2$
  fi
od[4]



Floyd - *Assigning Meanings to Programs* (1967)

Program Partial Correctness

affix assertions to each program control point...

Example

int : x

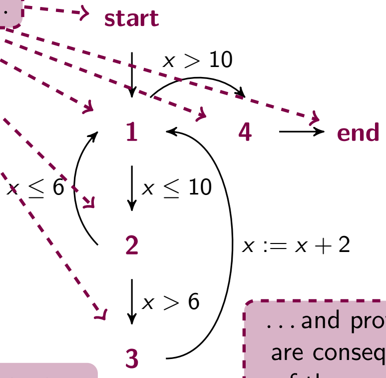
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



assertion = property of the program when control reaches the control point

...and prove they are consequences of the assertions of their predecessors

Floyd - *Assigning Meanings to Programs* (1967)

Program Partial Correctness

Example

int : x

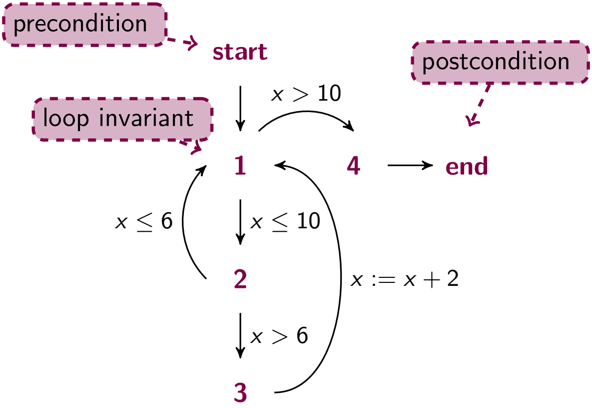
while ${}^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



assertions can be computed by abstract interpretation

Floyd - *Assigning Meanings to Programs* (1967)

Program Partial Correctness

Example

int : x

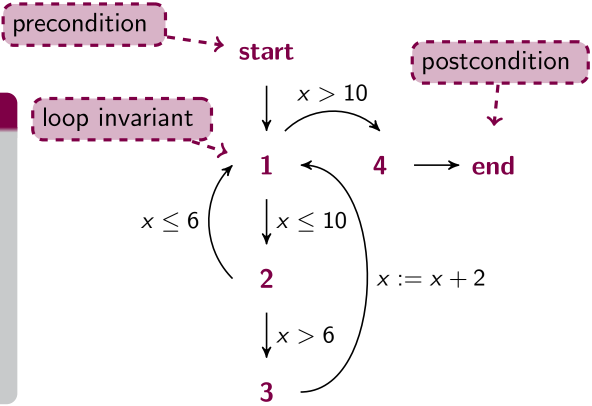
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



the program gives the correct result **if and when it terminates**

Floyd - *Assigning Meanings to Programs* (1967)

Program Total Correctness

Example

int : x

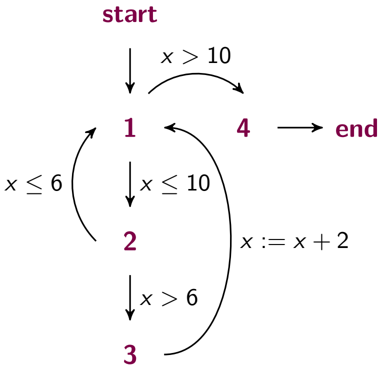
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



Total Correctness = Partial Correctness + **Termination**

Floyd - *Assigning Meanings to Programs* (1967)

Program Total Correctness

associate a function over a **well-ordered set** to each program control point...

Example

int : x

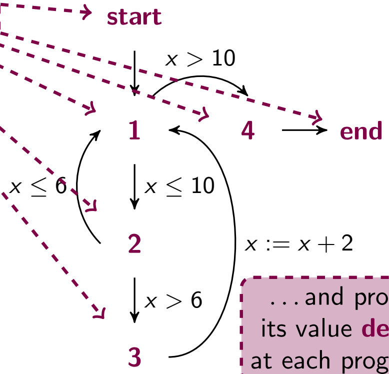
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



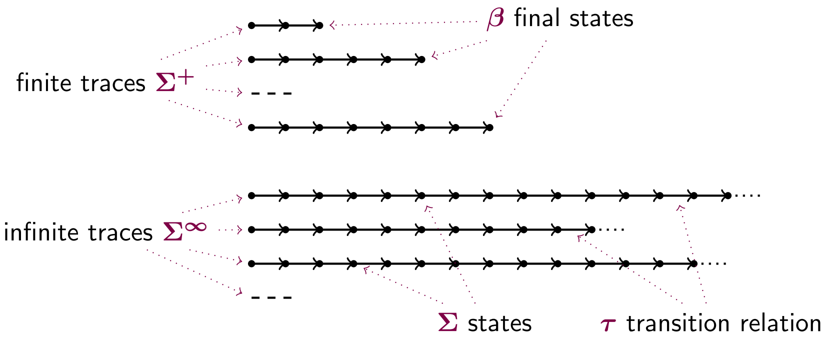
...and prove that its value **decreases** at each program step

ranking functions can be computed by abstract interpretation

Floyd - *Assigning Meanings to Programs* (1967)

Concrete Semantics

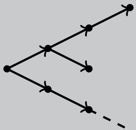
program $\mapsto$ **trace semantics**



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

idea = define a ranking function
that **counts the number of program steps**
from the end of the program

Example



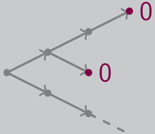
Theorem (Soundness and Completeness)

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



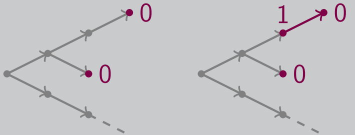
Theorem (Soundness and Completeness)

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



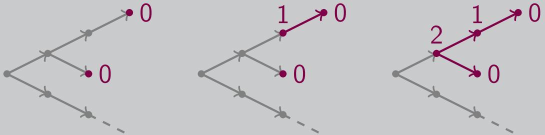
Theorem (Soundness and Completeness)

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



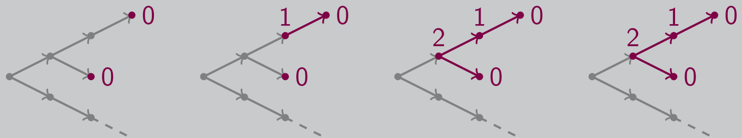
Theorem (Soundness and Completeness)

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



Theorem (Soundness and Completeness)

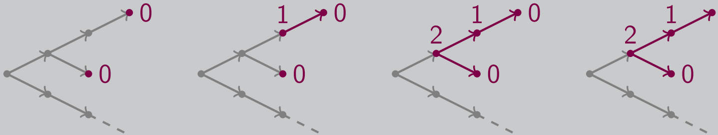
the termination semantics is **sound** *and* **complete**
to prove the termination of programs

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

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

the termination semantics
extracts the well-founded part
of the program transition relation

Example



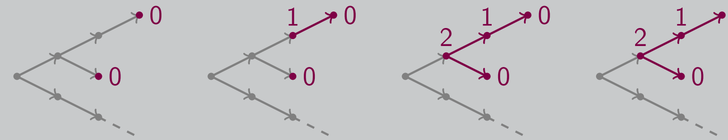
Theorem (Soundness and Completeness)

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



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

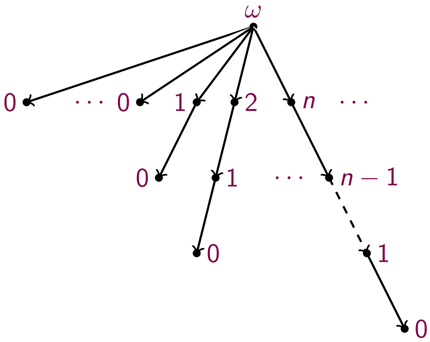
int : x

$x :=\ ?$

while $(x \geq 0)$ do

  $x := x - 1$

od



Example

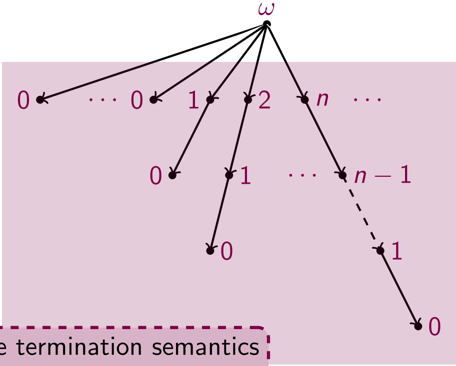
int : x

$x :=\ ?$

while ($x \geq 0$) do

  $x := x - 1$

od

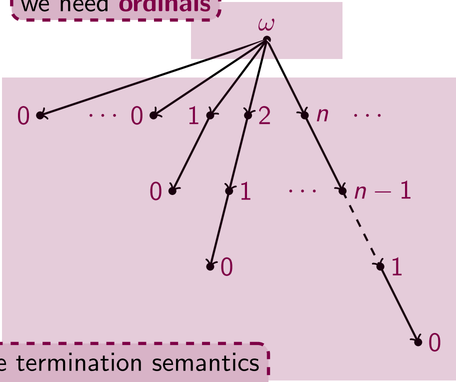


the termination semantics is **not computable**

Example

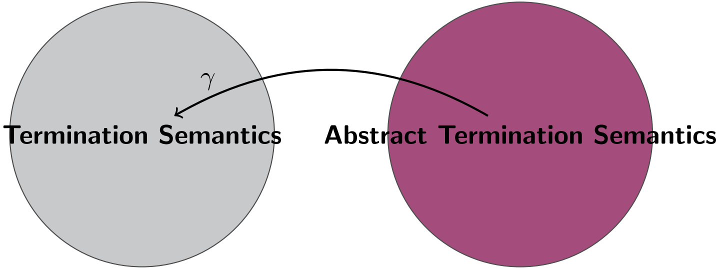
int : x
$x :=\ ?$
while $(x \geq 0)$ do
  $x := x - 1$
od

we need **ordinals**



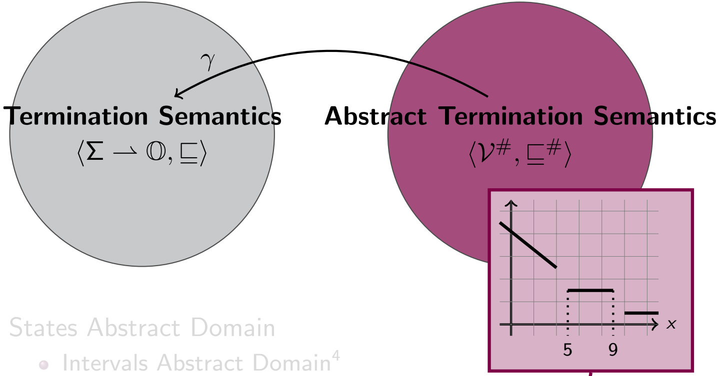
the termination semantics
is **not computable**

Piecewise-Defined Ranking Functions



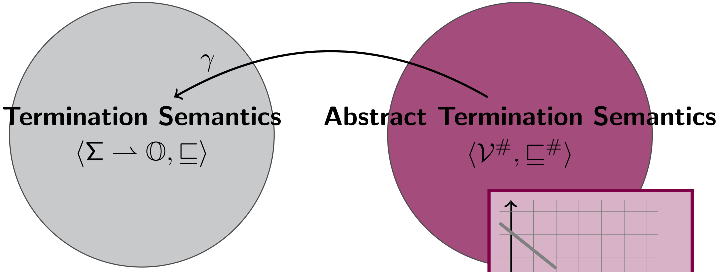
- States Abstract Domain S
  - Intervals Abstract Domain[4]
- Functions Abstract Domain F
  - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain $\mathsf{V}(\mathsf{S}, \mathsf{F})$

[4] Cousot&Cousot - *Static Determination of Dynamic Properties of Programs* (1976)

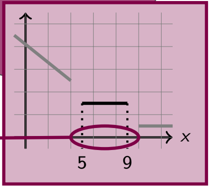


- States Abstract Domain — S
  - Intervals Abstract Domain[4]
- Functions Abstract Domain — F
  - Affine Ranking Functions
- **Piecewise-Defined Ranking Functions Abstract Domain** — $\mathsf{V}(\mathsf{S}, \mathsf{F})$

[4] Cousot&Cousot - *Static Determination of Dynamic Properties of Programs* (1976)



- States Abstract Domain — S
  - Intervals Abstract Domain[4]



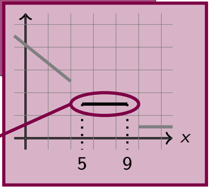
- Functions Abstract Domain — F
  - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, F)

[4] Cousot&Cousot - *Static Determination of Dynamic Properties of Programs* (1976)

γ

Termination Semantics
$\langle \Sigma \rightharpoonup \mathbb{O}, \sqsubseteq \rangle$

Abstract Termination Semantics
$\langle \mathcal{V}^{\#}, \sqsubseteq^{\#} \rangle$



- States Abstract Domain — S
  - Intervals Abstract Domain[4]
- Functions Abstract Domain — F
  - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain — $\mathsf{V}(\mathsf{S}, \mathsf{F})$

[4] Cousot&Cousot - *Static Determination of Dynamic Properties of Programs* (1976)

Why Piecewise-Defined Ranking Functions?

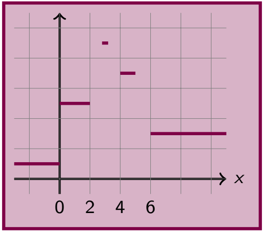
Example

int : x

while $^{1}(x \geq 0)$ do

$\quad ^{2}x := -2x + 10$

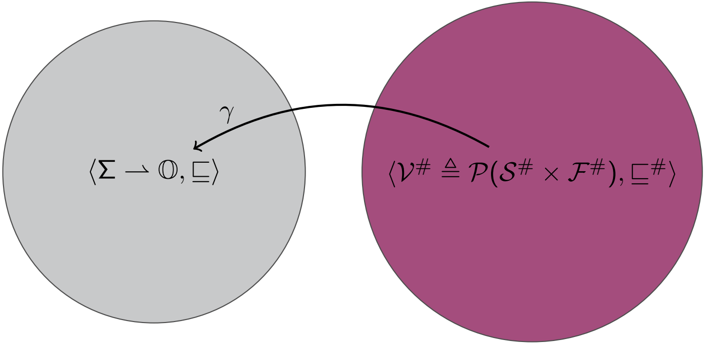
od^{3}



$$f(x) \triangleq \begin{cases} 1 & x < 0 \\ 5 & 0 \leq x \leq 2 \\ 9 & x = 3 \\ 7 & 4 \leq x \leq 5 \\ 3 & 5 < x \end{cases}$$

Natural-Valued Ranking Functions

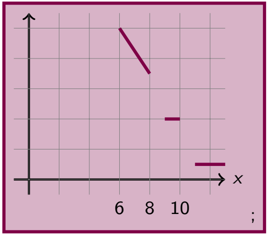
Natural-Valued Ranking Functions Domain



- $\mathcal{F}^\# \triangleq \{\bot_\mathsf{F}\} \cup \{f^\# \mid f^\# \in \mathbb{Z}^n \to \mathbb{N}\} \cup \{\top_\mathsf{F}\}$
  where $f^\# \equiv y = f(x_1, \ldots, x_n) = m_1 x_1 + \cdots + m_n x_n + q$

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

$$\langle \mathcal{V}^{\#} \triangleq \mathcal{P}(\mathcal{S}^{\#} \times \mathcal{F}^{\#}), \sqsubseteq^{\#} \rangle$$



;

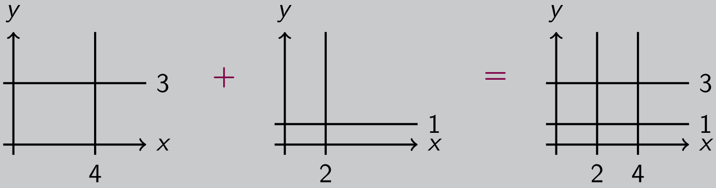
$$v^{\#} \triangleq \begin{cases} s_1^{\#} \mapsto f_1^{\#} \\ s_2^{\#} \mapsto f_2^{\#} \\ \dots \\ s_k^{\#} \mapsto f_k^{\#} \end{cases}$$

Example

$$v^{\#}(x) \triangleq \begin{cases} x \in [-\infty, 5] \mapsto \bot_F \\ x \in [6, 8] \mapsto -3x + 38 \\ x \in [9, 10] \mapsto 4 \\ x \in [11, +\infty] \mapsto 1 \end{cases}$$

- segmentation unification

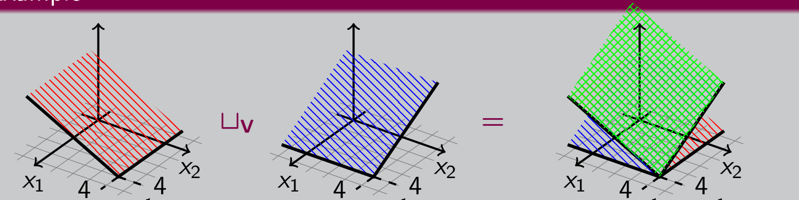
Example



- join: $\sqcup_{\mathsf{V}}$
- widening: $\triangledown_{\mathsf{V}}$
- backward assignments: $\mathrm{ASSIGN}_{\mathsf{V}}$

- segmentation unification
- join[5]: $\sqcup_V$

Example



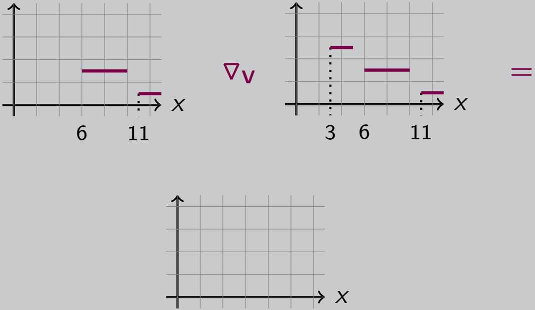
$f_1(x_1, x_2) = -\frac{1}{2}x_2 + 2$ $\quad f_2(x_1, x_2) = -\frac{1}{2}x_1 + 2$ $\quad f(x_1, x_2) = -\frac{1}{2}x_1 - \frac{1}{2}x_2 + 4$

- widening: ∇_V
- backward assignments: ASSIGN_V

[5] Cousot&Halbwachs - *Automatic Discovery of Linear Restraints Among Variables of a Program* (POPL 1978)

- segmentation unification
- join: $\sqcup_V$
- widening: ∇_V

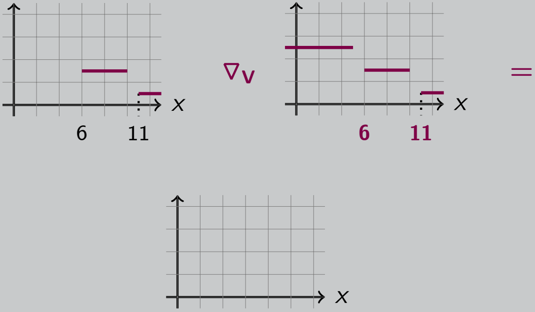
Example



- backward assignments: ASSIGN_V

- segmentation unification
- join: $\sqcup_V$
- widening: ∇_V

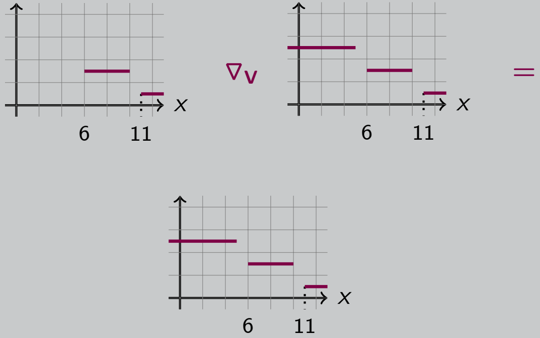
Example



- backward assignments: ASSIGN_V

- segmentation unification
- join: $\sqcup_V$
- widening: ∇_V

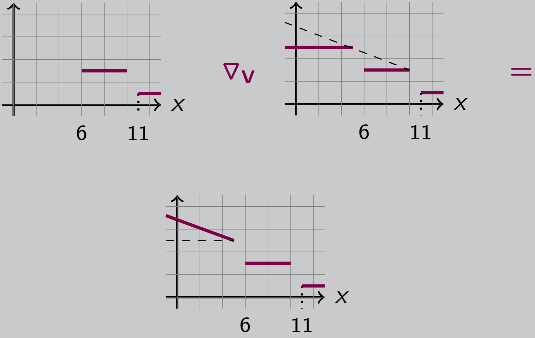
Example



- backward assignments: ASSIGN_V

- segmentation unification
- join: $\sqcup_V$
- widening: ∇_V

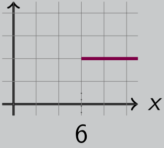
Example



- backward assignments: ASSIGN_V

- segmentation unification
- join: $\sqcup_V$
- widening: ∇_V
- backward assignments: ASSIGN_V

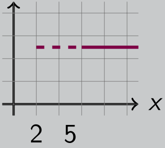
Example



$\langle x \mapsto (-\infty, 5], \bot_F \rangle$

$\langle x \mapsto [6, +\infty), y = 4 \rangle$

$\mathbf{x := x + [0, 4]}$
$\Longrightarrow$

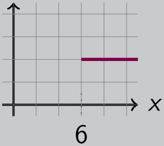


$\langle x \mapsto (-\infty, \mathbf{5}], \bot_F \rangle$

$\langle x \mapsto [\mathbf{2}, +\infty), y = 4 + \mathbf{1} \rangle$

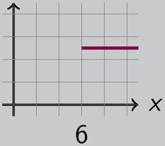
- segmentation unification
- join: $\sqcup_V$
- widening: $\triangledown_V$
- backward assignments: ASSIGN_V

Example

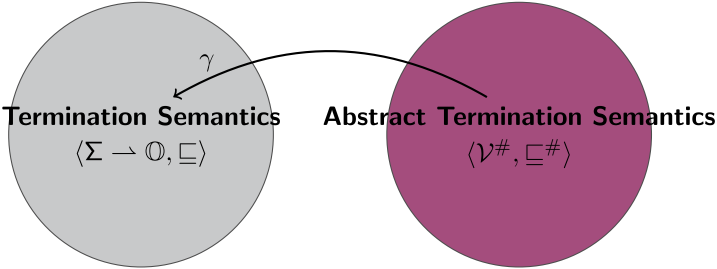


$\langle x \mapsto (-\infty, 5], \bot_F \rangle$
$\langle x \mapsto [6, +\infty), y = 4 \rangle$

$\mathbf{x := x + [0, 4]}$
$\Longrightarrow$



$\langle x \mapsto (-\infty, 1], \bot_F \rangle$
$\langle x \mapsto [2, 5], \bot_F \rangle$
$\langle x \mapsto [6, +\infty), y = 5 \rangle$



Theorem (Soundness)

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

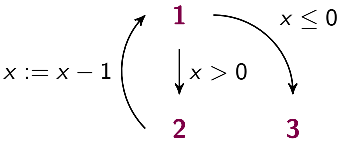
Example

int : x

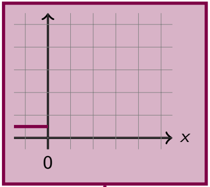
while [1]$(x > 0)$ do

  [2]$x := x - 1$

od[3]



we map each point to a function of x giving an upper bound on the steps before termination we take into account $x \leq 0$ and we have 1 step to termination



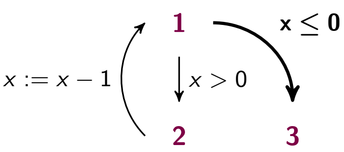
Example

int : x

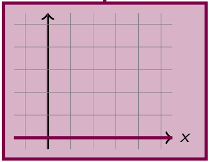
while [1]$(x > 0)$ do

[2]$x := x - 1$

od[3]



we start at the end with 0 steps before termination



we consider $x > 0$
and we do the join

Example

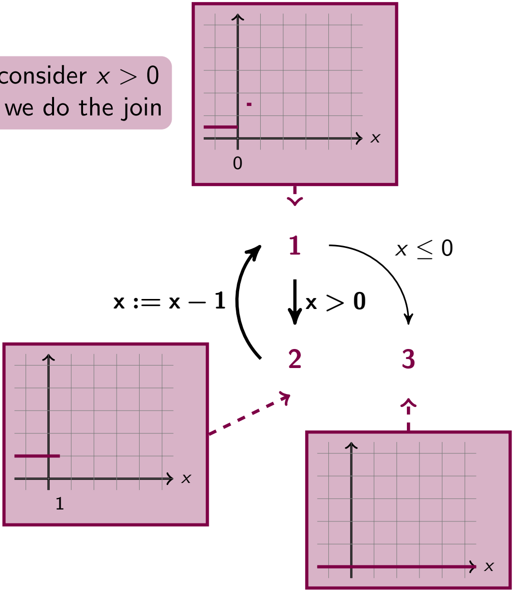
int : x

while ${}^{1}(x > 0)$ do

${}^{2}x := x - 1$

od^{3}

we consider the assignment
$x := x - 1$ and we are
at 2 steps to termination



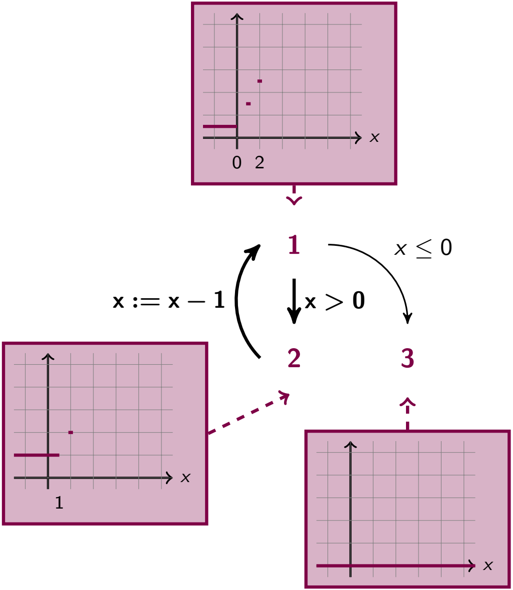
Example

int : x

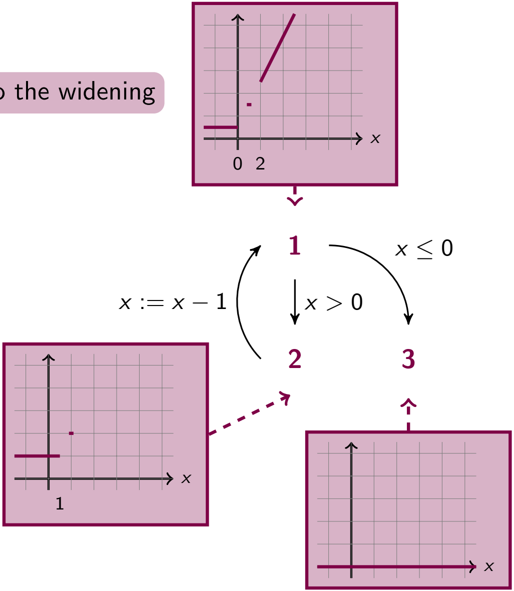
while $^{1}(x > 0)$ do

$^{2}x := x - 1$

od^{3}



we do the widening



Example

int : x

while [1]$(x > 0)$ do

[2]$x := x - 1$

od[3]

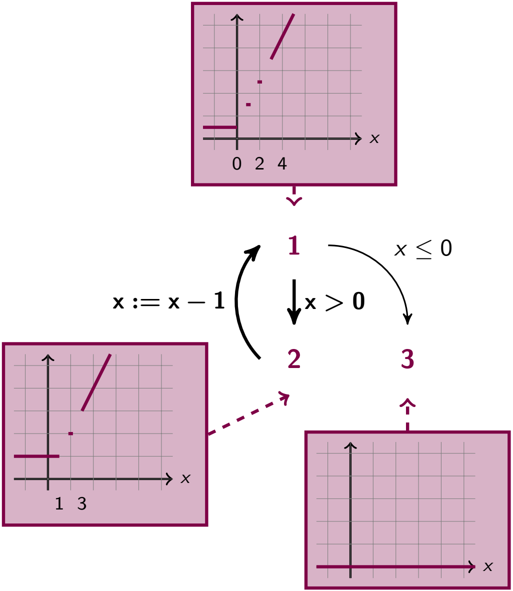
Example

int : x

while ${}^{1}(x > 0)$ do

${}^{2}x := x - 1$

od^{3}



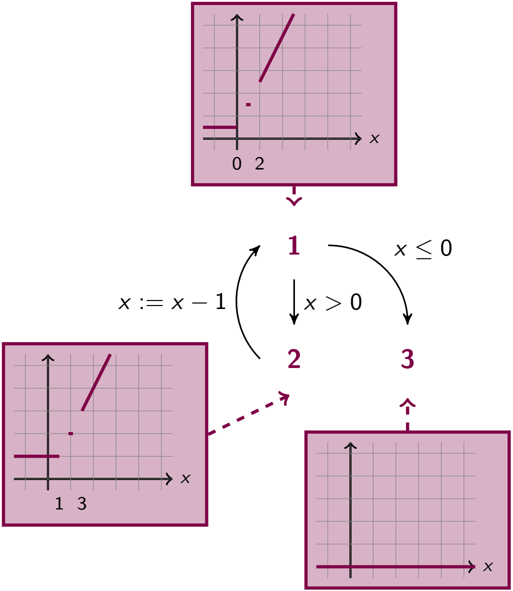
Example

int : x

while ${}^{1}(x > 0)$ do

${}^{2}x := x - 1$

od^{3}



the analysis gives true as sufficient precondition for termination

Example

int : x

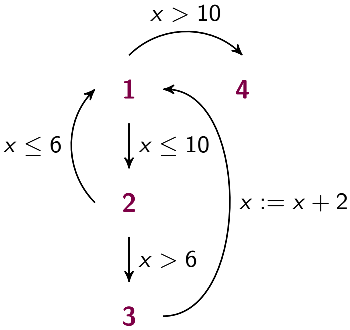
while ${}^{1}(x \leq 10)$ do

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

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

  fi

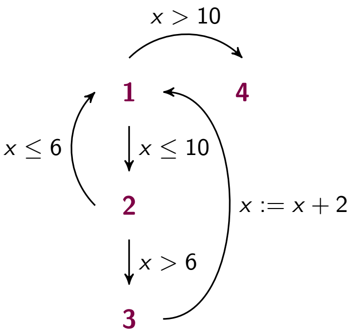
od^{4}



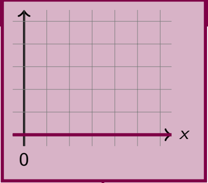
we map each point to a function of x giving an upper bound on the steps before termination

Example

int : x
while ${}^{1}(x \leq 10)$ do
  if ${}^{2}(x > 6)$ then
    ${}^{3}x := x + 2$
  fi
od^{4}



we start at the end with 0 steps before termination



Example

int : x

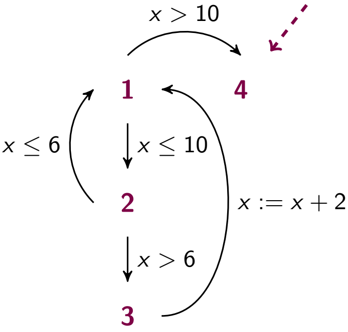
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



we take into account $x > 10$ and we have now 1 step to termination

Example

int : x

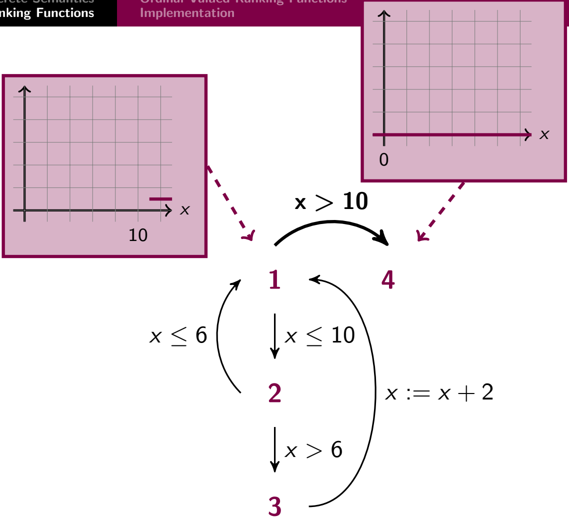
while ${}^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



Example

int : x

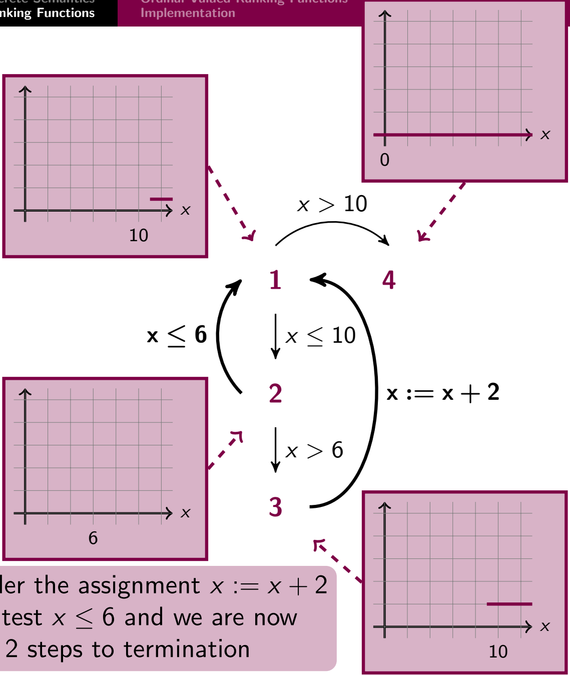
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



we consider the assignment $x := x + 2$
or the test $x \leq 6$ and we are now
at 2 steps to termination

Example

int : x

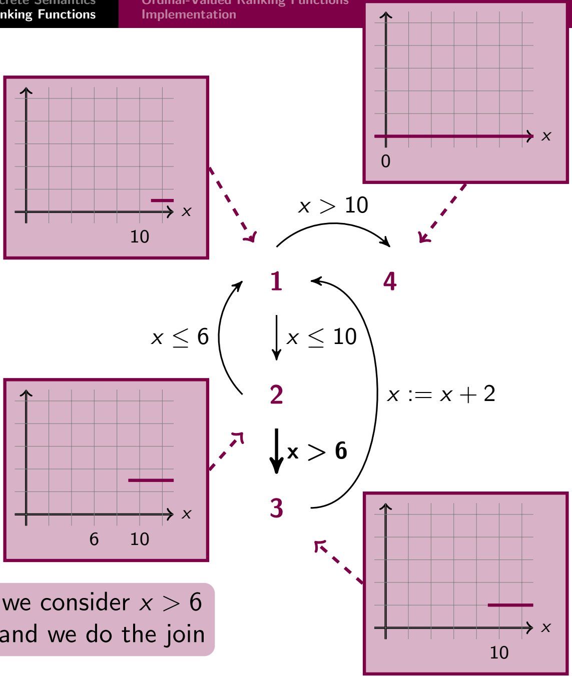
while ${}^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



we consider $x > 6$ and we do the join

we consider $x \leq 10$
and we do the join

Example

int : x

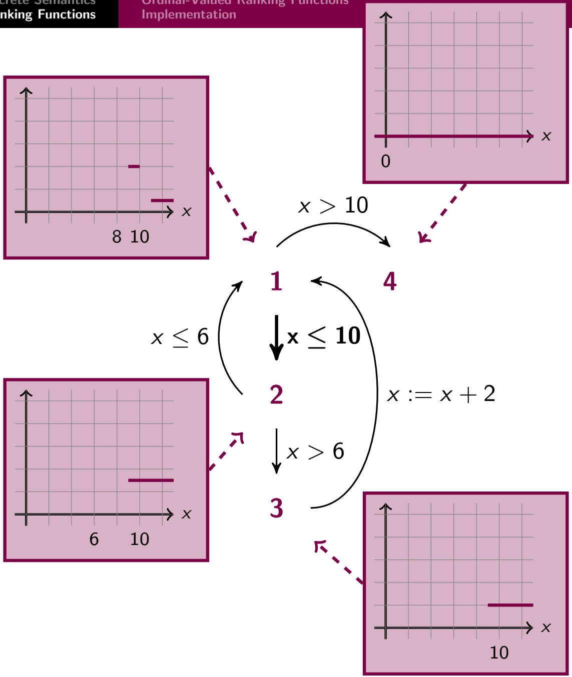
while ${}^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



Example

int : x

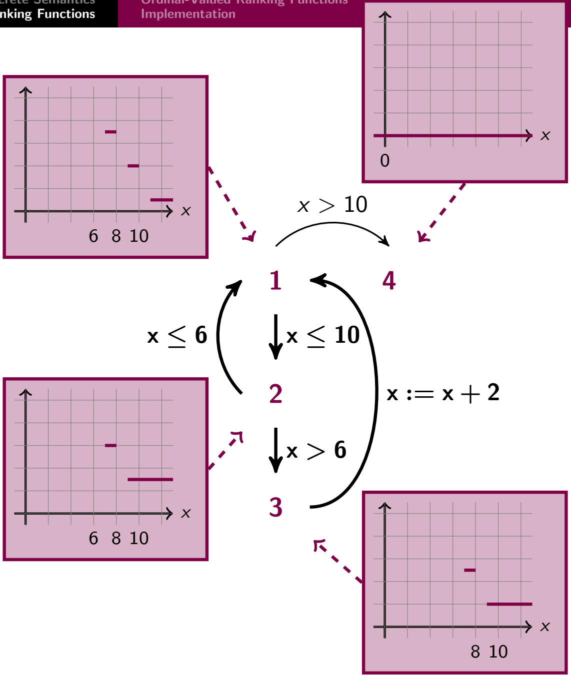
while ${}^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



we do the widening

Example

int : x

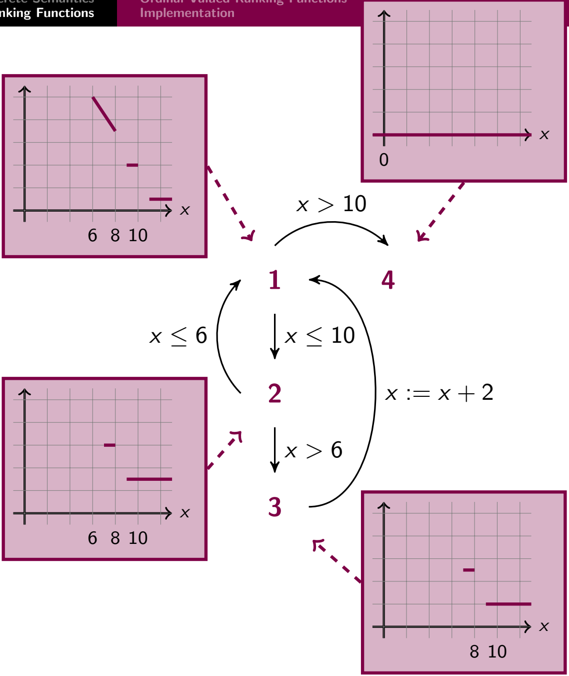
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



Example

int : x

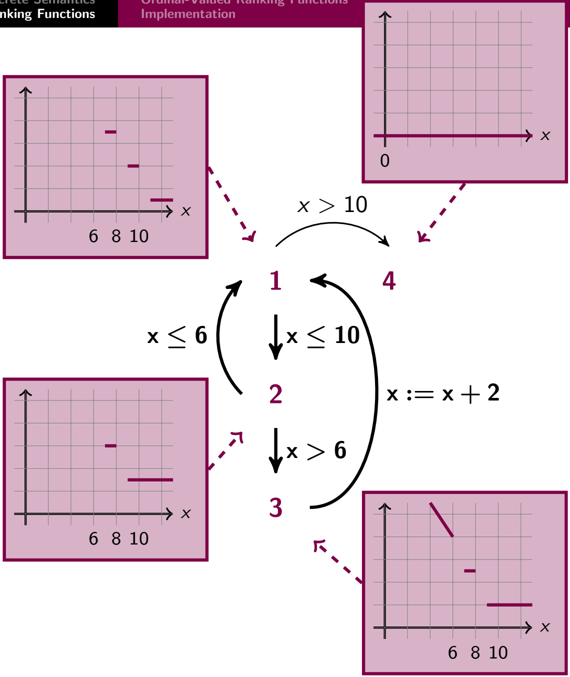
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



the analysis provides $x > 6$ as sufficient precondition for termination

Example

int : x

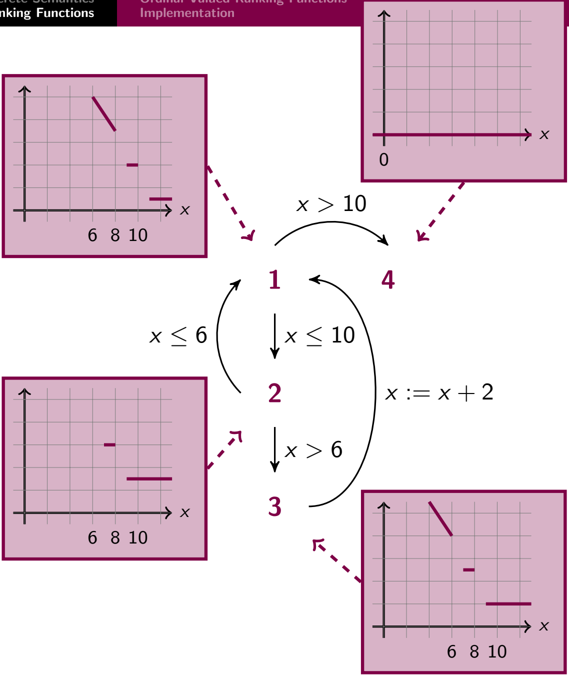
while ${}^{1}(x \leq 10)$ do

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

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

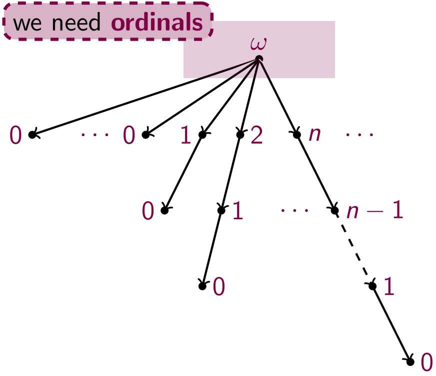
  fi

od^{4}



Example

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



Ordinal-Valued Ranking Functions

Ordinal Numbers

$$
\begin{aligned}
&0,\ 1,\ 2,\ \dots \\
&\omega,\ \omega+1,\ \omega+2,\ \dots \\
&\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \dots \\
&\vdots \\
&\omega^2,\ \dots \\
&\vdots \\
&\omega^\omega,\ \dots \\
&\vdots \\
&\epsilon_0,\ \dots \\
&\vdots
\end{aligned}
$$

Ordinal Numbers

$$\emptyset,\ 1,\ 2,\ \ldots$$

$$\omega,\ \omega+1,\ \omega+2,\ \ldots$$

$$\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots$$

$$\vdots$$

$$\omega^2,\ \ldots$$

$$\vdots$$

$$\omega^\omega,\ \ldots$$

$$\vdots$$

$$\epsilon_0,\ \ldots$$

$$\vdots$$

Ordinal Numbers

$$0,\ \{0\},\ 2,\ \ldots$$
$$\omega,\ \omega+1,\ \omega+2,\ \ldots$$
$$\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots$$
$$\vdots$$
$$\omega^2,\ \ldots$$
$$\vdots$$
$$\omega^\omega,\ \ldots$$
$$\vdots$$
$$\epsilon_0,\ \ldots$$
$$\vdots$$

Ordinal Numbers

$$
\begin{array}{l}
0,\ 1,\ \mathbf{\{0,1\}},\ \dots \\
\omega,\ \omega+1,\ \omega+2,\ \dots \\
\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \dots \\
\vdots \\
\omega^2,\ \dots \\
\vdots \\
\omega^\omega,\ \dots \\
\vdots \\
\epsilon_0,\ \dots \\
\vdots
\end{array}
$$

Ordinal Numbers

$$
\begin{array}{l}
0,\ 1,\ 2,\ \ldots \\
\boldsymbol{\{0, 1, 2, \ldots\}},\ \omega + 1,\ \omega + 2,\ \ldots \\
\omega \cdot 2,\ \omega \cdot 2 + 1,\ \omega \cdot 2 + 2,\ \ldots \\
\vdots \\
\omega^2,\ \ldots \\
\vdots \\
\omega^\omega,\ \ldots \\
\vdots \\
\epsilon_0,\ \ldots \\
\vdots
\end{array}
$$

Ordinal Numbers

$$
\begin{aligned}
&0,\ 1,\ 2,\ \ldots\\
&\omega,\ \boldsymbol{\{0, 1, 2, \ldots, \omega\}},\ \omega+2,\ \ldots\\
&\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots\\
&\vdots\\
&\omega^2,\ \ldots\\
&\vdots\\
&\omega^\omega,\ \ldots\\
&\vdots\\
&\epsilon_0,\ \ldots\\
&\vdots
\end{aligned}
$$

Ordinal Numbers

$$
\begin{array}{l}
0,\ 1,\ 2,\ \ldots \\
\omega,\ \omega+1,\ \boldsymbol{\{0,1,2,\ldots,\omega,\omega+1\}},\ \ldots \\
\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots \\
\vdots \\
\omega^2,\ \ldots \\
\vdots \\
\omega^\omega,\ \ldots \\
\vdots \\
\epsilon_0,\ \ldots \\
\vdots
\end{array}
$$

Ordinal Numbers

$$
\begin{array}{l}
0,\ 1,\ 2,\ \ldots \\
\omega,\ \omega+1,\ \omega+2,\ \ldots \\
\boldsymbol{\{0, 1, 2, \ldots, \omega, \omega+1, \omega+2, \ldots\}},\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots \\
\vdots \\
\omega^2,\ \ldots \\
\vdots \\
\omega^\omega,\ \ldots \\
\vdots \\
\epsilon_0,\ \ldots \\
\vdots
\end{array}
$$

Ordinal Numbers

$0,\ \mathbf{1},\ \mathbf{2},\ \dots$

$\omega,\ \boldsymbol{\omega+1},\ \boldsymbol{\omega+2},\ \dots$

$\omega\cdot 2,\ \boldsymbol{\omega\cdot 2+1},\ \boldsymbol{\omega\cdot 2+2},\ \dots$

$\vdots$

$\omega^2,\ \dots$

$\vdots$

$\omega^\omega,\ \dots$

$\vdots$

$\epsilon_0,\ \dots$

$\vdots$

successor ordinals
$\alpha + 1 \triangleq \alpha \cup \{\alpha\}$

Ordinal Numbers

$$0,\ 1,\ 2,\ \ldots$$
$$\boldsymbol{\omega},\ \omega + 1,\ \omega + 2,\ \ldots$$
$$\boldsymbol{\omega \cdot 2},\ \omega \cdot 2 + 1,\ \omega \cdot 2 + 2,\ \ldots$$
$$\vdots$$
$$\boldsymbol{\omega^2},\ \ldots$$
$$\vdots$$
$$\boldsymbol{\omega^\omega},\ \ldots$$
$$\vdots$$
$$\boldsymbol{\epsilon_0},\ \ldots$$
$$\vdots$$

limit ordinals

Ordinal Numbers

finite ordinals

$$0,\ 1,\ 2,\ \ldots$$
$$\omega,\ \omega+1,\ \omega+2,\ \ldots$$
$$\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots$$
$$\vdots$$
$$\omega^2,\ \ldots$$
$$\vdots$$
$$\omega^\omega,\ \ldots$$
$$\vdots$$
$$\epsilon_0,\ \ldots$$
$$\vdots$$

Ordinal Numbers

$$0,\ 1,\ 2,\ \ldots$$

$$\omega,\ \omega+1,\ \omega+2,\ \ldots$$

$$\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots$$

$$\vdots$$

$$\omega^2,\ \ldots$$

$$\vdots$$

$$\omega^\omega,\ \ldots$$

transfinite ordinals

$$\vdots$$

$$\epsilon_0,\ \ldots$$

$$\vdots$$

Ordinal Numbers

$$0,\ 1,\ 2,\ \ldots$$
$$\omega,\ \omega+1,\ \omega+2,\ \ldots$$
$$\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots$$
$$\vdots$$
$$\omega^2,\ \ldots$$
$$\vdots$$
$$\omega^\omega,\ \ldots$$
$$\vdots$$
$$\epsilon_0,\ \ldots$$
$$\vdots$$

Ordinal Arithmetic

- **addition**

$$\alpha + 0 = \alpha \qquad \text{(zero case)}$$
$$\alpha + (\beta + 1) = (\alpha + \beta) + 1 \qquad \text{(successor case)}$$
$$\alpha + \beta = \bigcup_{\gamma < \beta} (\alpha + \gamma) \qquad \text{(limit case)}$$

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

- **multiplication**
- **exponentiation**

Ordinal Arithmetic

- **addition**

$$\alpha + 0 = \alpha \qquad \text{(zero case)}$$
$$\alpha + (\beta + 1) = (\alpha + \beta) + 1 \qquad \text{(successor case)}$$
$$\alpha + \beta = \bigcup_{\gamma<\beta} (\alpha + \gamma) \qquad \text{(limit case)}$$

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

- **multiplication**
- **exponentiation**

Ordinal Arithmetic

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

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

$$\alpha \cdot (\beta + 1) = (\alpha \cdot \beta) + \alpha \qquad \text{(successor case)}$$

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

  - associative: $(\alpha \times \beta) \times \gamma = \alpha \times (\beta \times \gamma)$
  - left distributive: $\alpha \times (\beta + \gamma) = (\alpha \times \beta) + (\alpha \times \gamma)$
  - not commutative: $2 \times \omega = \omega \neq \omega \times 2$
  - not right distributive: $(\omega + 1) \times \omega = \omega \times \omega \neq \omega \times \omega + \omega$
- **exponentiation**

Ordinal Arithmetic

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

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

$$\alpha \cdot (\beta + 1) = (\alpha \cdot \beta) + \alpha \qquad \text{(successor case)}$$

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

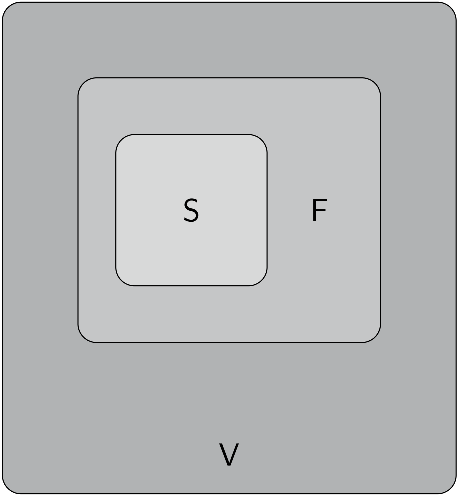
  - associative: $(\alpha \times \beta) \times \gamma = \alpha \times (\beta \times \gamma)$
  - left distributive: $\alpha \times (\beta + \gamma) = (\alpha \times \beta) + (\alpha \times \gamma)$
  - not commutative: $2 \times \omega = \omega \neq \omega \times 2$
  - not right distributive: $(\omega + 1) \times \omega = \omega \times \omega \neq \omega \times \omega + \omega$
- **exponentiation**

Ordinal Arithmetic

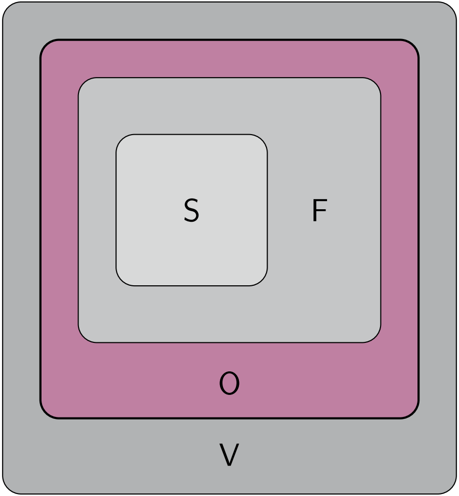
- **addition**
- **multiplication**
- **exponentiation**

$$
\begin{aligned}
\alpha^0 &= 1 && \text{(zero case)}\\
\alpha^{\beta+1} &= (\alpha^\beta) \cdot \alpha && \text{(successor case)}\\
\alpha^\beta &= \bigcup_{\gamma<\beta} (\alpha^\gamma) && \text{(limit case)}
\end{aligned}
$$

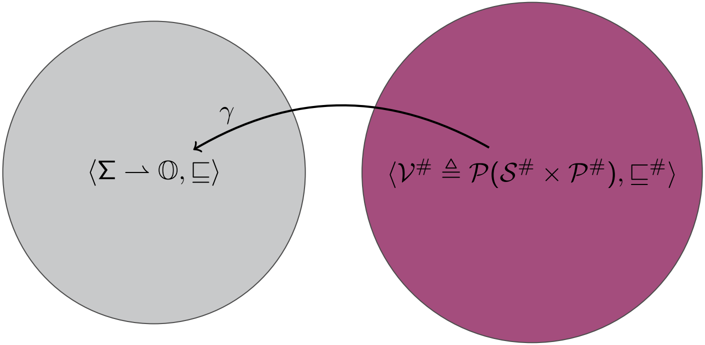
Ordinal-Valued Ranking Functions Domain



Ordinal-Valued Ranking Functions Domain

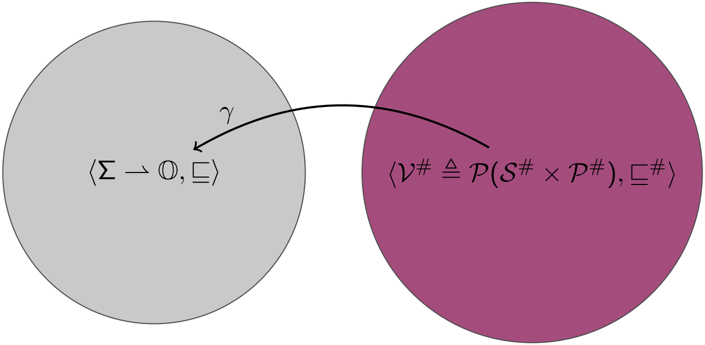


Ordinal-Valued Ranking Functions Domain



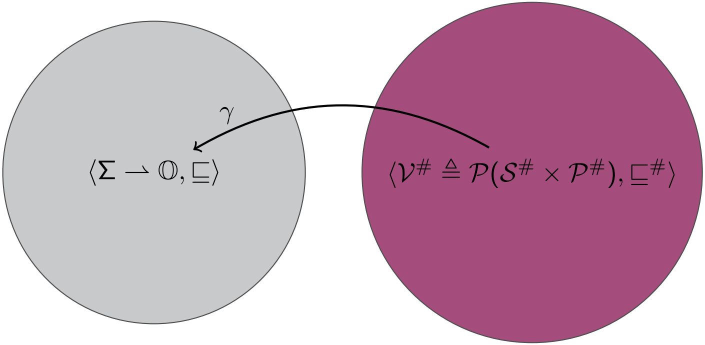
- $\mathcal{P}^\# \triangleq \{\bot_\mathsf{P}\} \cup \{p^\# \mid p^\# \in \mathbb{Z}^n \to \mathbb{O}\} \cup \{\top_\mathsf{P}\}$

Ordinal-Valued Ranking Functions Domain



- $\mathcal{P}^\# \triangleq \{\bot_\mathsf{P}\} \cup \{p^\# \mid p^\# \in \mathbb{Z}^n \to \mathbb{O}\} \cup \{\top_\mathsf{P}\}$
  $= \{\bot_\mathsf{P}\} \cup \{p^\# \mid p^\# = \sum_i \omega^i \cdot f_i^\#, f_i^\# \in \mathcal{F}^\#\} \cup \{\top_\mathsf{P}\}$

Ordinal-Valued Ranking Functions Domain



- $\mathcal{P}^\# \triangleq \{\bot_\mathsf{P}\} \cup \{p^\# \mid p^\# \in \mathbb{Z}^n \to \mathbb{O}\} \cup \{\top_\mathsf{P}\}$
  $= \{\bot_\mathsf{P}\} \cup \{p^\# \mid p^\# = \sum_i \omega^i \cdot f_i^\#, f_i^\# \in \mathcal{F}^\#\} \cup \{\top_\mathsf{P}\}$
  where $f^\# \equiv y = f(x_1, \ldots, x_n) = m_1 x_1 + \cdots + m_n x_n + q$

$$\langle \mathcal{V}^\# \triangleq \mathcal{P}(\mathcal{S}^\# \times \mathcal{P}^\#), \sqsubseteq^\# \rangle$$

$$v^\# \triangleq \begin{cases} s_1^\# \mapsto p_1^\# \\ s_2^\# \mapsto p_2^\# \\ \dots \\ s_k^\# \mapsto p_k^\# \end{cases}$$

$$p^\# \triangleq \omega^k \cdot f_k^\# + \ldots + \omega^2 \cdot f_2^\# + \omega \cdot f_1^\# + f_0^\#$$

Example

$$v^\#(x) \triangleq \begin{cases} x \in [-\infty, -1], y \in [-\infty, 0] \mapsto 1 \\ x \in [-\infty, -1], y \in [1, +\infty] \mapsto \omega^2 + \omega \cdot (y-1) - 4x + 9y - 2 \\ x \in [0, 0], y \in [-\infty, +\infty] \mapsto 1 \\ x \in [1, +\infty], y \in [-\infty, 0] \mapsto 1 \\ x \in [1, +\infty), y \in [1, +\infty) \mapsto \omega \cdot (x-1) + 9x + 4y - 7 \end{cases}$$

Lexicographic Ranking Functions

$$\omega^k \cdot \underbrace{f_k^\#}_{\in\mathbb{N}} + \ldots + \omega^2 \cdot \underbrace{f_2^\#}_{\in\mathbb{N}} + \omega \cdot \underbrace{f_1^\#}_{\in\mathbb{N}} + \underbrace{f_0^\#}_{\in\mathbb{N}} \in \mathbb{O}$$

Lexicographic Ranking Functions

$$\omega^k \cdot \underbrace{f_k^\#}_{\in\mathbb{N}} + \ldots + \omega^2 \cdot \underbrace{f_2^\#}_{\in\mathbb{N}} + \omega \cdot \underbrace{f_1^\#}_{\in\mathbb{N}} + \underbrace{f_0^\#}_{\in\mathbb{N}} \in \mathbb{O}$$

$$\Updownarrow$$

$$(f_k^\#, \ldots, f_2^\#, f_1^\#, f_0^\#) \in \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{k}$$

- join: $\sqcup_V$

Example

$$
\begin{array}{ccccccccc}
v_1^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & x_1 & + & x_2 \\
v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & (x_1 - 1) & - & x_2 \\
\hline
v_1^\# \sqcup_V v_2^\# & \triangleq & ? & \mapsto & ? & & & &
\end{array}
$$

- backward assignments: ASSIGN_V

- join: $\sqcup_V$

Example

$$
\begin{array}{ccccccc}
v_1^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega \cdot x_1 + x_2 \\
v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega \cdot (x_1 - 1) - x_2 \\
\hline
v_1^\# \sqcup_V v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & ?
\end{array}
$$

- backward assignments: ASSIGN_V

- join: $\sqcup_V$

Example

$$\begin{array}{ccccccccc}
v_1^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & x_1 & + & \mathbf{x_2} \\
v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & (x_1 - 1) & - & \mathbf{x_2} \\
\hline
v_1^\# \sqcup_V v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & & & \mathbf{1} & + & \mathbf{0}
\end{array}$$

- backward assignments: ASSIGN_V

- join: $\sqcup_V$

Example

$$
\begin{array}{ccccccccc}
v_1^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & \mathbf{x_1} & & + & x_2 \\
v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & \mathbf{(x_1 - 1)} & & - & x_2 \\
\hline
v_1^\# \sqcup_V v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & \mathbf{x_1} & \mathbf{1} & + & 0
\end{array}
$$

- backward assignments: ASSIGN_V

- join: $\sqcup_V$

Example

$$
\begin{array}{ccccccccc}
v_1^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & \mathbf{x_1} & + & x_2 \\
v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & (\mathbf{x_1 - 1}) & - & x_2 \\
\hline
v_1^\# \sqcup_V v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & (\mathbf{x_1 + 1}) & + & 0
\end{array}
$$

- backward assignments: ASSIGN_V

- join: $\sqcup_{\mathrm{V}}$

Example

$$
\begin{array}{ccccccccc}
v_1^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & x_1 & + & x_2 \\
v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & (x_1 - 1) & - & x_2 \\
\hline
v_1^\# \sqcup_{\mathrm{V}} v_2^\# & \triangleq & [-\infty, +\infty] & \mapsto & \omega & \cdot & (x_1 + 1) & &
\end{array}
$$

- backward assignments: $\mathrm{ASSIGN}_{\mathrm{V}}$

- join: $\sqcup_V$
- backward assignments: ASSIGN_V

Example

$$p^\# \triangleq \omega \cdot x_1 + x_2$$

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

$$p^\# \triangleq ?$$

- join: $\sqcup_V$
- backward assignments: ASSIGN_V

Example

$$p^\# \triangleq \omega \cdot x_1 + x_2$$

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

$$p^\# \triangleq \qquad + \ \mathbf{1}$$

- join: $\sqcup_V$
- backward assignments: ASSIGN_V

Example

$$p^\# \triangleq \omega \cdot x_1 + \mathbf{x_2}$$

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

$$p^\# \triangleq + \ \mathbf{x_2} + 1$$

- join: $\sqcup_V$
- backward assignments: ASSIGN_V

Example

$$p^\# \triangleq \omega \cdot \mathbf{x_1} + x_2$$

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

$$p^\# \triangleq \mathbf{1} + \omega \cdot \mathbf{0} + x_2 + 1$$

- join: $\sqcup_V$
- backward assignments: ASSIGN_V

Example

$$p^\# \triangleq \omega \cdot x_1 + x_2$$

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

$$p^\# \triangleq \omega^2 \cdot 1 + \omega \cdot 0 + x_2 + 1$$

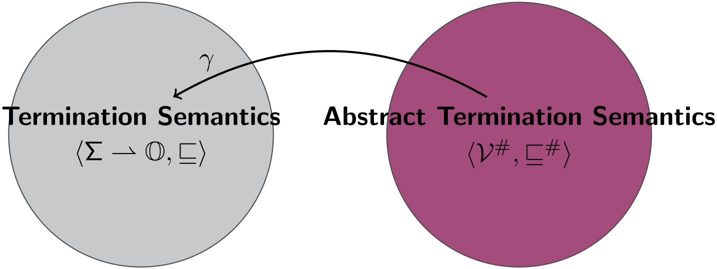
- join: $\sqcup_V$
- backward assignments: ASSIGN_V

Example

$$p^{\#} \triangleq \omega \cdot x_1 + x_2$$

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

$$p^{\#} \triangleq \omega^2 + x_2 + 1$$



Theorem (Soundness)

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

Example

int : x_1, x_2

while $^{1}(x_1 > 0 \wedge x_2 > 0)$ do

  if $^{2}(?)$ then

    $^{3}x_1 := x_1 - 1$

    $^{4}x_2 := \ ?$

  else

    $^{5}x_2 := x_2 - 1$

  fi

od^{6}

$$f(x_1, x_2) = \begin{cases} x \in [-\infty, 0], y \in [-\infty, 0] \mapsto 1 \\ x \in [-\infty, 0], y \in [1, +\infty] \mapsto 1 \\ x \in [1, +\infty], y \in [-\infty, 0] \mapsto 1 \\ x \in [1, +\infty), y \in [1, +\infty) \mapsto \omega \cdot (x_1 - 1) + 7x_1 + 3x_2 - 5 \end{cases}$$

Non-Linear Computational Complexity

Example

```
int : x1, x2
1 x1 := N
while 2 (x1 ≥ 0) do
  3 x2 := N
  while 4 (x2 ≥ 0) do
    5 x2 := x2 − 1
  od
  6 x1 := x1 − 1
od 7
```

the loop terminates in a **finite number of iterations**

$$f(x_1, x_2) = \begin{cases} x \in [-\infty, 0], y \in [-\infty, +\infty] \mapsto 1 \\ x \in [1, +\infty), y \in [-\infty, +\infty) \mapsto \omega + 2 \end{cases}$$

`http://www.di.ens.fr/~urban/FuncTion.html`

- written in OCaml
- implemented on top of Apron[5]

- forward reachability analysis to improve precision

Example

int : x_1, x_2
$^1x_2 := 1$
while $^2(x_1 < 10)$ do
  $^3x_1 := x_1 + x_2$
od^4

[5] `http://apron.cri.ensmp.fr/library/`

`http://www.di.ens.fr/~urban/FuncTion.html`

- written in OCaml
- implemented on top of Apron[5]

- forward reachability analysis to improve precision

Example

int : x_1, x_2

$^{1}x_2 := 1$

while $^{2}(x_1 < 10)$ do

  $^{3}x_1 := x_1 + x_2$

od^{4}

[5] `http://apron.cri.ensmp.fr/library/`