The Abstract Domain of Piecewise-Defined Ranking Functions

Caterina Urban

Département d'Informatique
École Normale Supérieure

AVDCPS 2013
Changsha, 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**[4]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (to appear)

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**[4]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (to appear)

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**[4]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (to appear)

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**[4]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (to appear)

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**[4]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (to appear)

Program Partial Correctness

Example

int : x

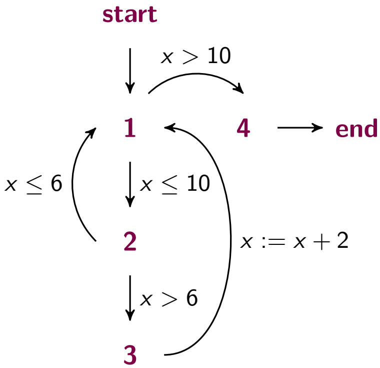
while $^{1}(x \leq 10)$ do

  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

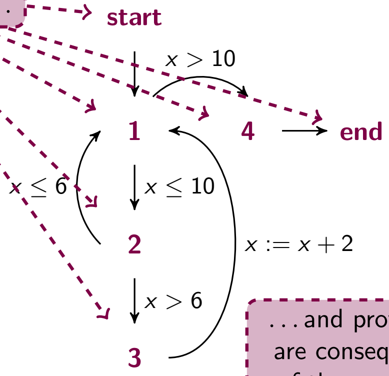
while $^{\mathbf{1}}(x \leq 10)$ do

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

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

  fi

od$^{\mathbf{4}}$



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

---

Floyd - *Assigning Meanings to Programs* (1967)

Program Partial Correctness

Example

int : x

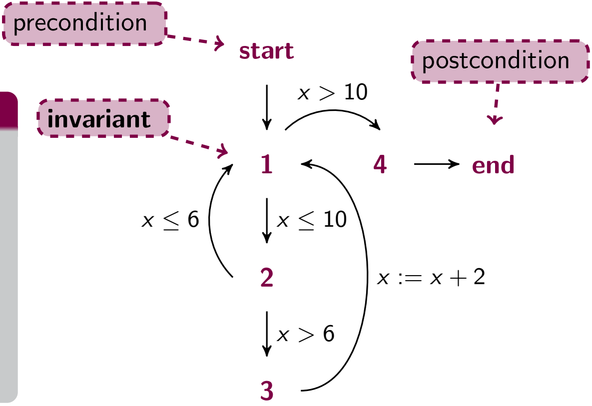
while [1]$(x \leq 10)$ do

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

  [3]$x := x + 2$

 fi

od[4]



Floyd - *Assigning Meanings to Programs* (1967)

Program Partial Correctness

Example

int : x

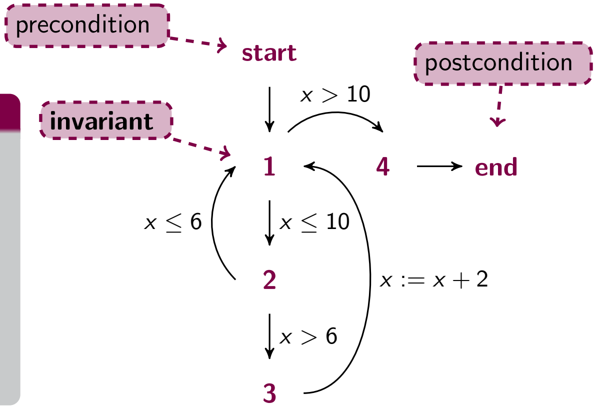
while ${}^{1}(x \leq 10)$ do

  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

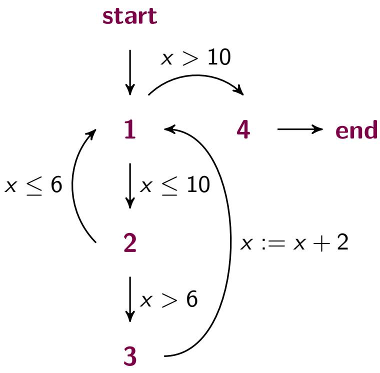
while [1]$(x \leq 10)$ do

  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

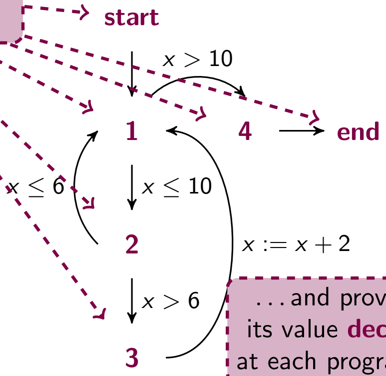
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



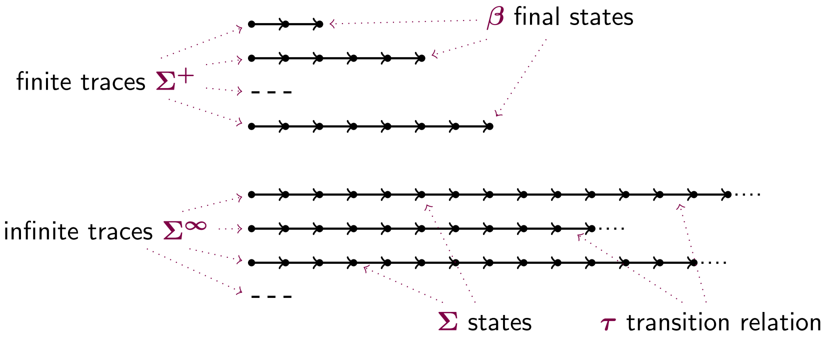
...and prove that its value **decreases** at each program step

Total Correctness = Partial Correctness + **Termination**

Floyd - *Assigning Meanings to Programs* (1967)

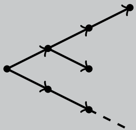
Concrete Semantics

program $\mapsto$ **trace semantics**



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

Example



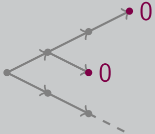
Theorem (Soundness and Completeness)

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



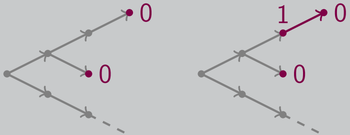
Theorem (Soundness and Completeness)

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



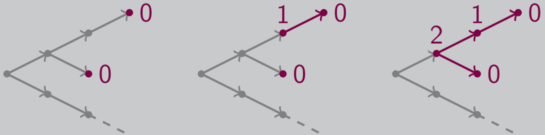
Theorem (Soundness and Completeness)

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



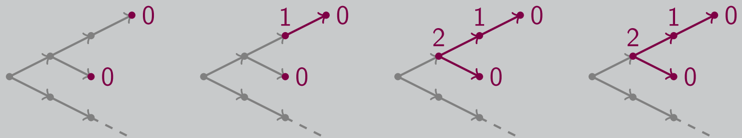
Theorem (Soundness and Completeness)

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**

Example



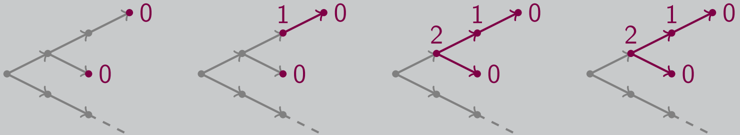
Theorem (Soundness and Completeness)

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

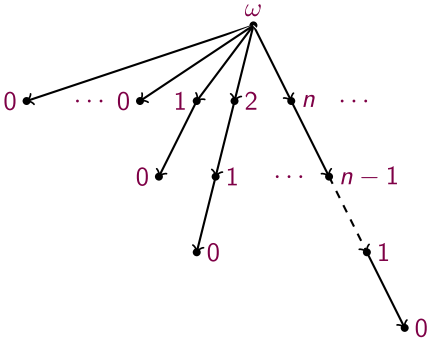
int : x

$x :=\ ?$

while $(x \geq 0)$ do

$\quad x := x - 1$

od



Example

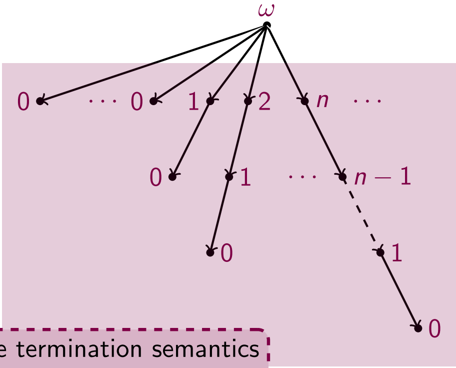
int : x

$x :=\ ?$

while ($x \geq 0$) do

$x := x - 1$

od

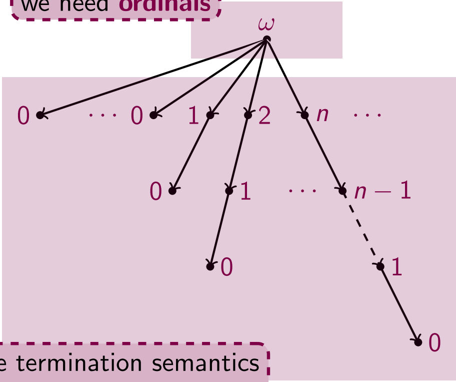


the termination semantics is **not computable**

Example

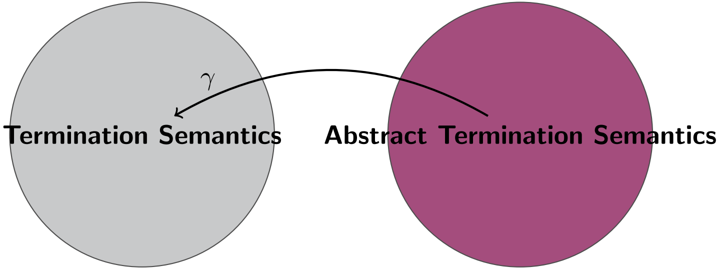
int : x
$x :=\ ?$
while $(x \geq 0)$ do
  $x := x - 1$
od

we need **ordinals**



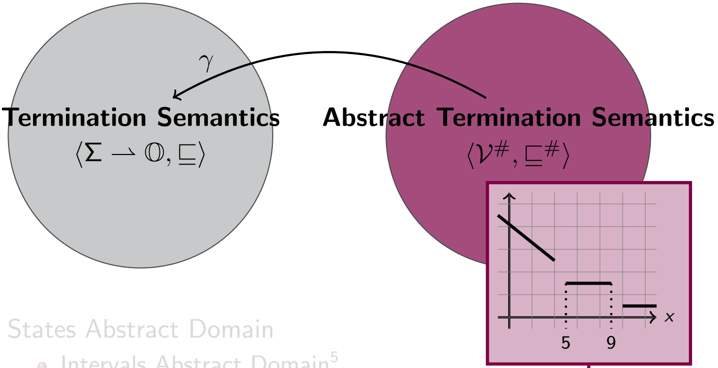
the termination semantics is **not computable**

Piecewise-Defined Ranking Functions



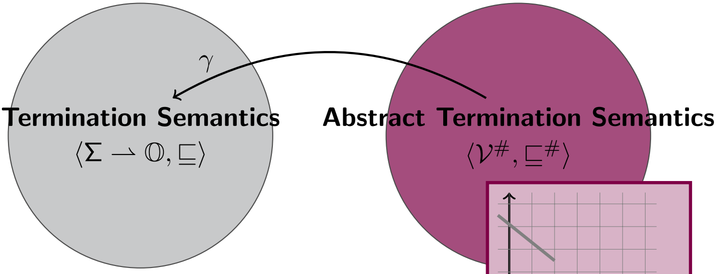
- States Abstract Domain $\mathcal{S}$
    - Intervals Abstract Domain[5]
- Functions Abstract Domain $\mathcal{F}$
    - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain $\mathcal{V}(\mathcal{S}, \mathcal{F})$

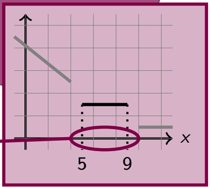
[5] Cousot&Cousot - *Static Determination of Dynamic Properties of Programs* (1976)



- States Abstract Domain — S
  - Intervals Abstract Domain[5]
- Functions Abstract Domain — F
  - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain — $\mathsf{V}(\mathsf{S}, \mathsf{F})$

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





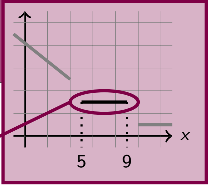
- States Abstract Domain — S
    - Intervals Abstract Domain[5]
- Functions Abstract Domain — F
    - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, F)

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

γ

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

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

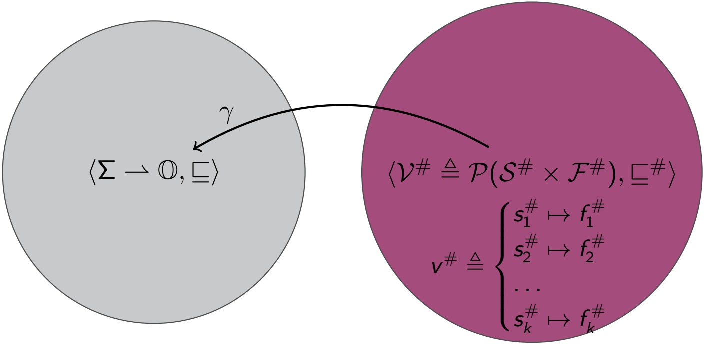


- States Abstract Domain — S
  - Intervals Abstract Domain[5]
- Functions Abstract Domain — F
  - Affine Ranking Functions
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, F)

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

Natural-Valued Ranking Functions

Natural-Valued Ranking Functions Domain

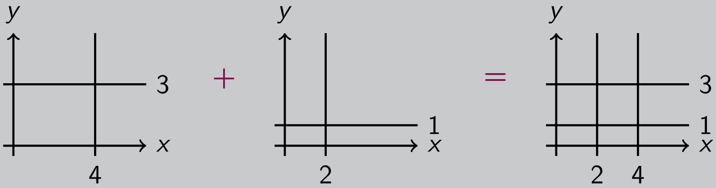


- $\mathcal{F}^\# \triangleq \{\bot_F\} \cup \{f^\# \mid f^\# \in \mathbb{Z}^n \to \mathbb{N}\} \cup \{\top_F\}$
  where $f^\# \equiv y = f(x_1, \dots, x_n) = m_1 x_1 + \cdots + m_n x_n + q$

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

- segmentation unification

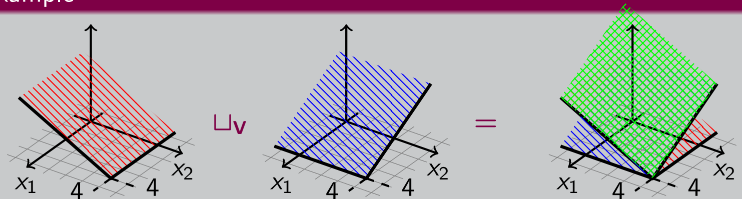
Example



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

- segmentation unification
- join[6]: $\sqcup_V$

Example



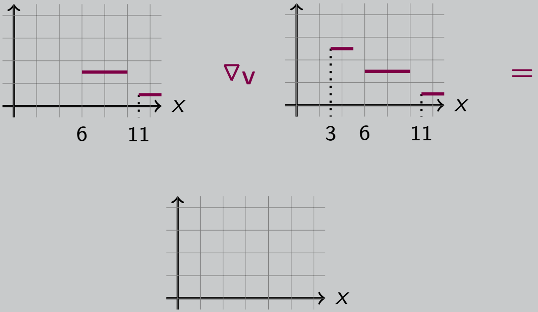
$f_1(x_1, x_2) = -\frac{1}{2}x_2 + 2$ $\quad$ $f_2(x_1, x_2) = -\frac{1}{2}x_1 + 2$ $\quad$ $f(x_1, x_2) = -\frac{1}{2}x_1 - \frac{1}{2}x_2 + 4$

- widening: ∇_V
- backward assignments: ASSIGN_V

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

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

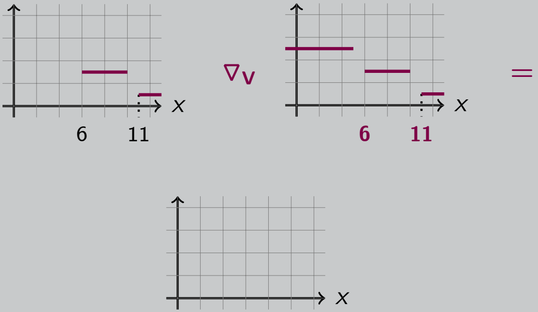
Example



- backward assignments: ASSIGN_V

- segmentation unification
- join: $\sqcup_V$
- widening: $\triangledown_V$

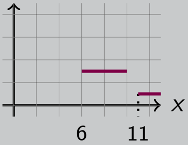
Example



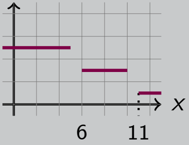
- backward assignments: ASSIGN_V

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

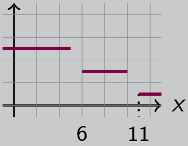
Example



∇_V



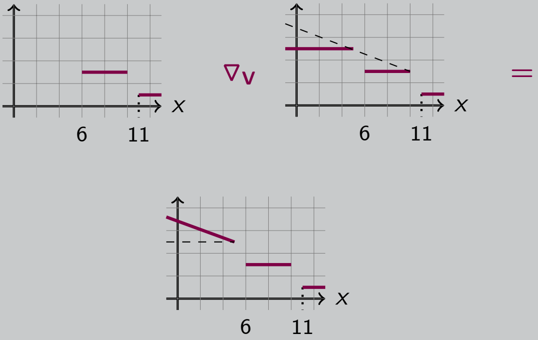
$=$



- backward assignments: ASSIGN_V

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

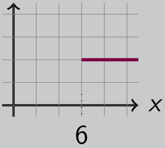
Example



- backward assignments: ASSIGN_V

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

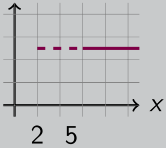
Example



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

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

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

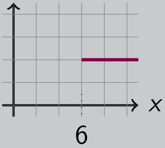


$\langle x \mapsto (-\infty, \mathbf{5}], \bot_F \rangle$

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

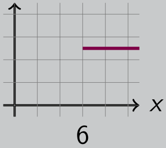
- segmentation unification
- join: $\sqcup_V$
- widening: ∇_V
- backward assignments: ASSIGN_V

Example



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

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



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

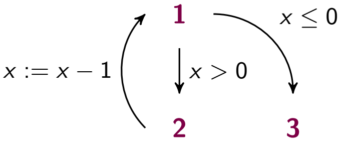
Example

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

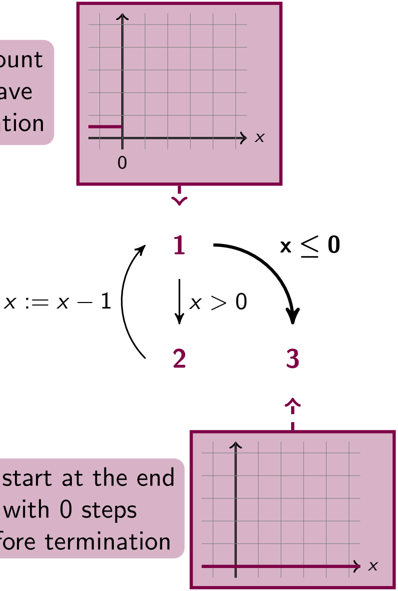
Example

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

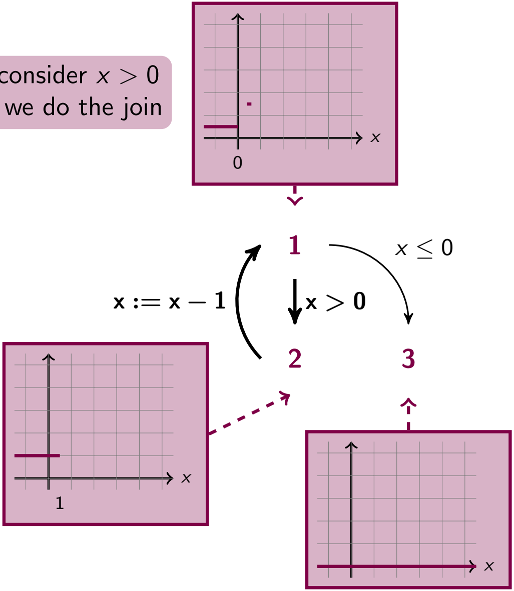
int : x

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



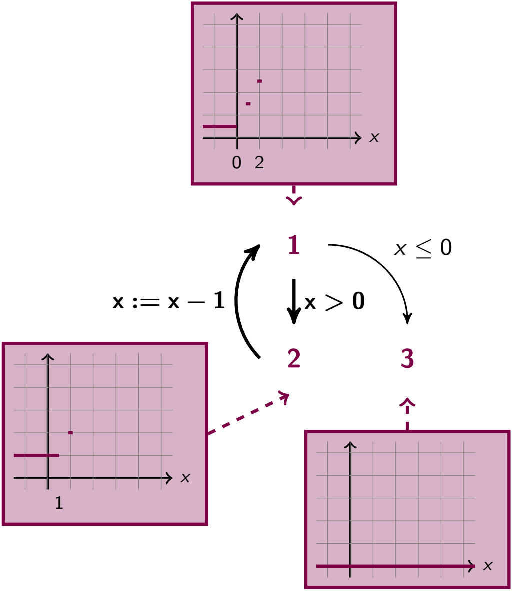
Example

int : x

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

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

od^{3}



we do the widening

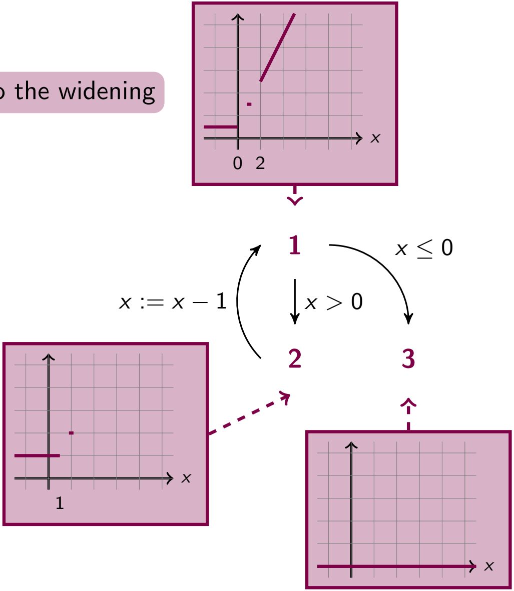
Example

int : x

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

[2]$x := x - 1$

od[3]



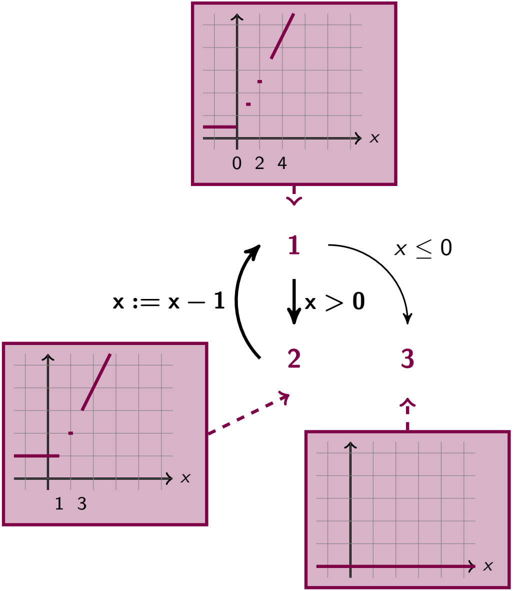
Example

int : x

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

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

od^{3}



Example

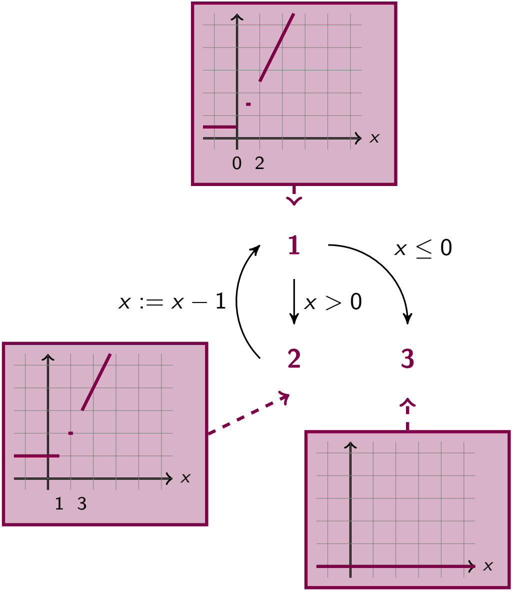
int : x

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

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

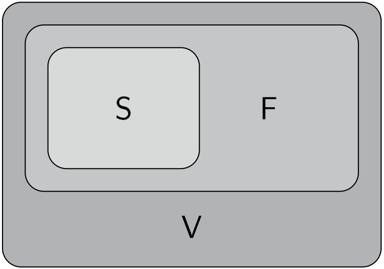
od^{3}

the analysis gives **true** as **sufficient precondition** for termination



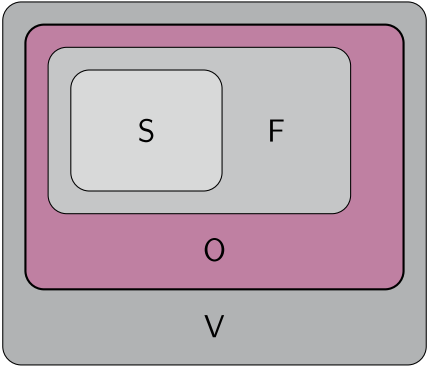
Ordinal-Valued Ranking Functions

Ordinal-Valued Ranking Functions Domain



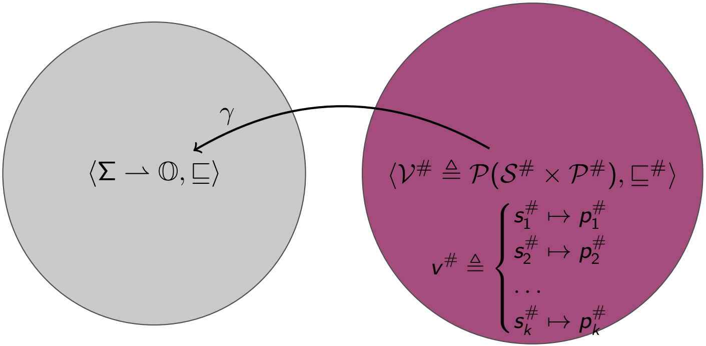
Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (to appear)

Ordinal-Valued Ranking Functions Domain



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

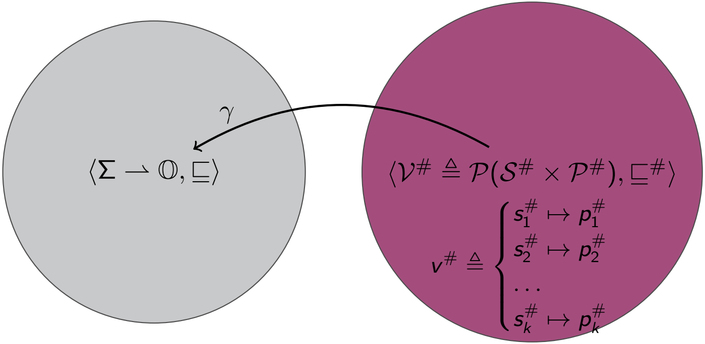
Ordinal-Valued Ranking Functions Domain



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

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

Ordinal-Valued Ranking Functions Domain



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

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

Ordinal-Valued Ranking Functions Domain

$$\langle \Sigma \rightharpoonup \mathbb{O}, \sqsubseteq \rangle \xleftarrow{\gamma} \langle \mathcal{V}^\# \triangleq \mathcal{P}(\mathcal{S}^\# \times \mathcal{P}^\#), \sqsubseteq^\# \rangle$$

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

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

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

- 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

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

- 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

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

- 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: $\sqcup_V$

Example

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

- backward assignments: ASSIGN_V

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

Example

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

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

$$p^\# \triangleq \mathbf{?}$$

- 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

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

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

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

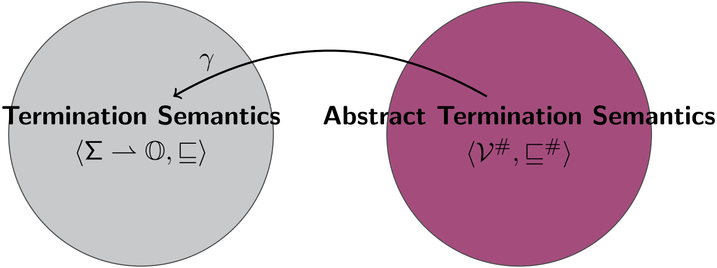
- join: $\sqcup_V$
- backward assignments: ASSIGN_V

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

~~Simple Loops~~

Example

int : x_1, x_2

while $^1(x_1 \geq 0 \land x_2 \geq 0)$ do

  if $^2(?)$ then

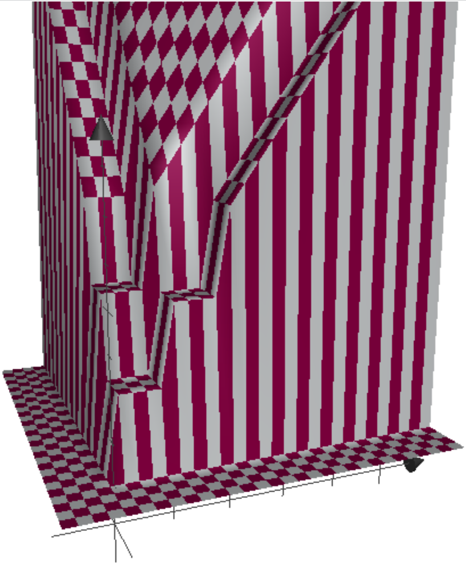
    $^3x_1 := x_1 - 1$

  else

    $^4x_2 := x_2 - 1$

  fi

od^5



~~Lexicographic Ranking Functions~~

Example

int : x_1, x_2

while $^1(x_1 \geq 0 \wedge x_2 \geq 0)$ do

  if $^2(?)$ then

    $^3x_1 := x_1 - 1$

    $^4x_2 := \ ?$

  else

    $^5x_2 := x_2 - 1$

  fi

od^6

$$f(x_1, x_2) = \begin{cases} 1 & x_1 \leq 0 \vee x_2 \leq 0 \\ 3x_2 + 2 & x_1 = 1 \\ \omega + 3x_2 + 9 & x_1 = 2 \\ \omega \cdot (x_1 - 1) + 7x_1 + 3x_2 - 5 & \text{otherwise} \end{cases}$$

Sufficient Preconditions for Termination

Example

int : x

while [1]$(x < 10)$ do

[2]$x := 2 * x$

od[3]

$$f(x) = \begin{cases} 3 & 5 \leq x \leq 9 \\ 1 & 10 \leq x \end{cases}$$

$$f(x) = \begin{cases} 9 & x = 1 \\ 7 & x = 2 \\ 5 & 3 \leq x \leq 4 \\ 3 & 5 \leq x \leq 9 \\ 1 & 10 \leq x \end{cases}$$

Sufficient Preconditions for Termination

$$f(x) = \begin{cases} 3 & 5 \leq x \leq 9 \\ 1 & 10 \leq x \end{cases}$$

Example

int : x

while [1]$(x < 10)$ do

[2]$x := 2 * x$

od[3]

$$f(x) = \begin{cases} 9 & x = 1 \\ 7 & x = 2 \\ 5 & 3 \leq x \leq 4 \\ 3 & 5 \leq x \leq 9 \\ 1 & 10 \leq x \end{cases}$$

Non-Linear Computational Complexity

Example

int : 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
  $^6 x_1 := x_1 - 1$
od^7

$$f(x_1, x_2) = \begin{cases} 1 & x_1 \leq 0 \\ \omega + 2 & \text{otherwise} \end{cases}$$

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

- written in OCaml
- implemented on top of Apron[6]

- 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}

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

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

- written in OCaml
- implemented on top of Apron[6]

- 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}

[6] http://apron.cri.ensmp.fr/library/

Experiments

Benchmarks: 38 programs

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

Results: proved 30 out of 38 programs

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

Conclusions

- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - backward invariance analysis
- instances based on **natural-valued functions**
  - analysis not limited to simple loops
  - sufficient conditions for termination
- instances based on **ordinal-valued functions**
  - ordinals remove the burden of finding lexicographic orders
  - analysis not limited to programs with linear computational complexity

Future Work

- **more abstract domains**
- other liveness properties
- complexity analysis

Conclusions

- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - backward invariance analysis
- instances based on **natural-valued functions**
  - analysis not limited to simple loops
  - sufficient conditions for termination
- instances based on **ordinal-valued functions**
  - ordinals remove the burden of finding lexicographic orders
  - analysis not limited to programs with linear computational complexity

Future Work

- **more abstract domains**
- other liveness properties
- complexity analysis

Thank You!

Questions?

Example

int : x

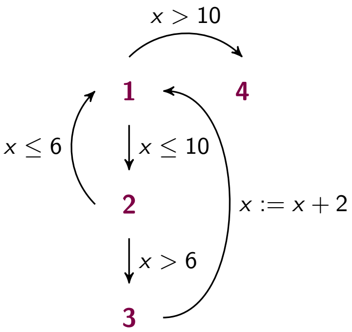
while ${}^{1}(x \leq 10)$ do

  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

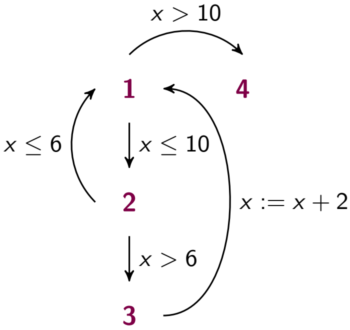
while ${}^{\mathbf{1}}(x \leq 10)$ do

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

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

  fi

od${}^{\mathbf{4}}$



Example

int : x

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

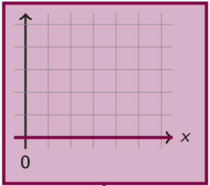
  if $^{2}(x > 6)$ then

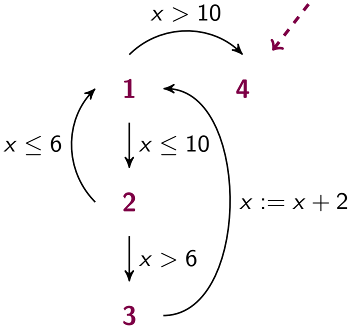
    $^{3}x := x + 2$

  fi

od^{4}

we start at the end with 0 steps before termination





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

Example

int : x

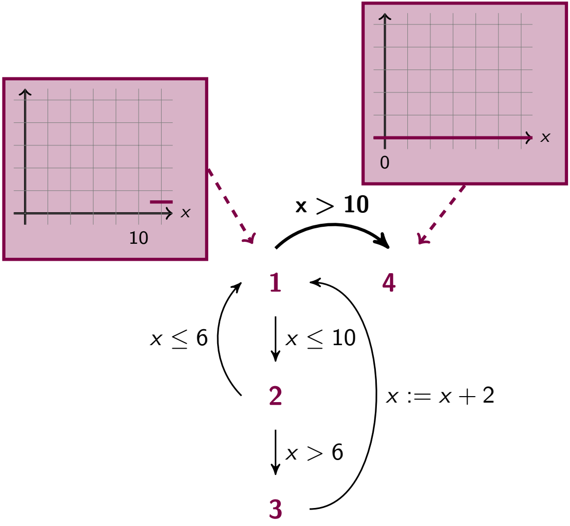
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



Example

int : x

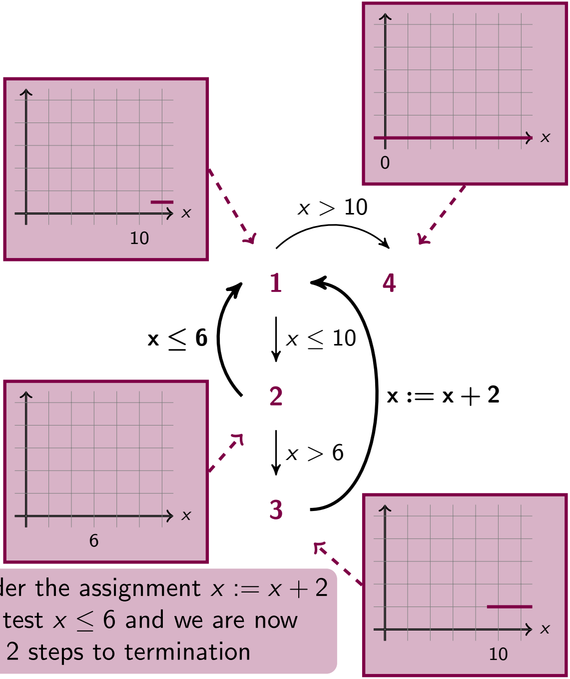
while ${}^{1}(x \leq 10)$ do

  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

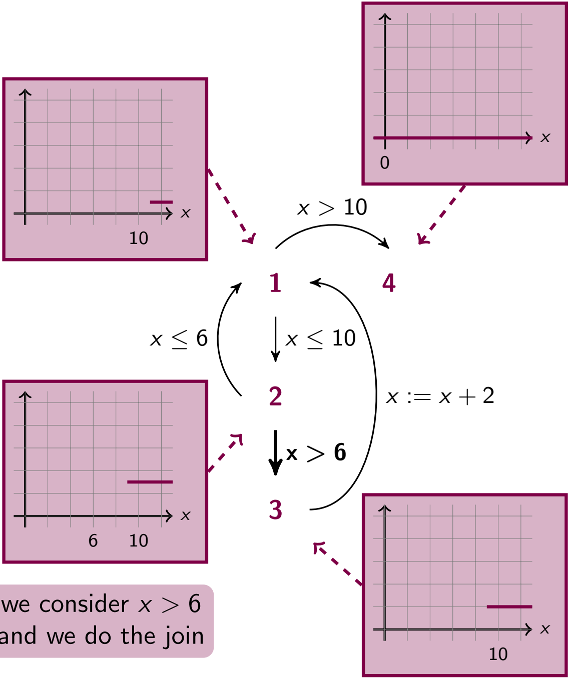
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



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

Example

int : x

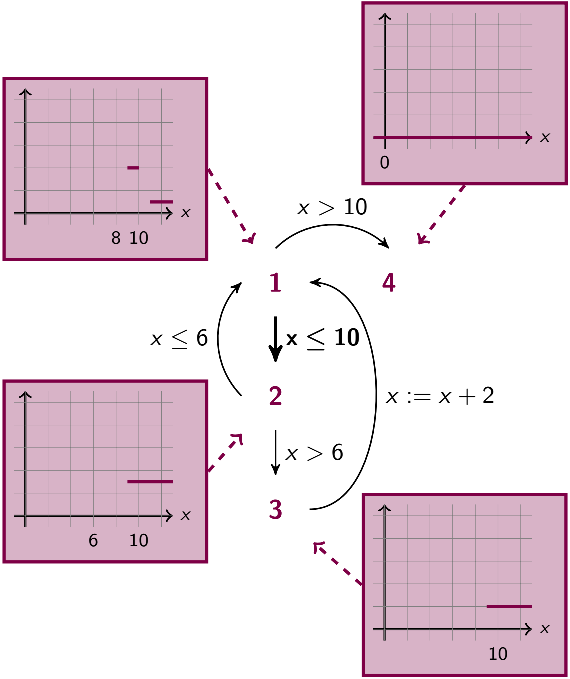
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



Example

int : x

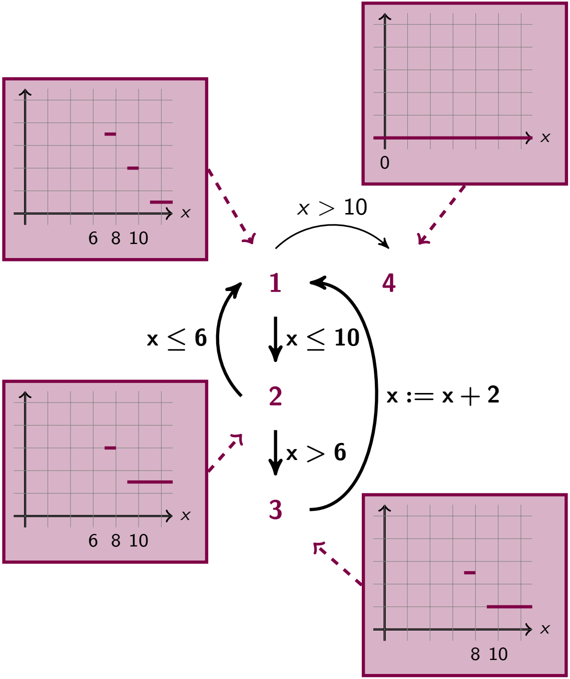
while $^{1}(x \leq 10)$ do

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

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

  fi

od^{4}



we do the widening

Example

int : x

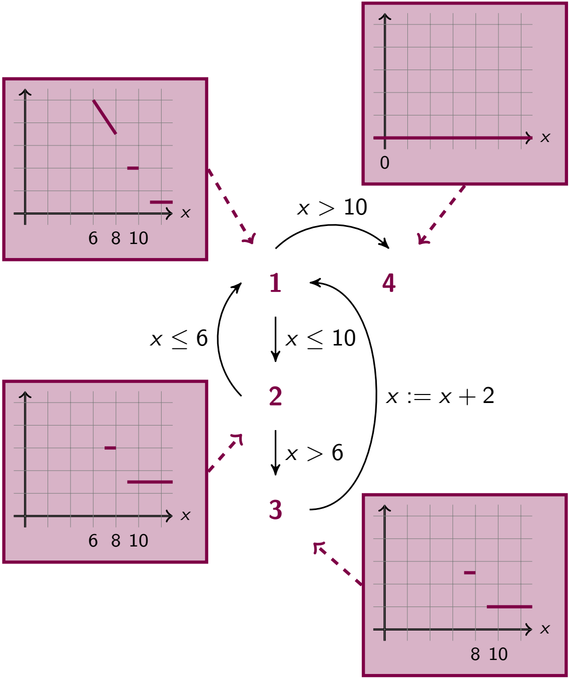
while $^{1}(x \leq 10)$ do

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

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

fi

od^{4}



Example

int : x

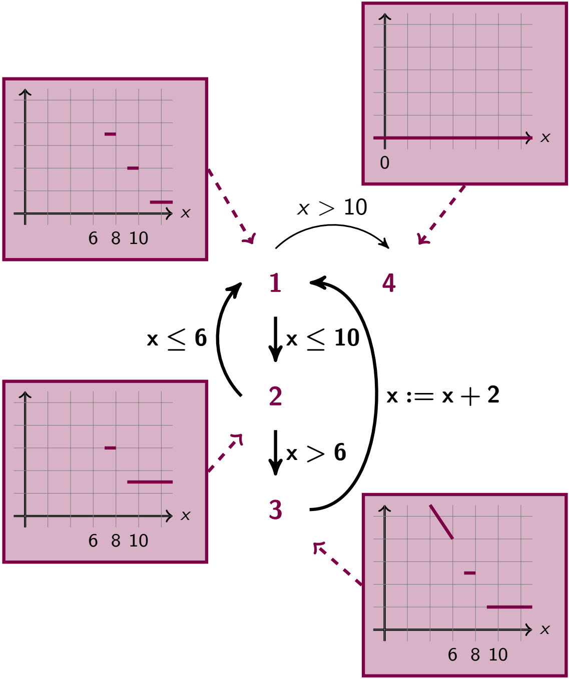
while [1]$(x \leq 10)$ do

  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}