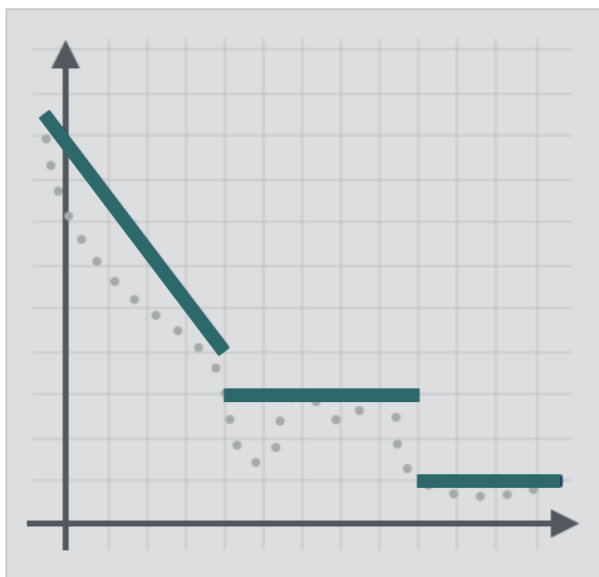


Liveness Analysis

MPRI 2-6: Abstract Interpretation,
Application to Verification and Static Analysis



Liveness Properties

- **Guarantee Properties**

“something good eventually happens at least once”

- Example: Program Termination

- **Recurrence Properties**

“something good eventually happens infinitely often”

- Example: Starvation Freedom



Zohar Manna

Amir Pnueli

Guarantee Properties

Guarantee Properties

“something good eventually happens at least once”

$AF \varphi$

$\varphi ::= e \bowtie 0 \mid \ell : e \bowtie 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \quad \ell \in \mathcal{L}$

Example:

$AF(x = 3)$ is satisfied for $\mathcal{F} \stackrel{\text{def}}{=} \{(1, \rho) \in \Sigma \mid \rho(x) \leq 3\}$

```
1x ← [-∞, +∞]
  while 2(x ≥ 0) do
    3x ← x + 1
  od4
  while 5(0 ≥ 0) do
    if 6(x ≤ 10) do
      7x ← x + 1
    else
      8x ← -x
    od9
```

Guarantee Semantics

$$\mathcal{R}_G^\varphi \stackrel{\text{def}}{=} \text{lfp}^{\preceq} \bar{F}_G[\{\sigma \in \Sigma \mid \sigma \models \varphi\}]$$

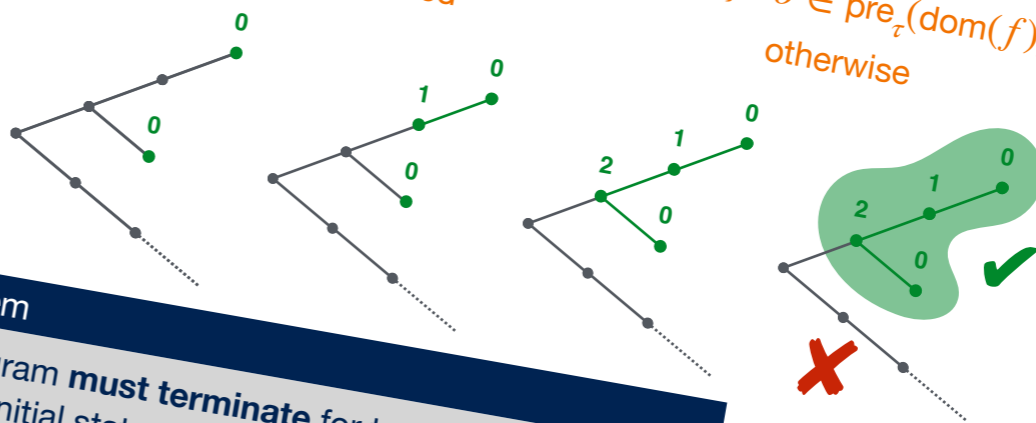
$$\bar{F}_G[S]f \stackrel{\text{def}}{=} \lambda\sigma. \begin{cases} 0 & \sigma \in S \\ \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \notin S \wedge \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Definite Termination Semantics

$$\mathcal{R}_M \stackrel{\text{def}}{=} \bar{\alpha}_M(\mathcal{T}_M) = \text{lfp}^{\preceq} \bar{F}_M$$

$f_1 \leq f_2 \stackrel{\text{def}}{=} \text{dom}(f_1) \subseteq \text{dom}(f_2) \wedge \forall x \in \text{dom}(f_1): f_1(x) \leq f_2(x)$
computational order

$$\bar{F}_M(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \\ \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$



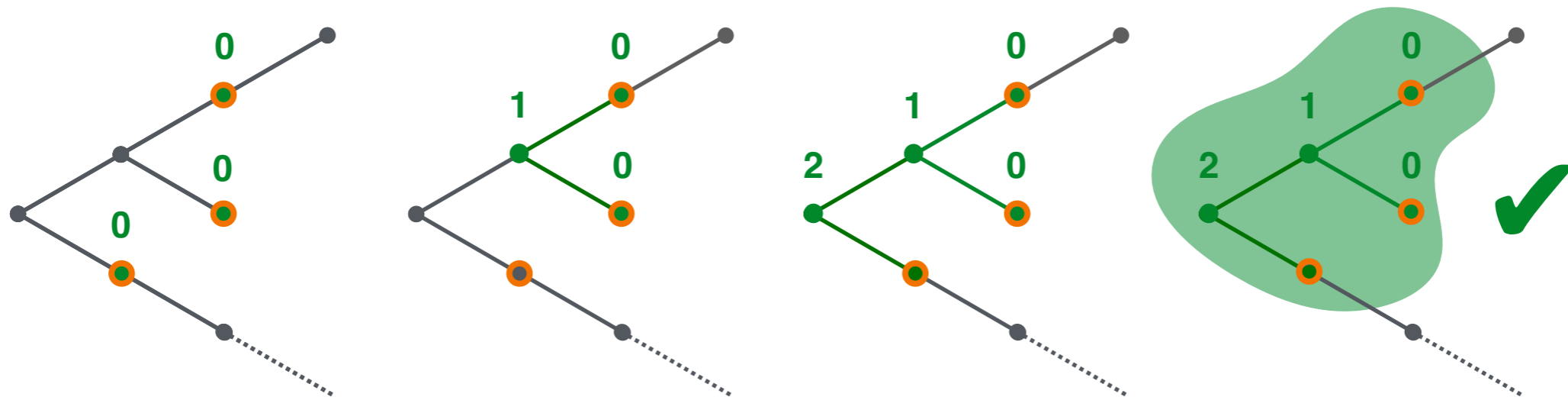
Theorem

A program **must terminate** for traces starting from a set of initial states \mathcal{I} if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_M)$

Guarantee Semantics

$$\mathcal{R}_G^\varphi \stackrel{\text{def}}{=} \text{lfp}^{\preceq} \bar{F}_G[\{\sigma \in \Sigma \mid \sigma \models \varphi\}]$$

$$\bar{F}_G[S]f \stackrel{\text{def}}{=} \lambda\sigma. \begin{cases} 0 & \sigma \in S \\ \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \notin S \wedge \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$



Theorem

A program satisfies a **guarantee property** $\text{AF } \varphi$ for traces starting from a set of initial states \mathcal{I} if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_G^\varphi)$

Abstract Guarantee Semantics

For each program instruction stat , we define $\mathcal{R}_G^{\varphi\#}[\![\text{stat}]\!] : \mathcal{A} \rightarrow \mathcal{A}$:

- $\mathcal{R}_G^{\varphi\#}[\![\ell X \leftarrow e]\!]t \stackrel{\text{def}}{=} \text{RESET}_A^G[\![\varphi]\!](\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]t)$

Abstract Definite Termination Semantics

For each program instruction stat , we define a transformer $\mathcal{R}_M^{\#}[\![\text{stat}]\!] : \mathcal{A} \rightarrow \mathcal{A}$:

- $\mathcal{R}_M^{\#}[\![\ell X \leftarrow e]\!]t \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]t$
- $\mathcal{R}_M^{\#}[\![\text{if } \ell e \bowtie 0 \text{ then } s]\!]t \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathcal{R}_M^{\#}[\![s]\!]t) \vee_T \text{FILTER}_A[\![e \bowtie 0]\!]t$
- $\mathcal{R}_M^{\#}[\![\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]\!]t \stackrel{\text{def}}{=} \text{lfp}^{\#} \overline{F}_M^{\#}$
 where $\overline{F}_M^{\#}(x) \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathcal{R}_M^{\#}[\![s]\!]x) \vee_T \text{FILTER}_A[\![e \bowtie 0]\!]t$
- $\mathcal{R}_M^{\#}[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathcal{R}_M^{\#}[\![s_1]\!](\mathcal{R}_M^{\#}[\![s_2]\!]t)$

Language syntax

Programs and executions

```

 $\text{stat}^{\ell} ::= X \leftarrow \text{exp}^{\ell}$  (assignment)
            $\text{if } \text{exp}^{\ell} \bowtie 0 \text{ then } \text{stat}^{\ell}$  (conditional)
            $\text{while } \text{exp}^{\ell} \bowtie 0 \text{ do } \text{stat}^{\ell} \text{ done}^{\ell}$  (loop)
            $\text{stat}^{\ell}; \text{stat}^{\ell}$  (sequence)
            $X$  (variable)
            $-\text{exp}^{\ell}$  (negation)
            $\text{exp}^{\ell} \diamond \text{exp}^{\ell}$  (binary operation)
            $c$  (constant  $c \in \mathbb{Z}$ )
            $[c, c']$  (random input,  $c, c' \in \mathbb{Z} \cup \{\pm\infty\}$ )
    
```

Simple structured, numeric language

- $X \in \mathcal{V}$, where \mathcal{V} is a finite set of **program variables**
- $\ell \in \mathcal{L}$, where \mathcal{L} is a finite set of **control points**
- numeric expressions: $\bowtie \in \{=, <, >, \dots\}$, $\diamond \in \{+, -, \cdot, /\}$
- **random inputs**: $X \leftarrow [c, c']$

model environment, parametric programs, unknown functions, ...

Course 2 Program Semantics and Properties Antoine Miné p. 3 / 99

Lesson 12 Termination Analysis Caterina Urban 84

Abstract Guarantee Semantics

For each program instruction stat , we define $\mathcal{R}_G^{\varphi\#}[\![\text{stat}]\!] : \mathcal{A} \rightarrow \mathcal{A}$:

- $\mathcal{R}_G^{\varphi\#}[\![\ell X \leftarrow e]\!]t \stackrel{\text{def}}{=} \text{RESET}_A^G[\![\varphi]\!](\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]t)$
- $\mathcal{R}_G^{\varphi\#}[\![\text{if } \ell e \bowtie 0 \text{ then } s]\!]t \stackrel{\text{def}}{=} \text{RESET}_A^G[\![\varphi]\!](X)$
 where $X \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathcal{R}_G^{\varphi\#}[\![s]\!]t) \vee_T \text{FILTER}_A[\![e \bowtie 0]\!]t)$
- $\mathcal{R}_G^{\varphi\#}[\![\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}\!]t \stackrel{\text{def}}{=} \text{lfp}^{\#} \overline{F}_G^{\varphi\#}$
 where $\overline{F}_G^{\varphi\#}(x) \stackrel{\text{def}}{=} \text{RESET}_A^G[\![\varphi]\!](X)$
 $X \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathcal{R}_G^{\varphi\#}[\![s]\!]x) \vee_T \text{FILTER}_A[\![e \bowtie 0]\!](t)$
- $\mathcal{R}_G^{\varphi\#}[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathcal{R}_G^{\varphi\#}[\![s_1]\!](\mathcal{R}_G^{\varphi\#}[\![s_2]\!]t)$

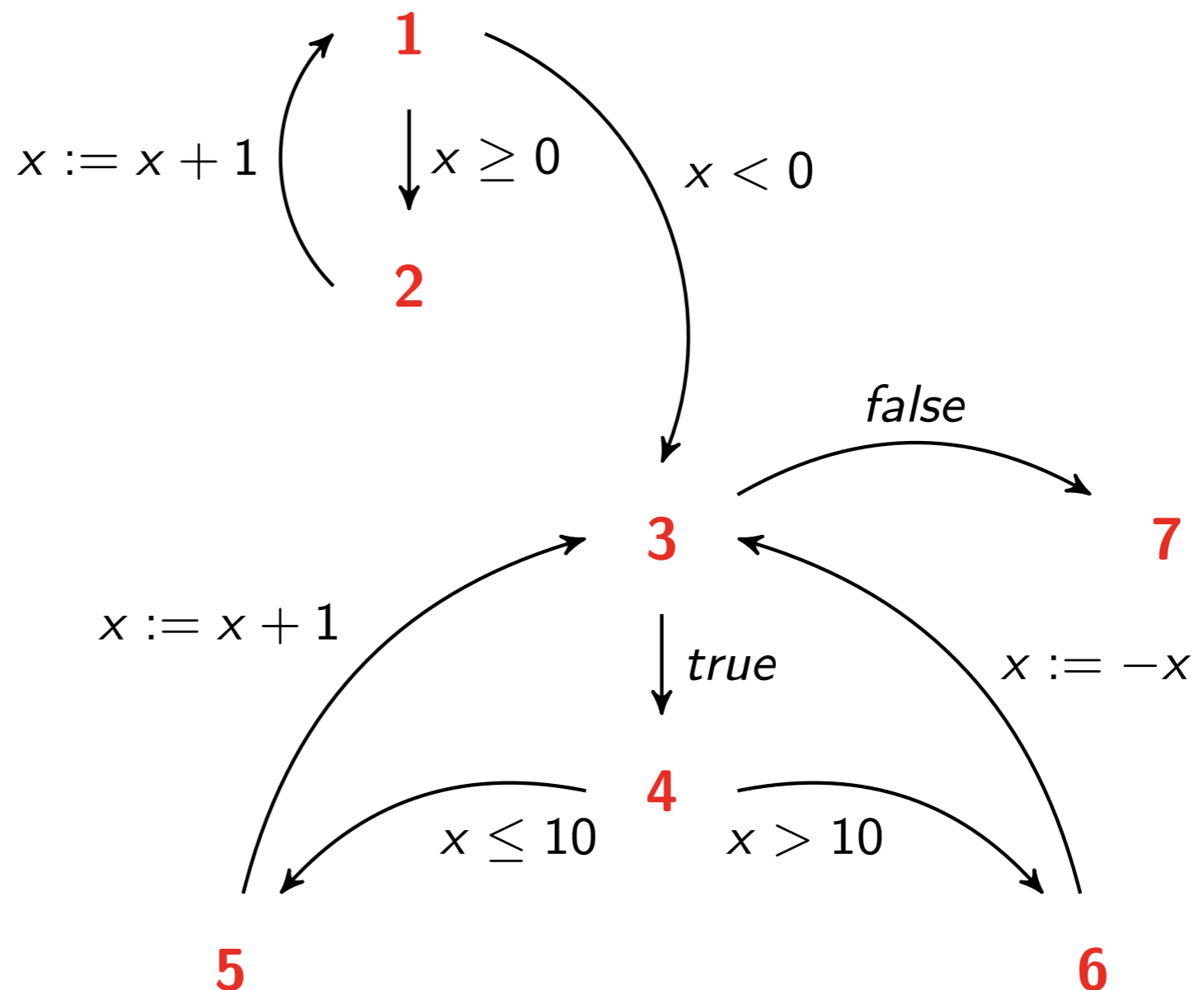
Abstract Guarantee Semantics

Example

```
int : x, y
while 1(x ≥ 0) do
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7
```

Property

AF (x = 3)



Abstract Guarantee Semantics

Example

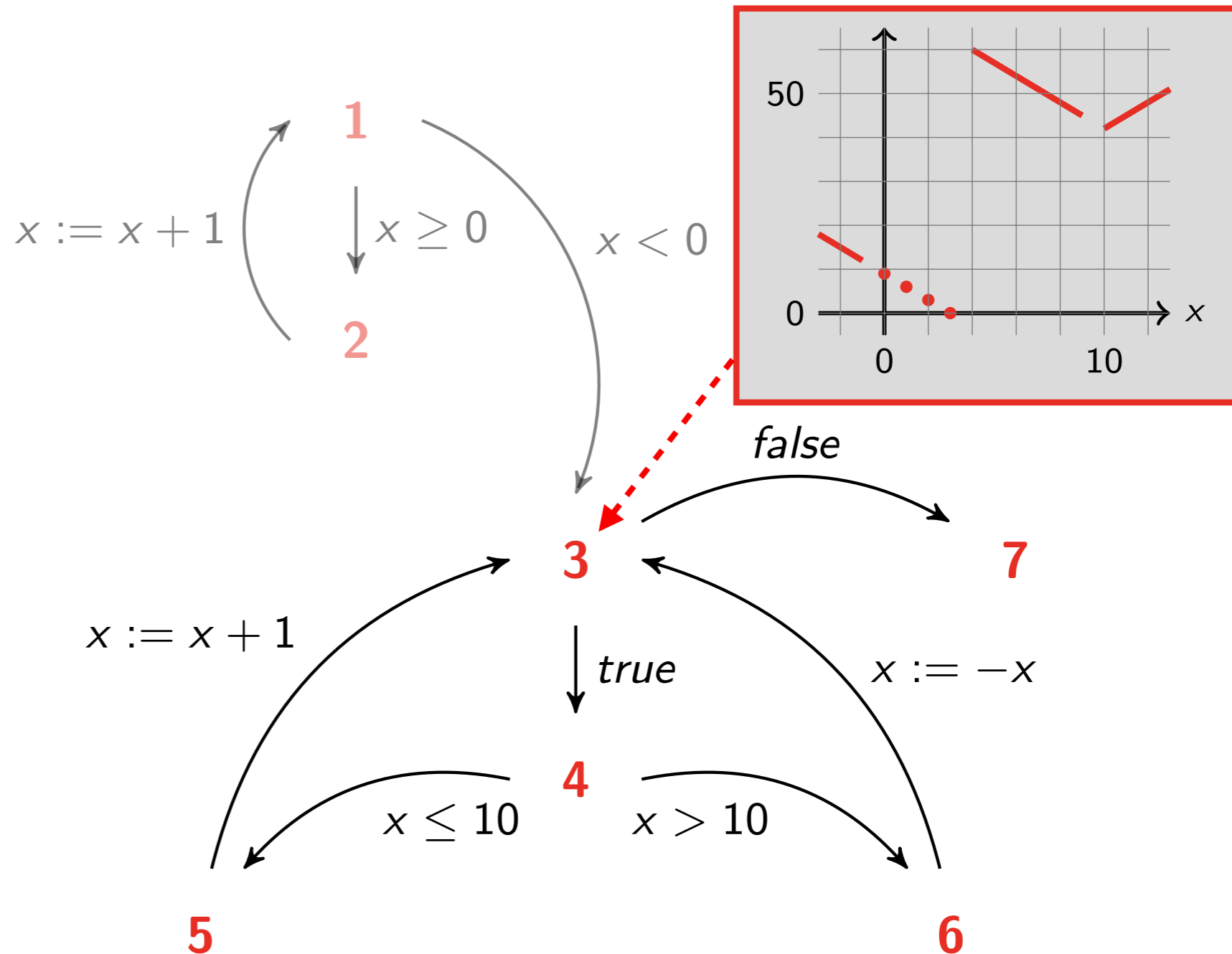
```

int : x, y
while 1(x ≥ 0) do
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```

Property

AF (x = 3)



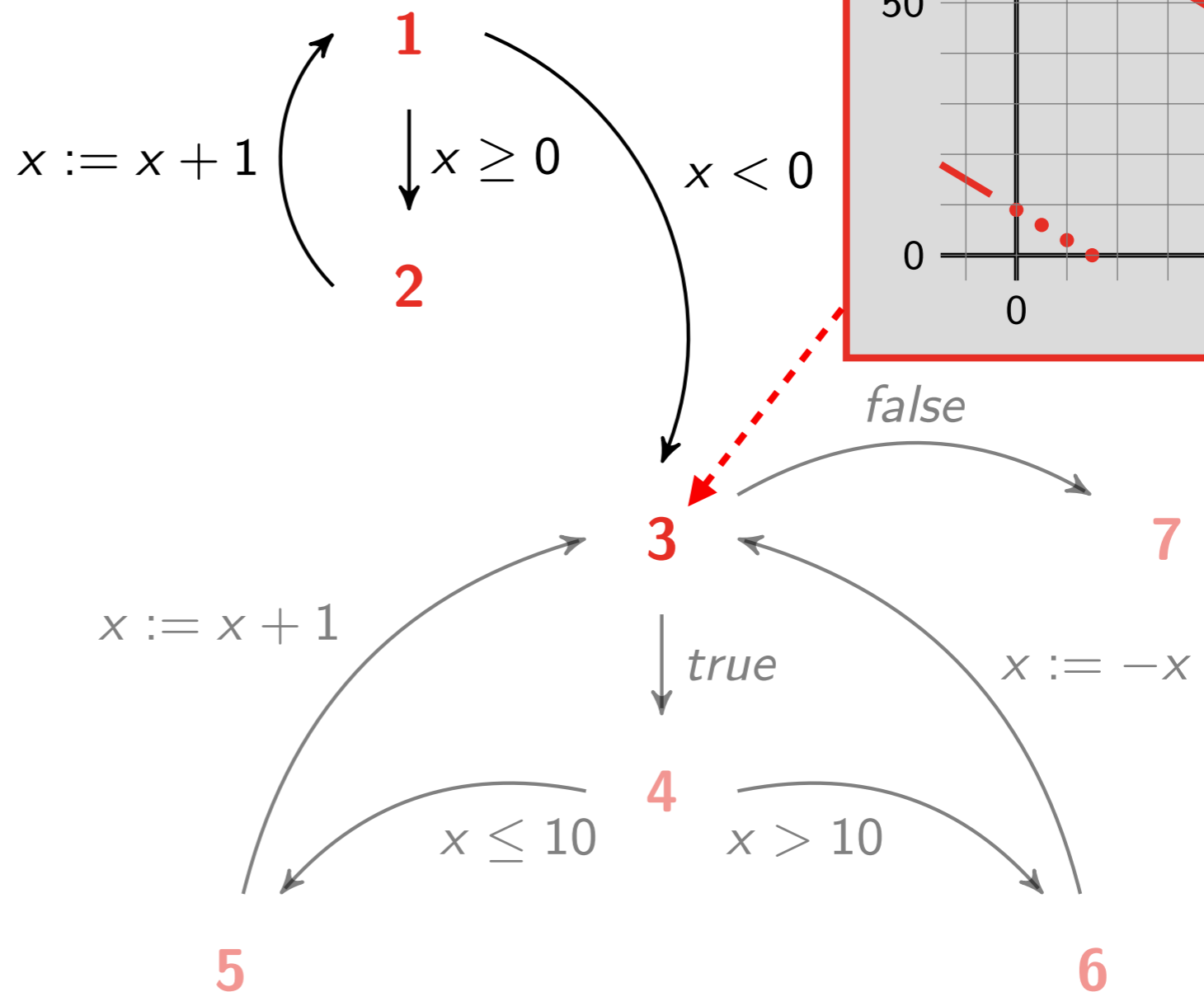
Abstract Guarantee Semantics

Example

```
int : x, y
while 1(x ≥ 0) do
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7
```

Property

AF (x = 3)



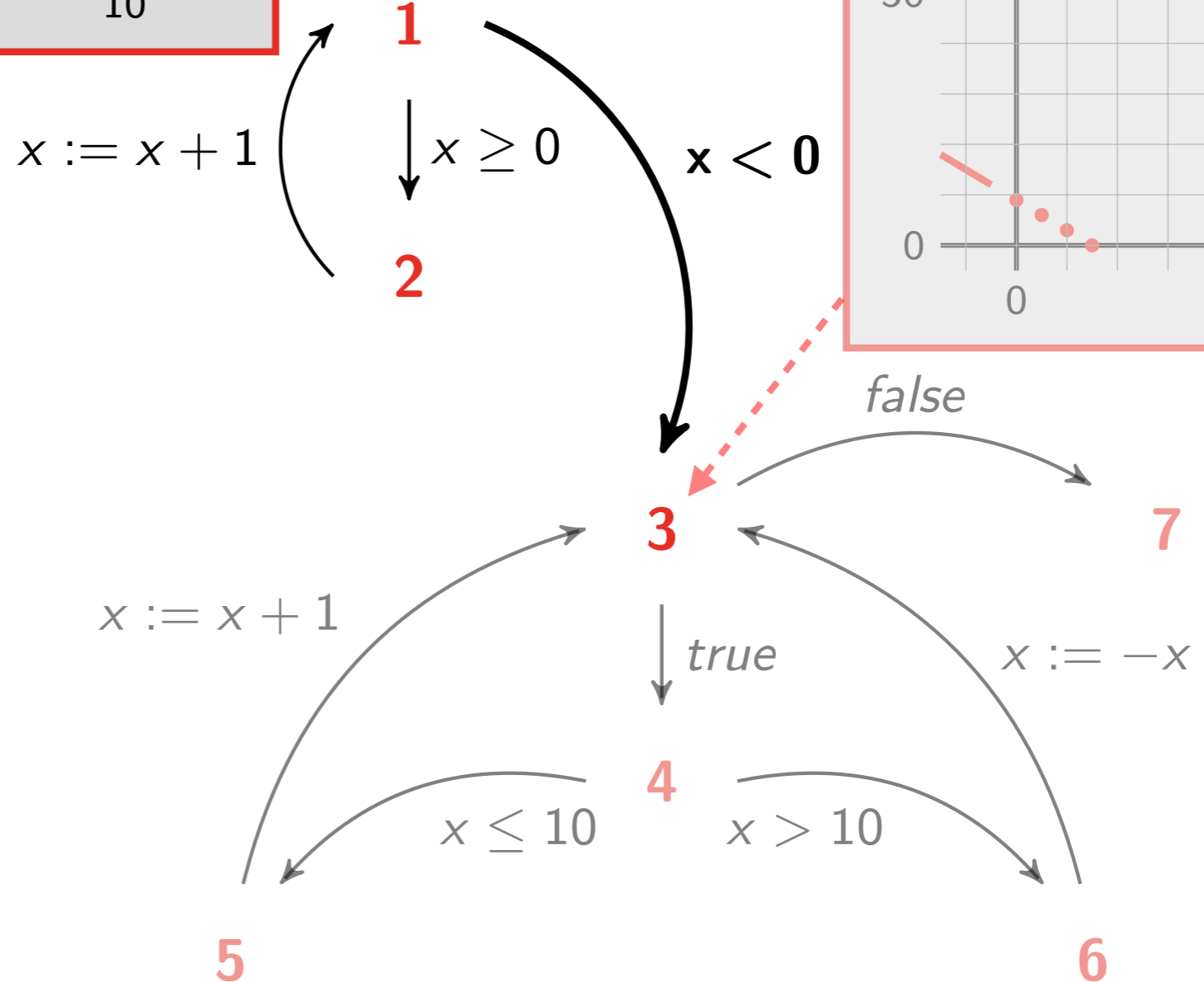
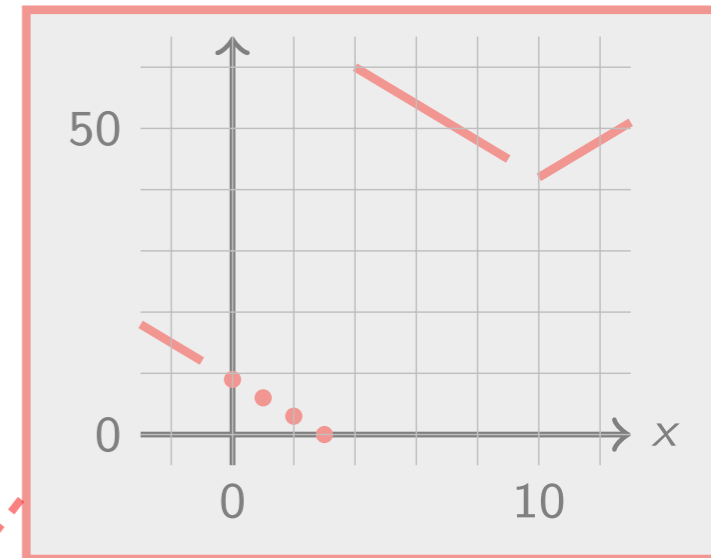
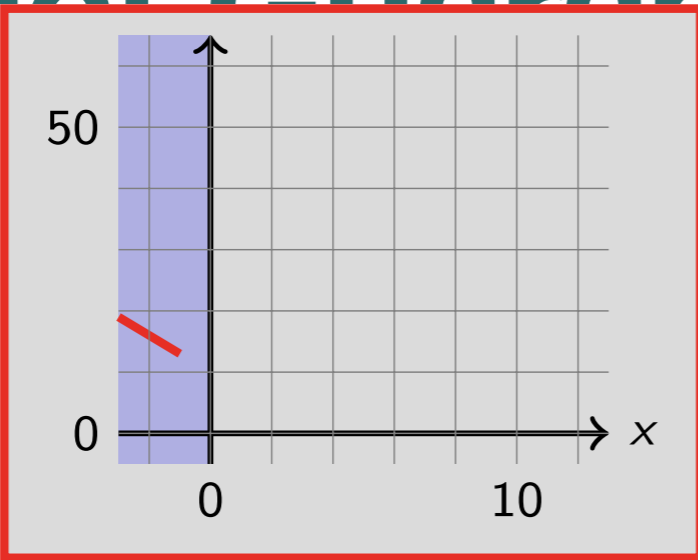
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



Property

AF (x = 3)

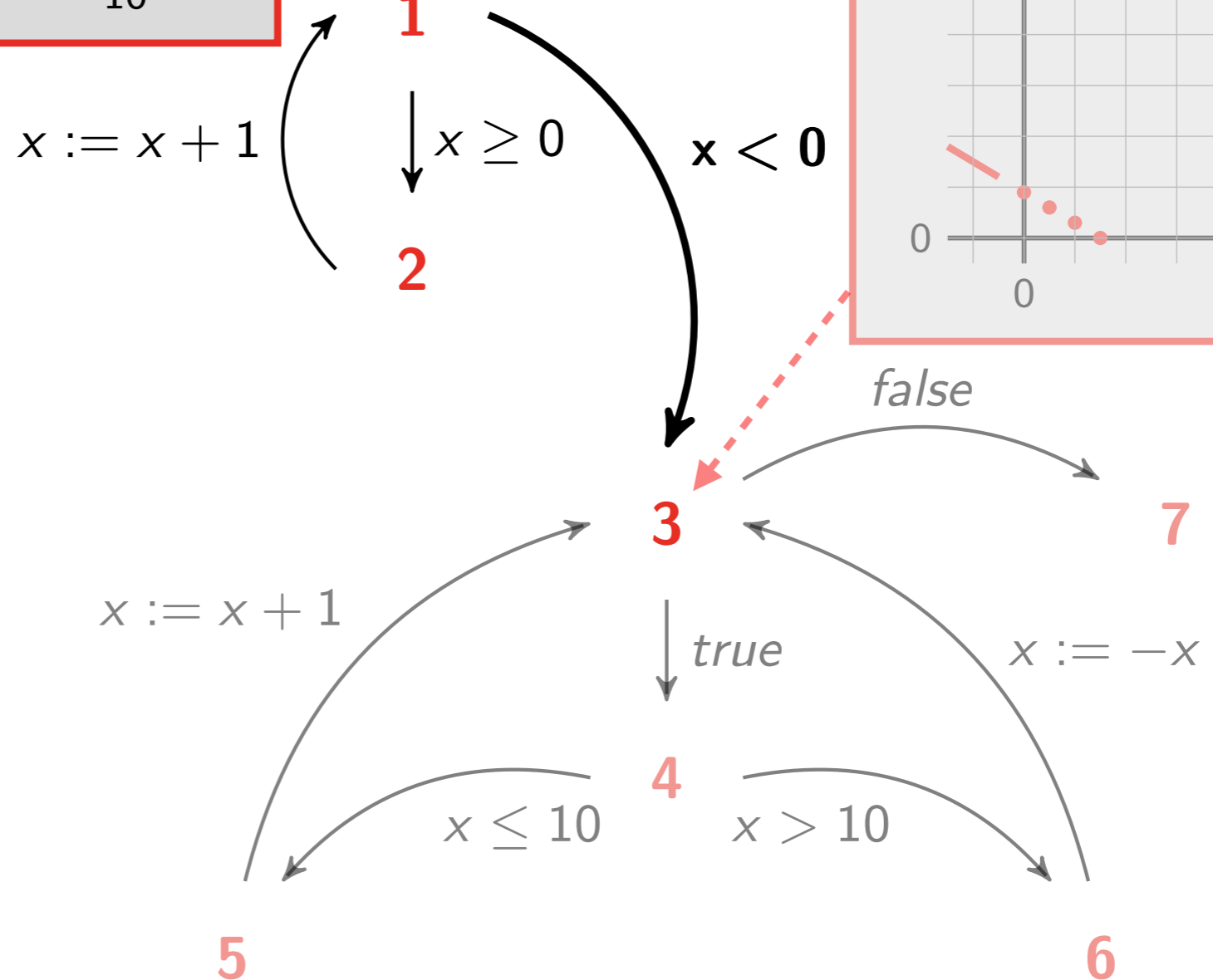
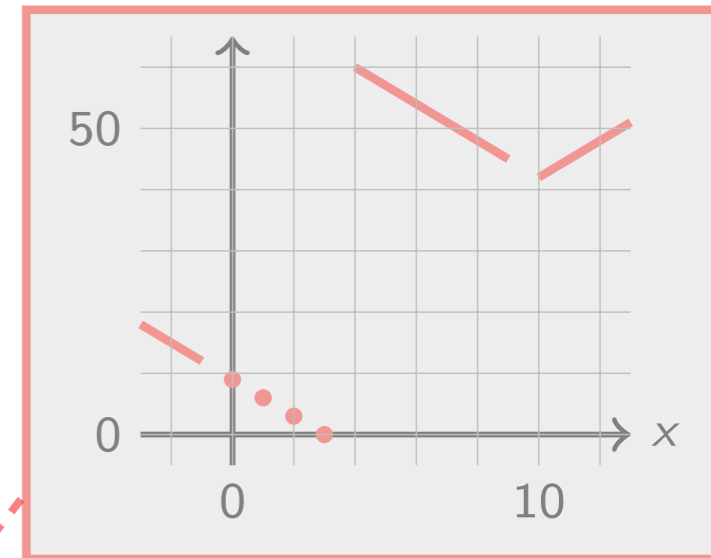
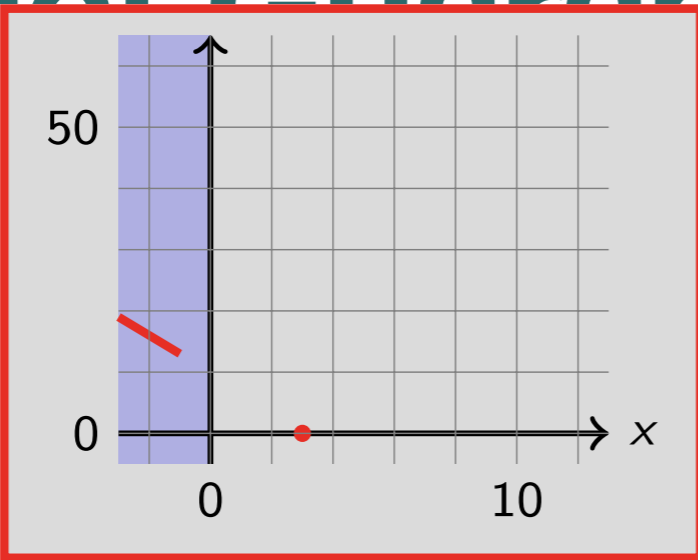
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



Property

AF (x = 3)

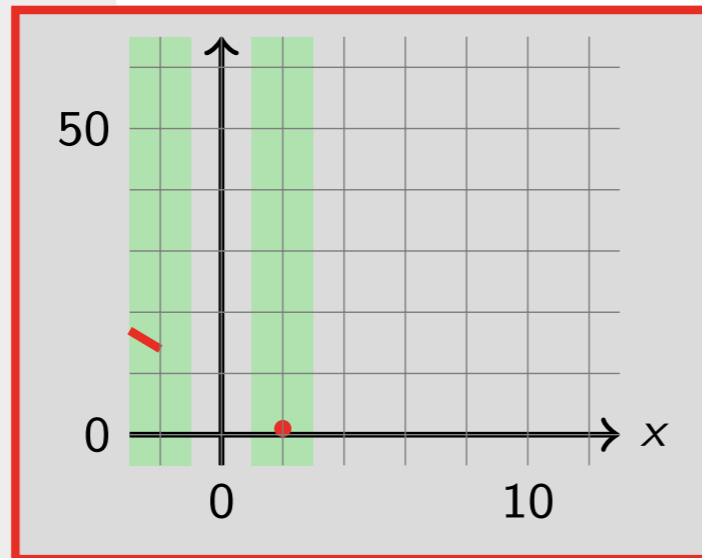
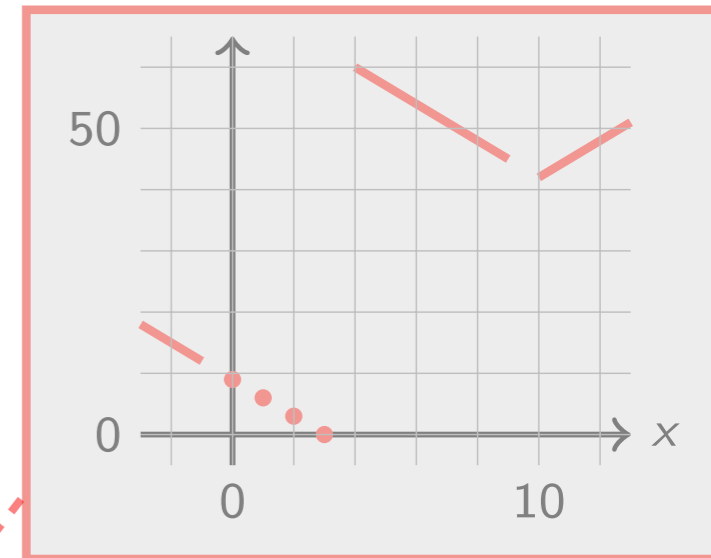
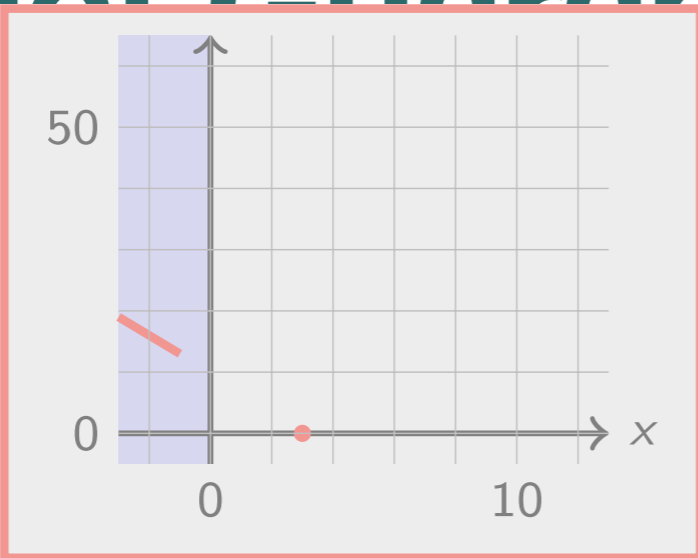
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



$x := x + 1$

1

$x \geq 0$

$x < 0$

2

false

3

7

true

$x := -x$

4

$x > 10$

$x \leq 10$

5

6

Property

$AF(x = 3)$

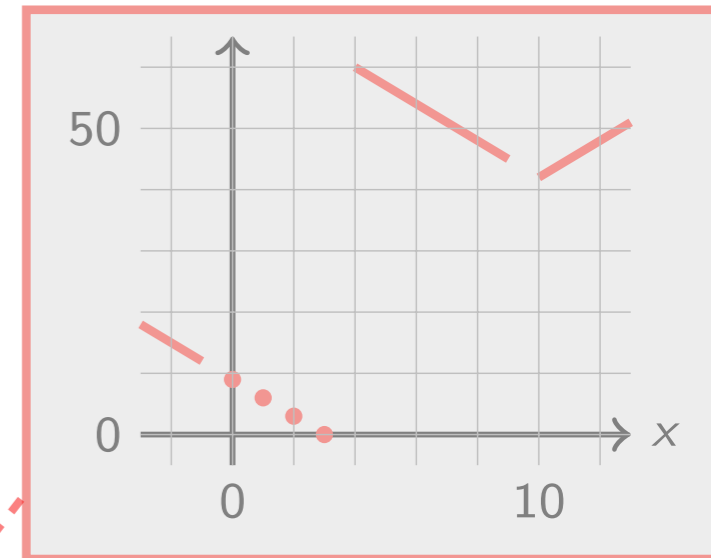
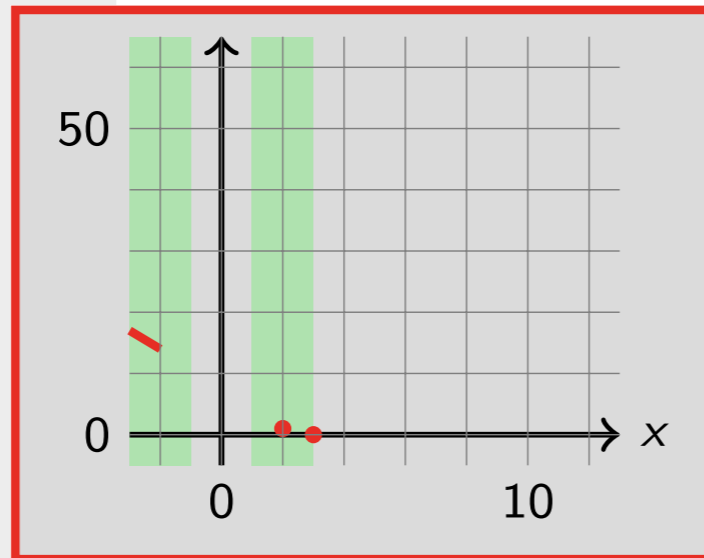
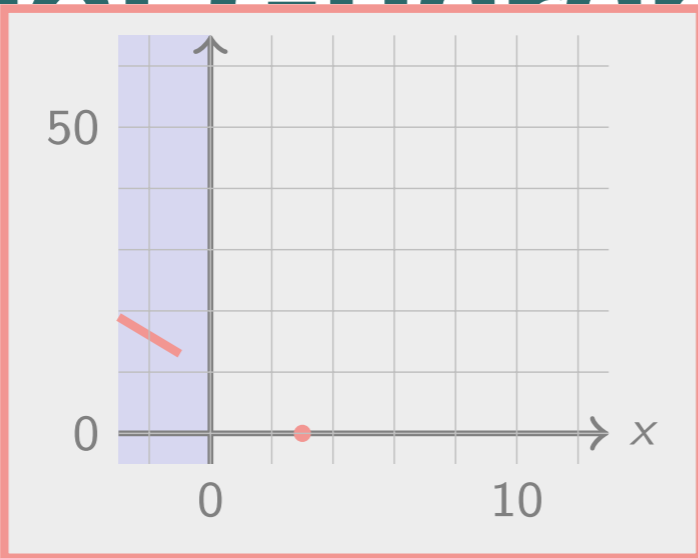
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



$x := x + 1$

1

$x \geq 0$

$x < 0$

2

false

3

7

true

$x := -x$

4

$x > 10$

$x \leq 10$

5

6

Property

$AF(x = 3)$

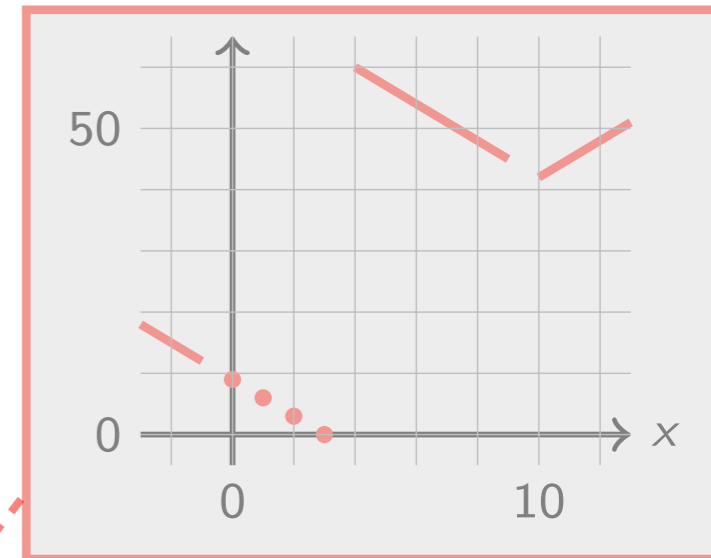
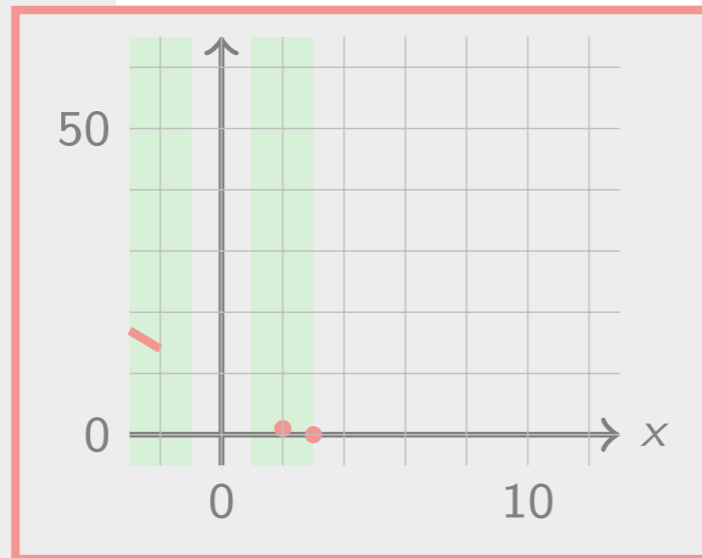
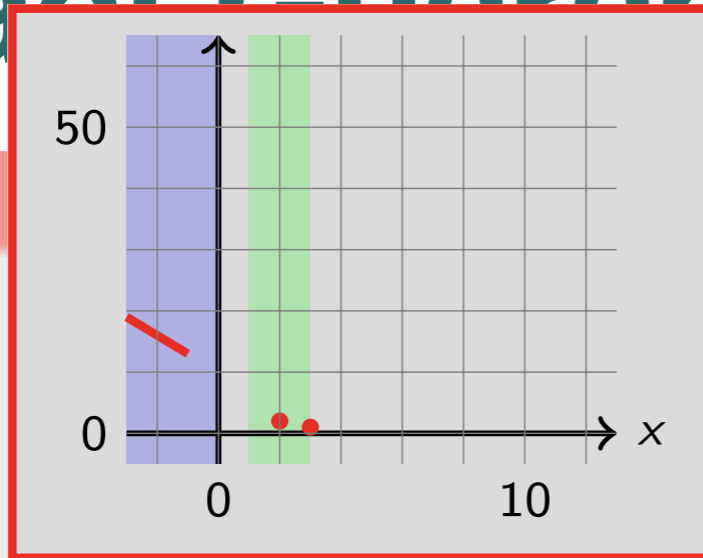
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od 7

```



$x := x + 1$

$x \geq 0$

$x < 0$

false

true

$x := -x$

$x \leq 10$

$x > 10$

Property

AF ($x = 3$)

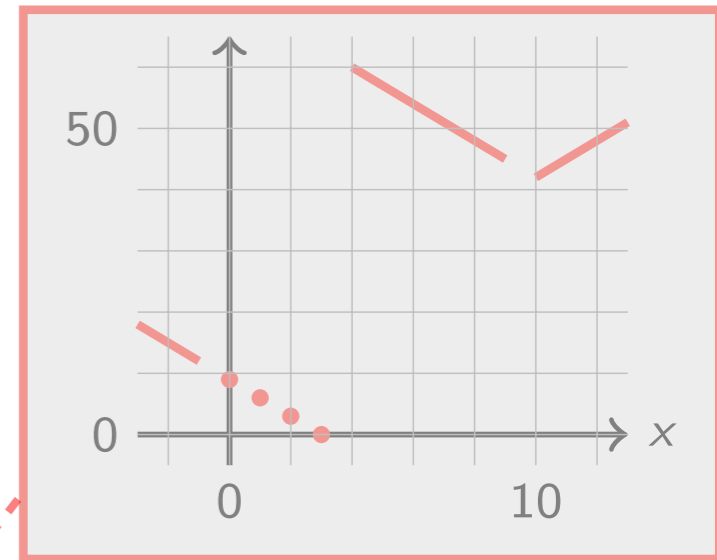
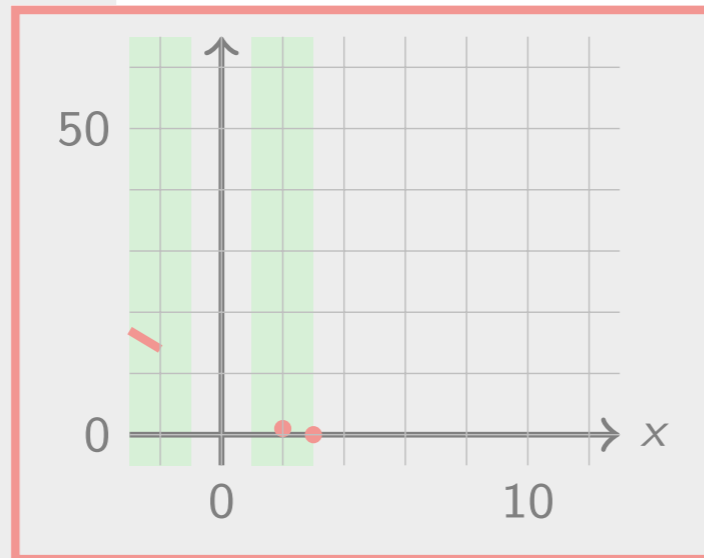
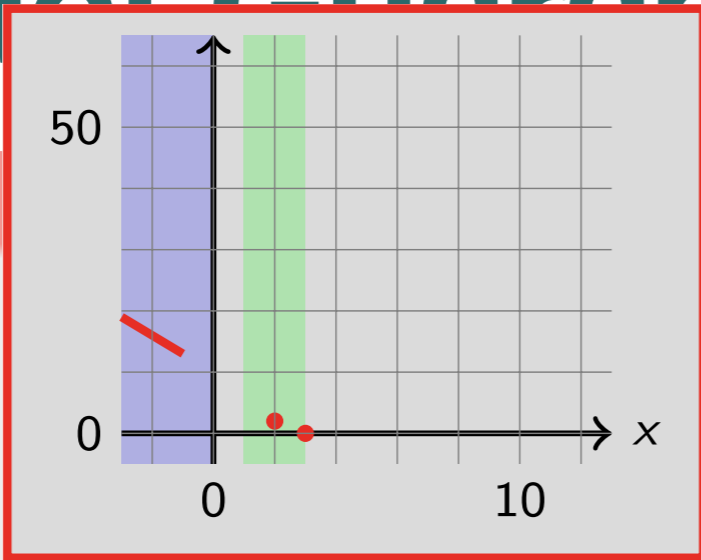
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od 7

```



$x := x + 1$

$x \geq 0$

$x < 0$

false

true

$x := -x$

$x \leq 10$

$x > 10$

5

6

Property

AF ($x = 3$)

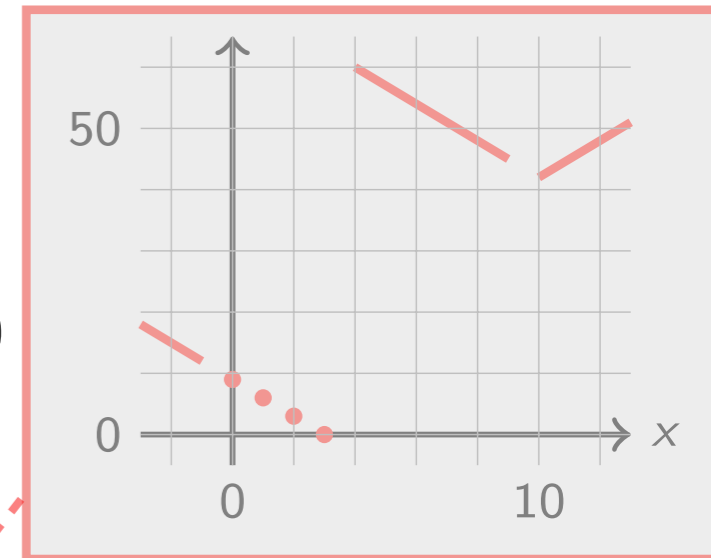
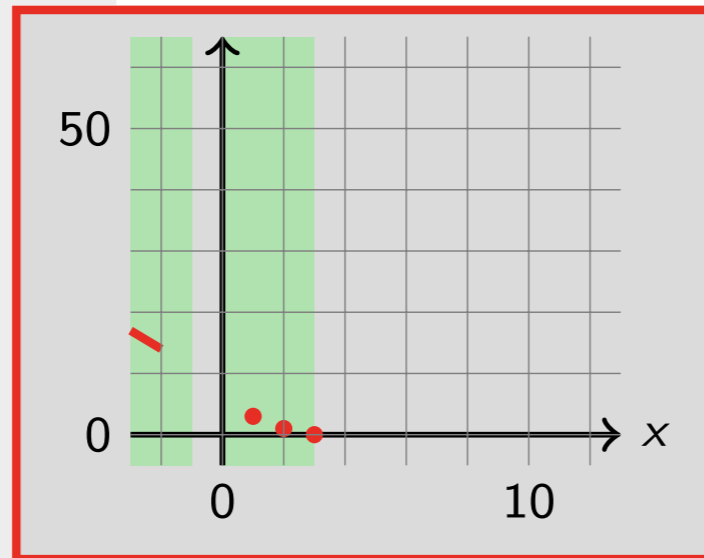
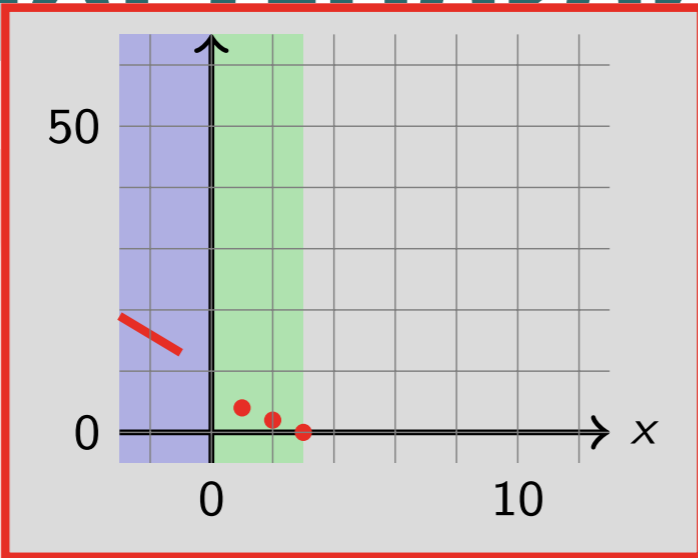
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od 7

```



$x := x + 1$

1
↓ $x \geq 0$

$x < 0$

2

false

7

true

$x := -x$

$x \leq 10$

$x > 10$

5

6

Property

$AF(x = 3)$

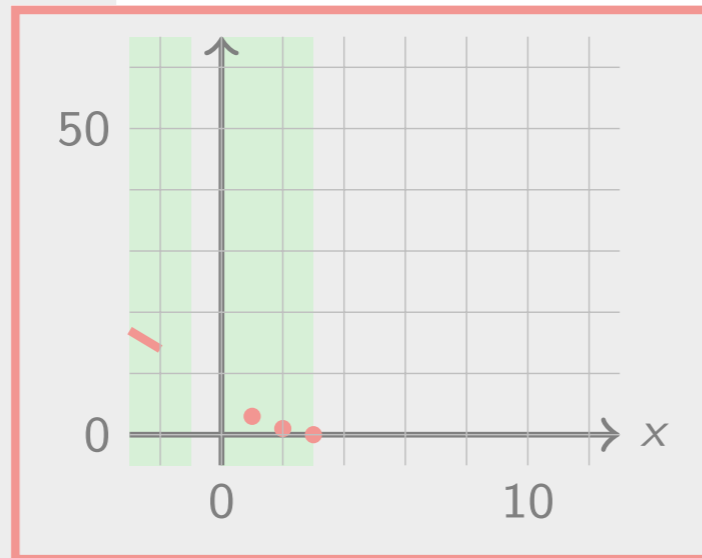
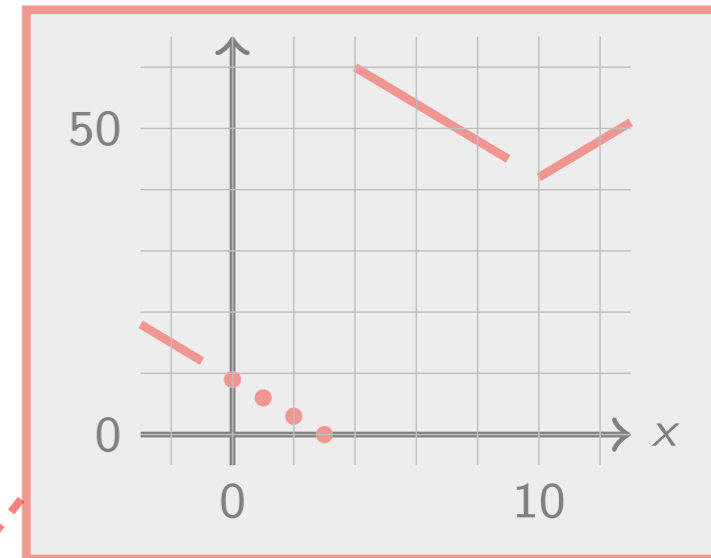
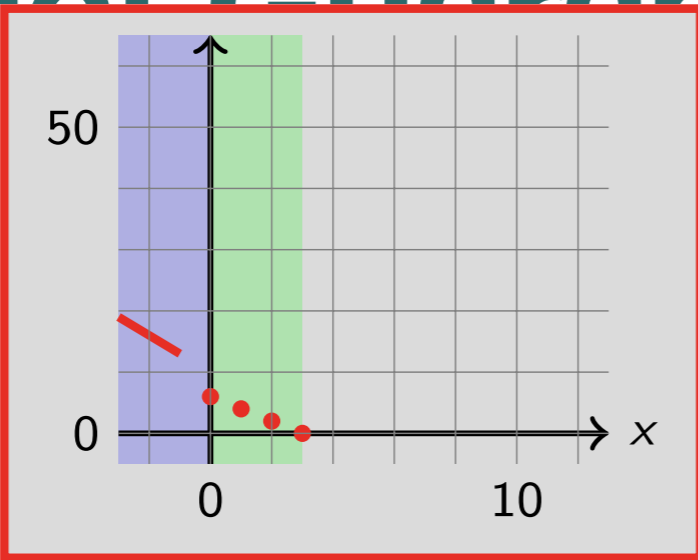
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od 7

```



$x := x + 1$

1

$x \geq 0$

$x < 0$

2

false

3

7

true

$x := -x$

4

$x > 10$

$x \leq 10$

5

6

Property

AF ($x = 3$)

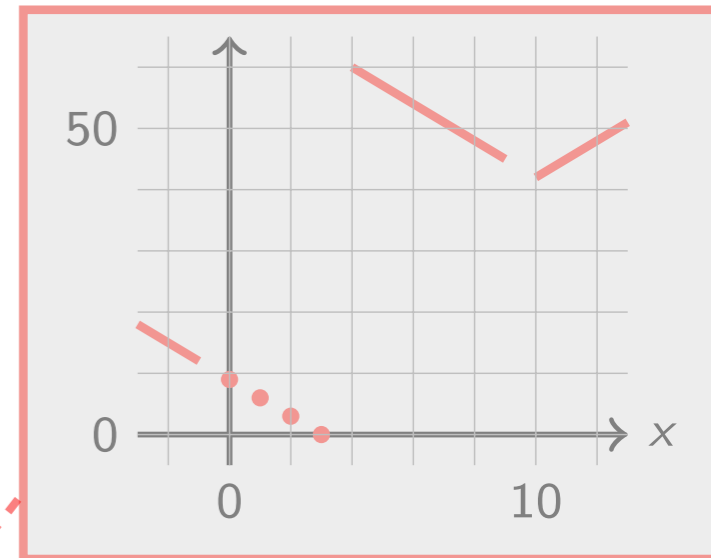
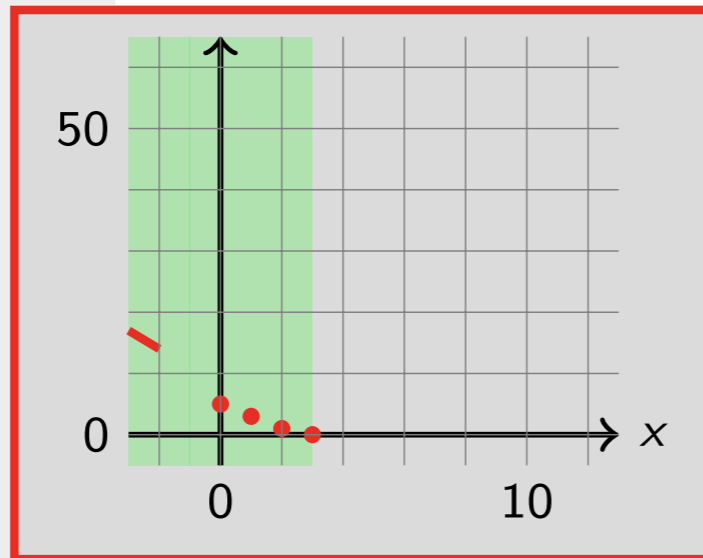
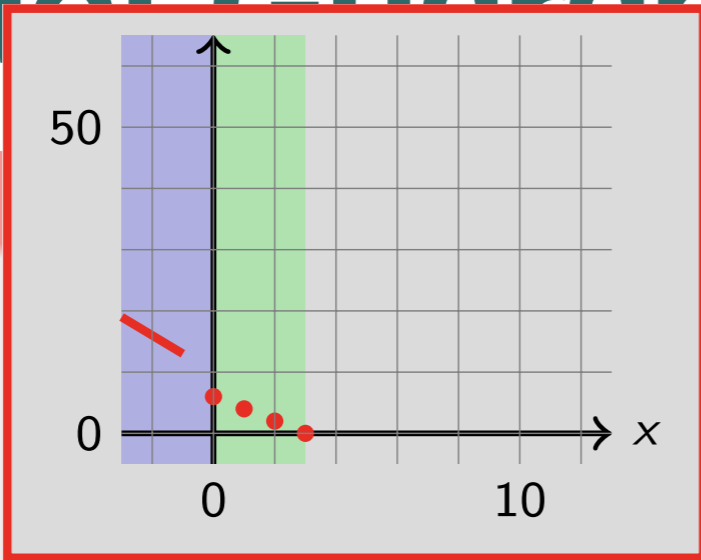
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



$x := x + 1$

1

$x \geq 0$

$x < 0$

2

false

3

7

true

$x := -x$

4

$x > 10$

$x \leq 10$

5

6

Property

AF ($x = 3$)

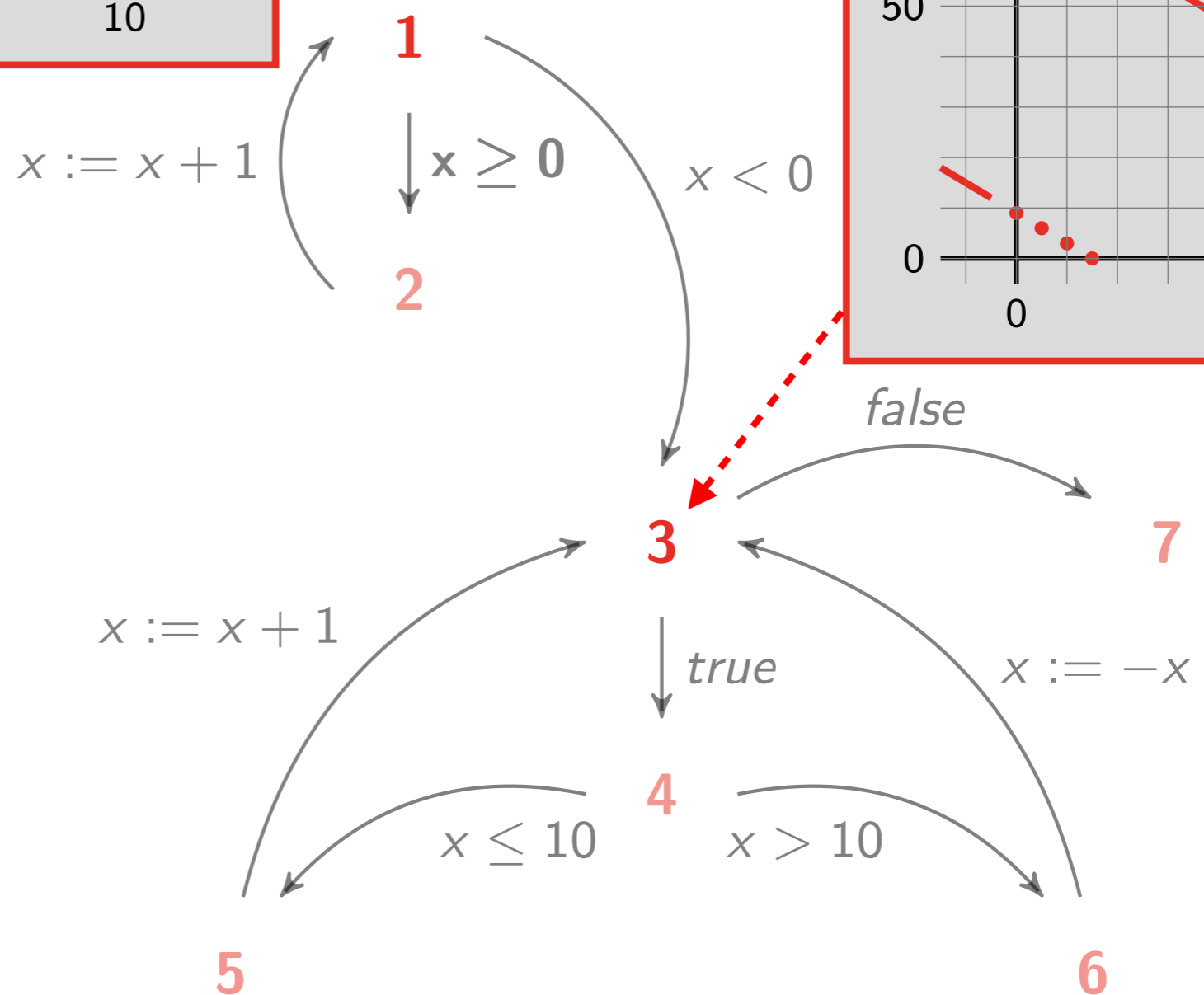
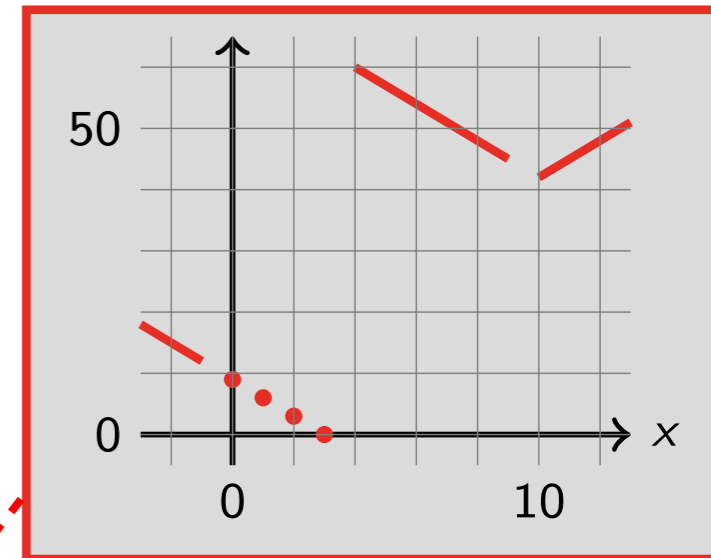
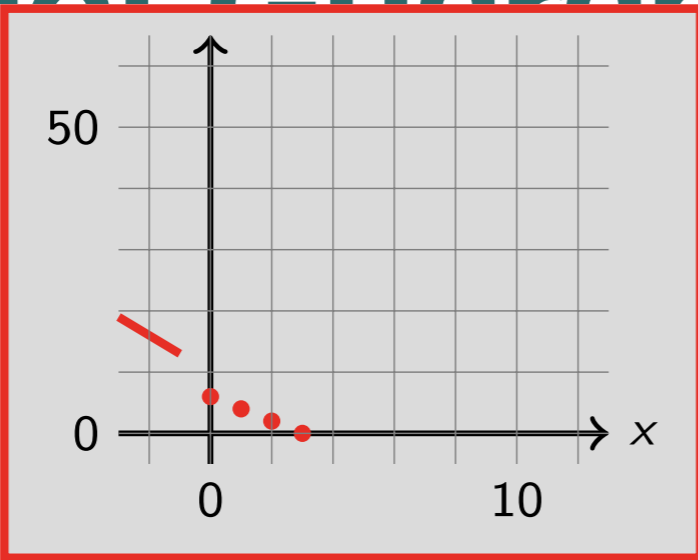
Abstract Guarantee Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



Property

AF (x = 3)

Abstract Guarantee Semantics

Example

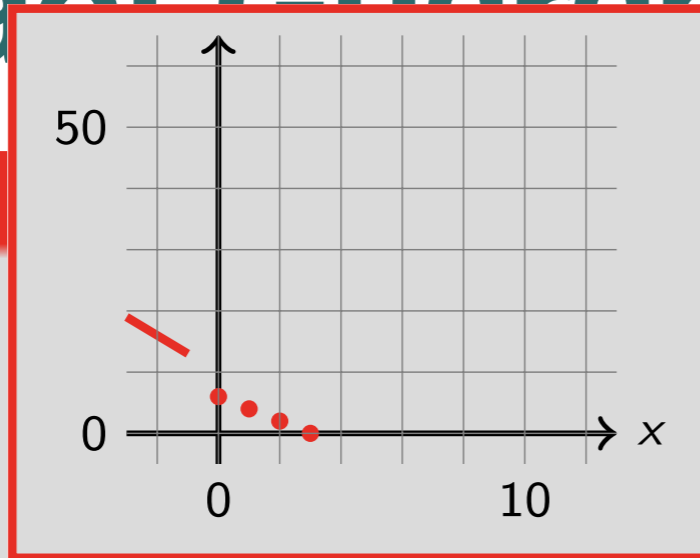
```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

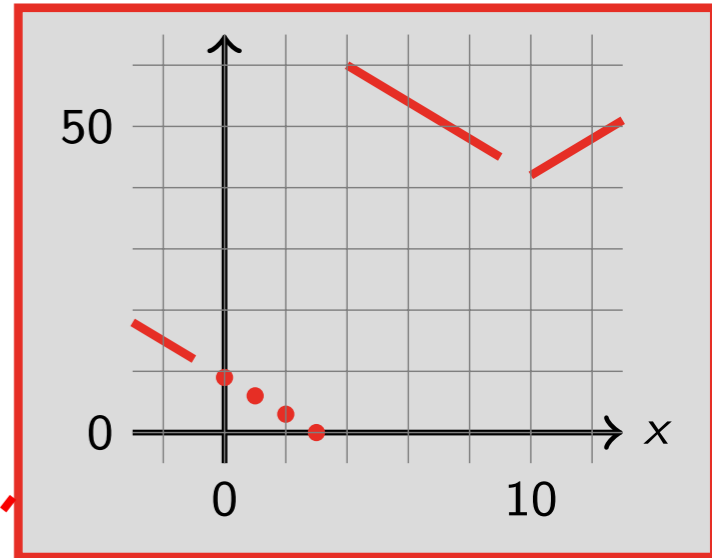
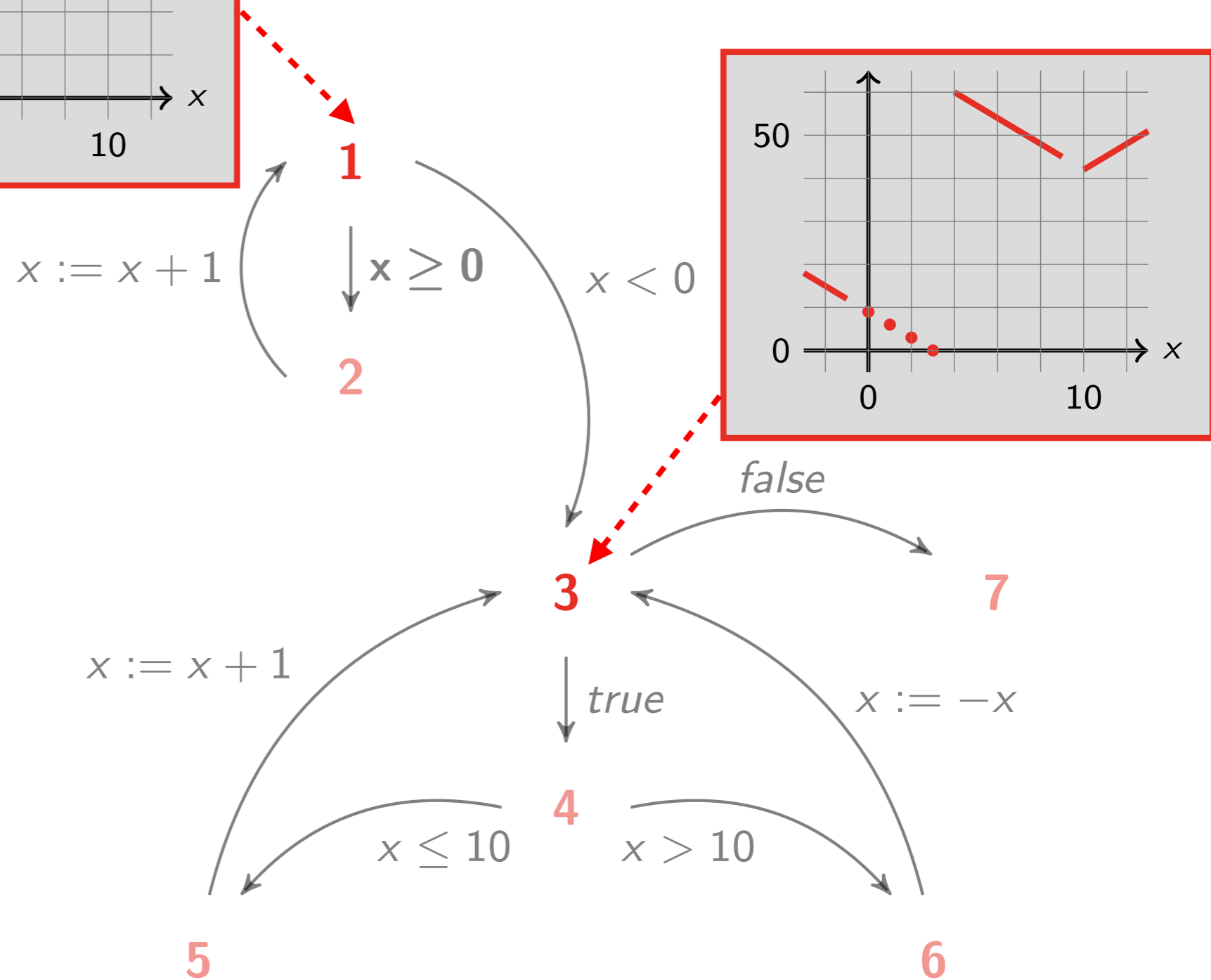
```

Property

AF (x = 3)



the analysis gives $x \leq 3$ as **sufficient precondition**



Abstract Guarantee Semantics

Definition

The **abstract guarantee semantics** $\mathcal{R}_G^{\varphi\#}[\text{stat}^\ell] \in \mathcal{A}$ of a program stat^ℓ is:

$$\mathcal{R}_G^{\varphi\#}[\text{stat}^\ell] \stackrel{\text{def}}{=} \mathcal{R}_G^{\varphi\#}[\text{stat}](\text{RESET}_A^G[\varphi](\text{LEAF}: \perp_F))$$

where $\mathcal{R}_G^{\varphi\#}[\text{stat}]: \mathcal{A} \rightarrow \mathcal{A}$ is the abstract guarantee semantics of each program instruction stat

Corollary (Soundness)

A program stat^ℓ satisfies a **guarantee property** $\text{AF } \varphi$ for traces starting from a set of initial states \mathcal{I} if $\mathcal{I} \subseteq \text{dom}(\gamma_A(\mathcal{R}_G^{\varphi\#}[\text{stat}^\ell]))$

Recurrence Properties

Recurrence Properties

“something good eventually happens infinitely often”

$AG AF \varphi$

$\varphi ::= e \bowtie 0 \mid \ell : e \bowtie 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \qquad \ell \in \mathcal{L}$

Example:

$\mathbf{1}x \leftarrow [-\infty, +\infty]$ $AG AF (x = 3)$ is satisfied for $\mathcal{F} \stackrel{\text{def}}{=} \{(1, \rho) \in \Sigma \mid \rho(x) < 0\}$
 $\mathbf{while} \mathbf{2}(x \geq 0) \mathbf{do}$
 $\mathbf{3}x \leftarrow x + 1$
 \mathbf{od}^4
 $\mathbf{while} \mathbf{5}(0 \geq 0) \mathbf{do}$
 $\mathbf{if} \mathbf{6}(x \leq 10) \mathbf{do}$
 $\mathbf{7}x \leftarrow x + 1$
 \mathbf{else}
 $\mathbf{8}x \leftarrow -x$
 \mathbf{od}^9

Recurrence Semantics

$$\mathcal{R}_R^\varphi \stackrel{\text{def}}{=} \text{gfp}_{\mathcal{R}_G^\varphi} \preceq \bar{F}_R$$

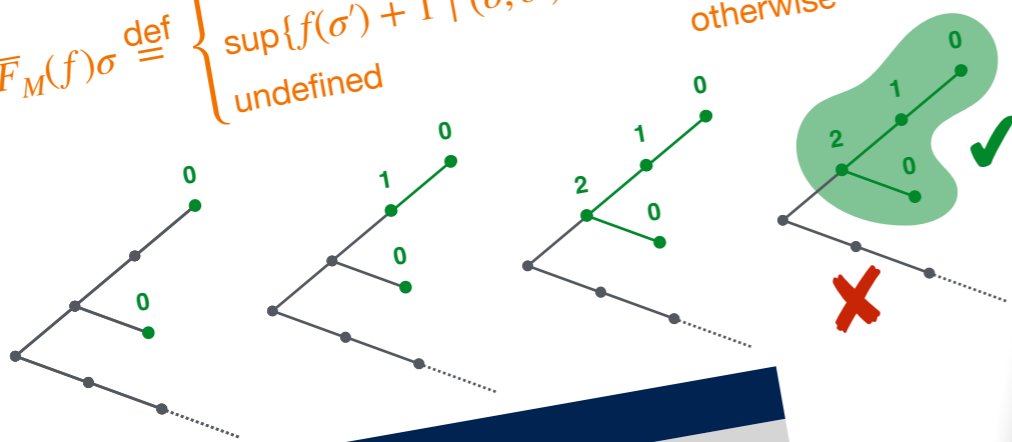
$$\bar{F}_R(f)\sigma \stackrel{\text{def}}{=} \begin{cases} f(s) & \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Definite Termination Semantics

$$\mathcal{R}_M \stackrel{\text{def}}{=} \bar{\alpha}_M(\mathcal{T}_M) = \text{lfp}_{\preceq} \bar{F}_M$$

$$\bar{F}_M(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \\ \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$f_1 \preceq f_2 \stackrel{\text{def}}{=} \text{dom}(f_1) \subseteq \text{dom}(f_2) \wedge \forall x \in \text{dom}(f_1): f_1(x) \leq f_2(x)$
computational order

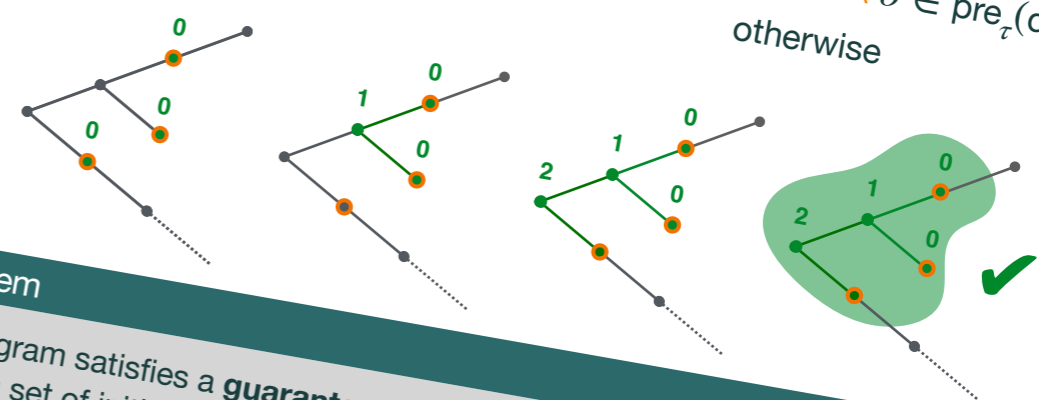


Theorem
A program must terminate for traces starting from a set of initial states \mathcal{I} if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_M)$

Guarantee Semantics

$$\mathcal{R}_G^\varphi \stackrel{\text{def}}{=} \text{lfp}_{\preceq} \bar{F}_G[\{\sigma \in \Sigma \mid \sigma \models \varphi\}]$$

$$\bar{F}_G[S]f \stackrel{\text{def}}{=} \lambda \sigma. \begin{cases} 0 & \sigma \in S \\ \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \notin S \wedge \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

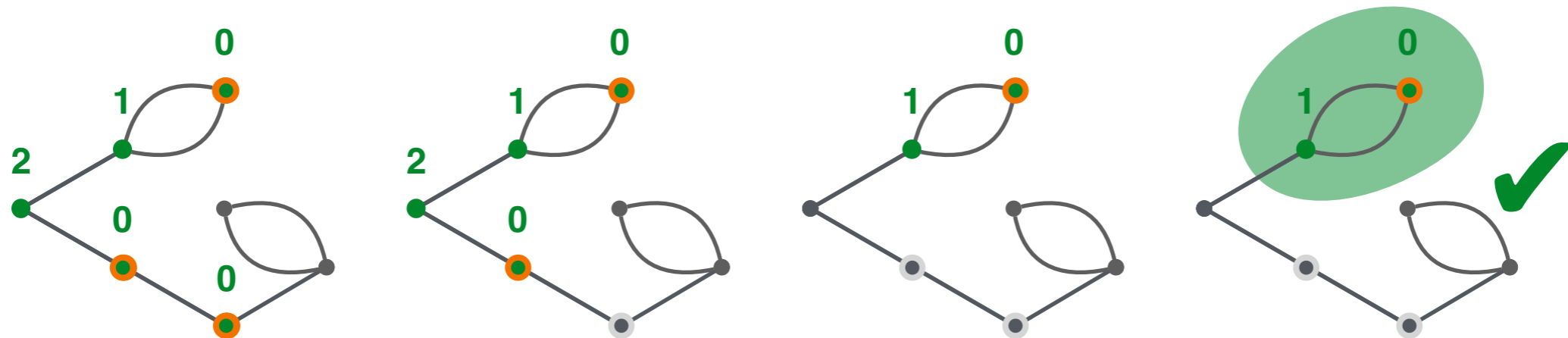


Theorem
A program satisfies a **guarantee property** $\text{AF } \varphi$ for traces starting from a set of initial states \mathcal{I} if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_G^\varphi)$

Recurrence Semantics

$$\mathcal{R}_R^\varphi \stackrel{\text{def}}{=} \text{gfp}_{\mathcal{R}_G^\varphi} \preceq \bar{F}_R$$

$$\bar{F}_R(f)\sigma \stackrel{\text{def}}{=} \begin{cases} f(s) & \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$



Theorem

A program satisfies a **recurrence property** $\text{AG AF } \varphi$ for traces starting from a set of initial states \mathcal{I} if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_R^\varphi)$

Abstract Recurrence Semantics

For each program instruction stat , we define $\mathcal{R}_G^{\varphi\#}[\text{stat}] : \mathcal{A} \rightarrow \mathcal{A}$:

- $\mathcal{R}_R^{\varphi\#}[\ell X \leftarrow e]t \stackrel{\text{def}}{=} \text{RESET}_A^R[\varphi](\overleftarrow{\text{ASSIGN}}_A[\ell X \leftarrow e]t)$
- $\mathcal{R}_R^{\varphi\#}[\text{if } \ell e \bowtie 0 \text{ then } s]t \stackrel{\text{def}}{=} \text{RESET}_A^R[\varphi](X)$
 where $X \stackrel{\text{def}}{=} \text{FILTER}_A[\ell e \bowtie 0](\mathcal{R}_G^{\varphi\#}[s]t) \vee_T \text{FILTER}_A[\ell e \bowtie 0]t$
- $\mathcal{R}_R^{\varphi\#}[\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]t \stackrel{\text{def}}{=} \text{gfp}_{G(t)}^{\#} \overline{F}_R^{\varphi\#}$
 where $G \stackrel{\text{def}}{=} \mathcal{R}_G^{\varphi\#}[\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]$
 $\overline{F}_R^{\varphi\#}(x) \stackrel{\text{def}}{=} \text{RESET}_A^R[\varphi](X)$
 $X \stackrel{\text{def}}{=} \text{FILTER}_A[\ell e \bowtie 0](\mathcal{R}_R^{\varphi\#}[s]x) \vee_T \text{FILTER}_A[\ell e \bowtie 0]t$
- $\mathcal{R}_R^{\varphi\#}[s_1; s_2]t \stackrel{\text{def}}{=} \mathcal{R}_R^{\varphi\#}[s_1](\mathcal{R}_R^{\varphi\#}[s_2]t)$

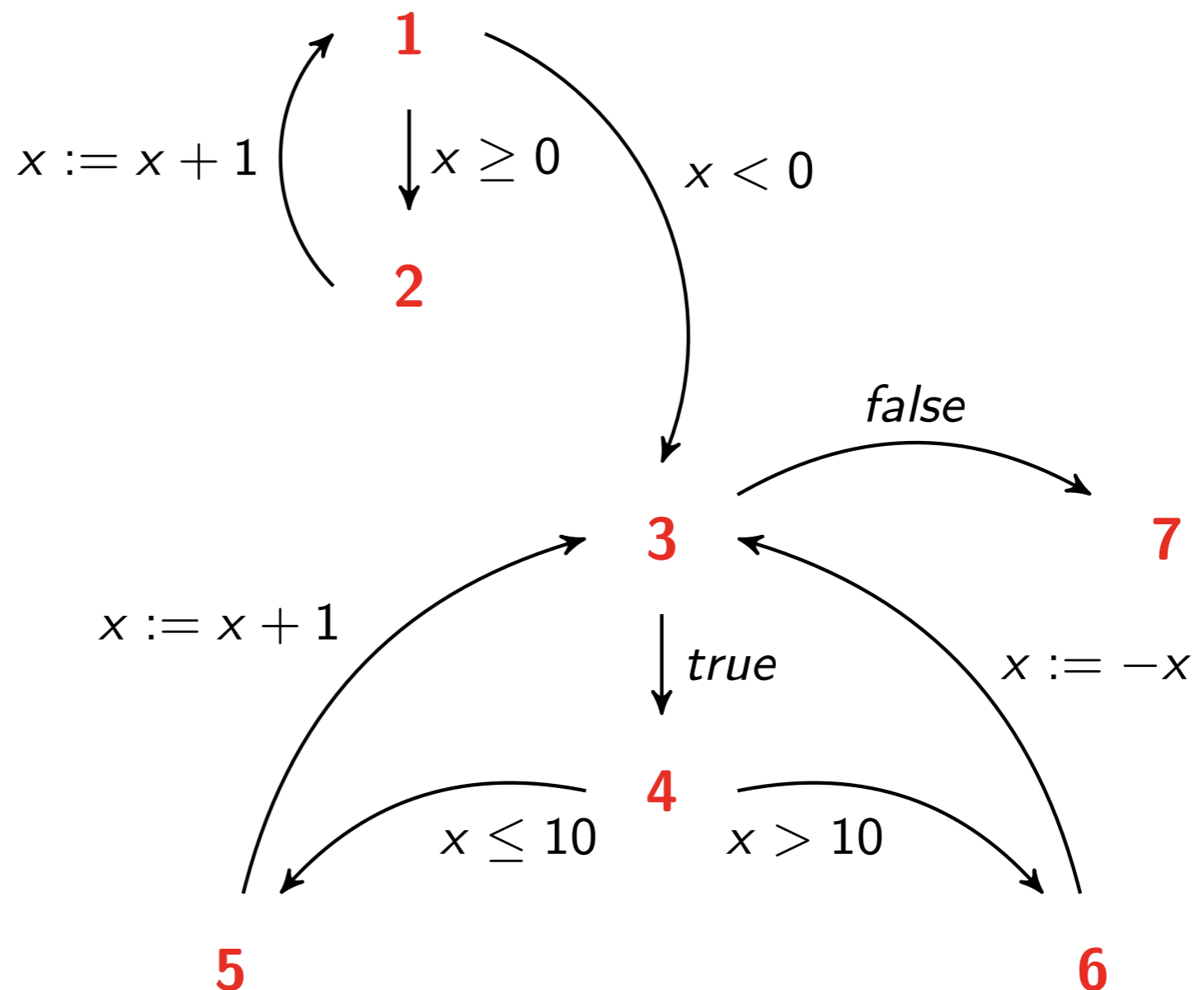
Abstract Recurrence Semantics

Example

```
int : x, y
while 1(x ≥ 0) do
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7
```

Property

AGAF (x = 3)



Abstract Recurrence Semantics

Example

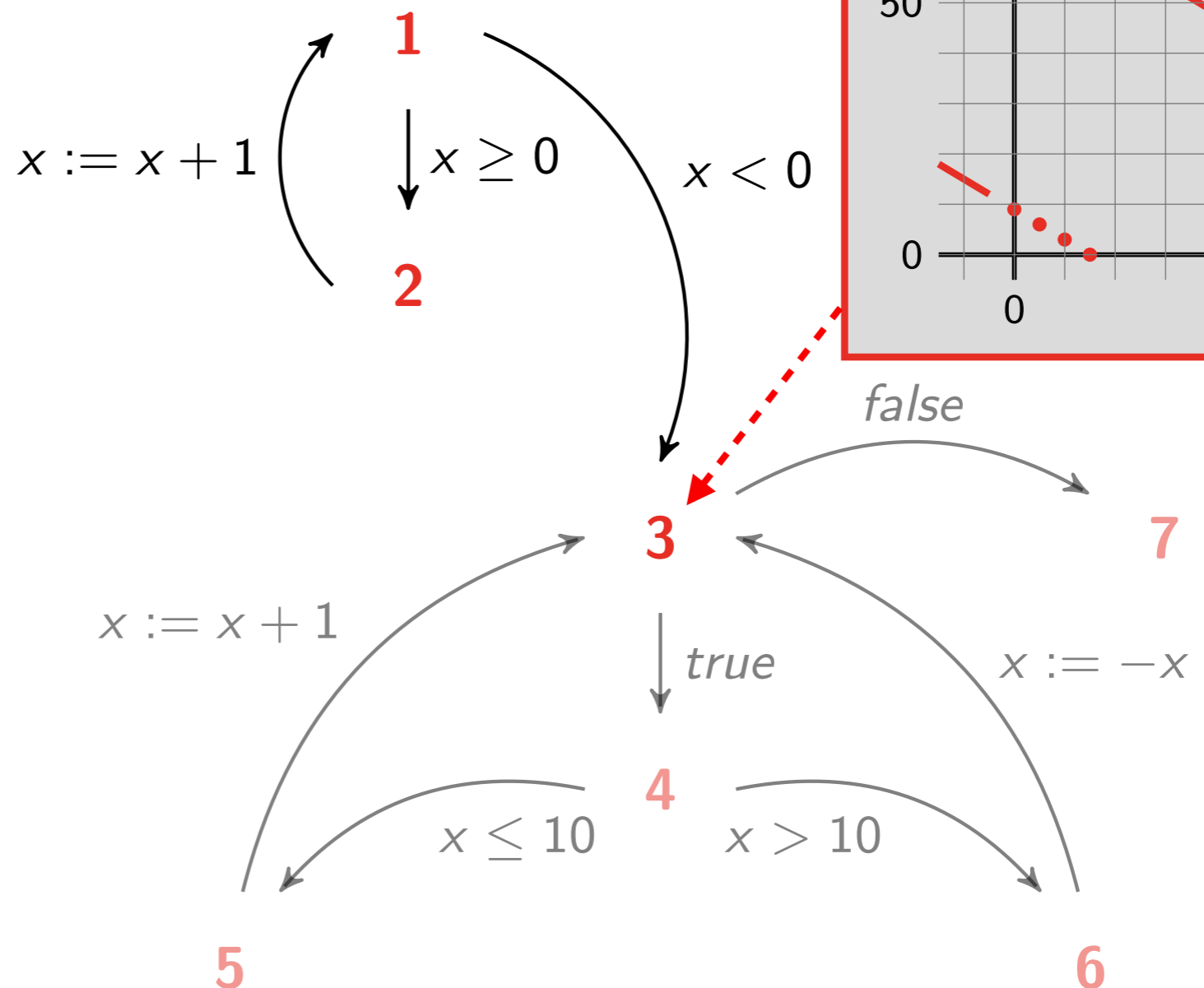
```

int : x, y
while 1(x ≥ 0) do
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```

Property

AGAF (x = 3)



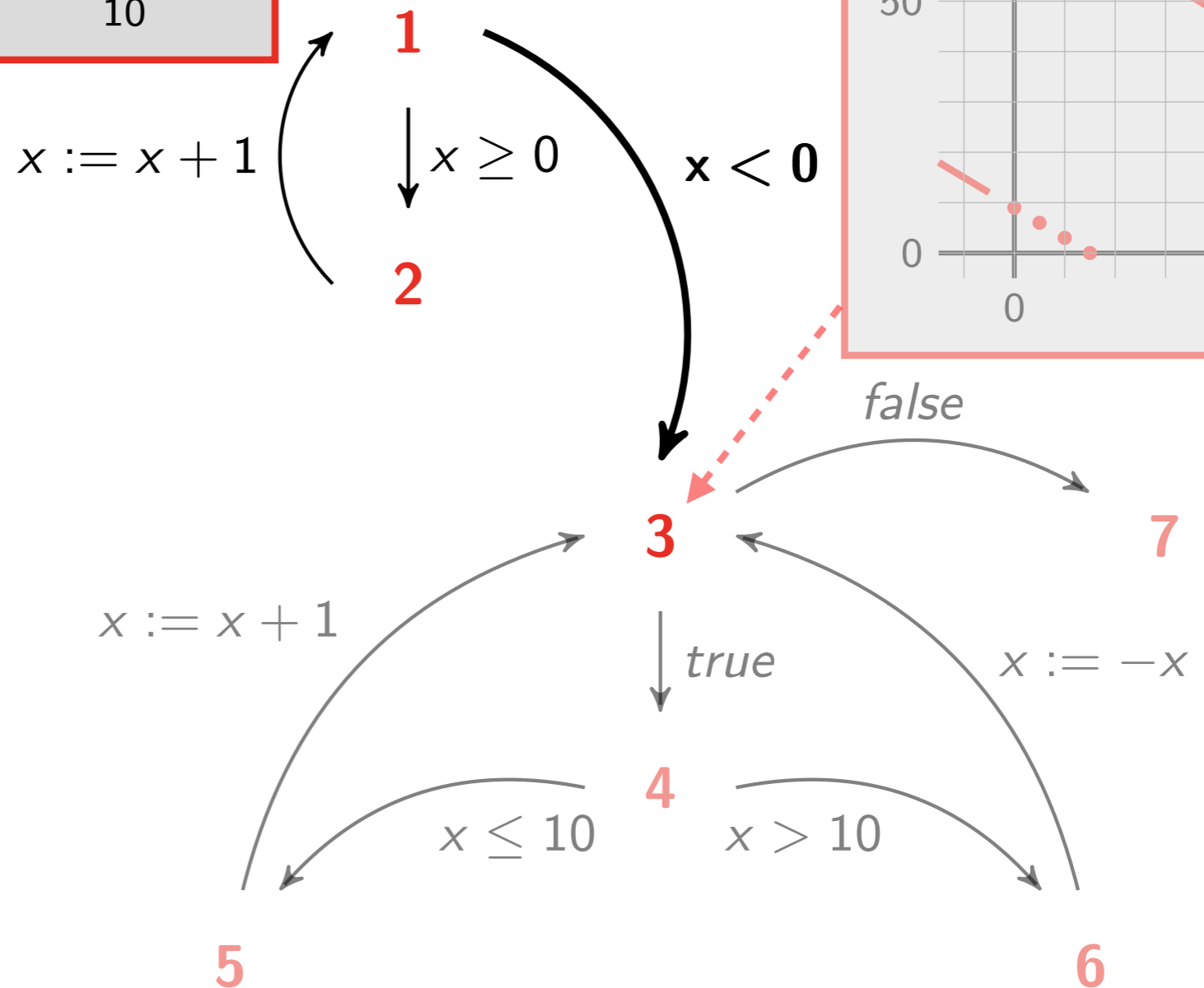
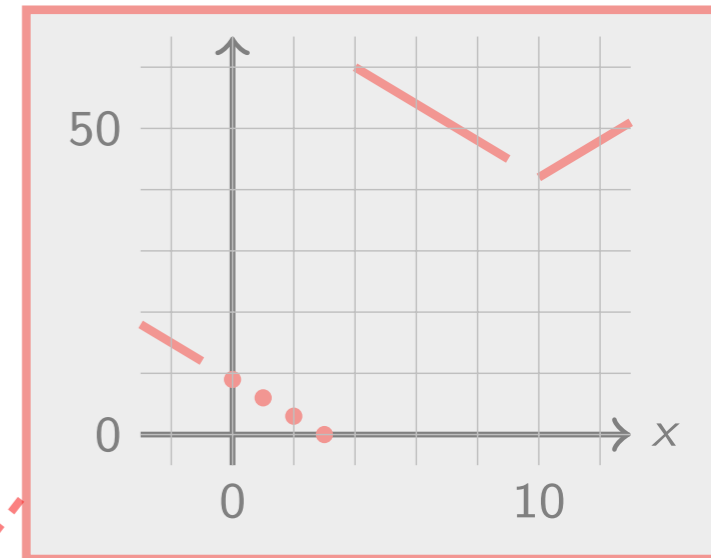
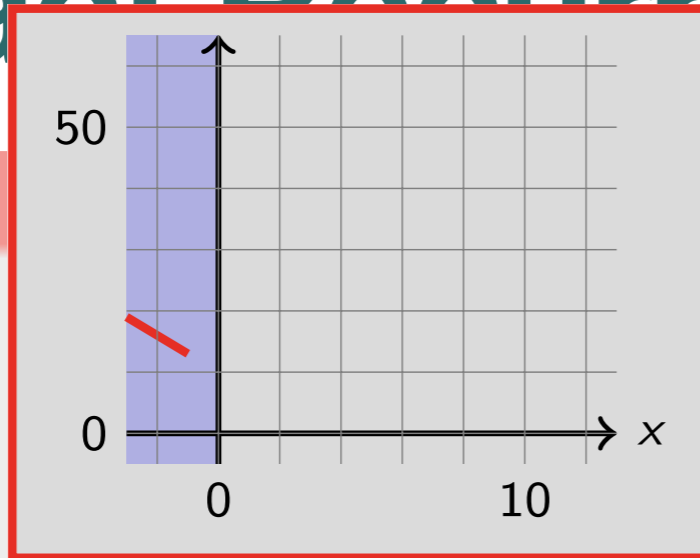
Abstract Recurrence Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



Property

AGAF ($x = 3$)

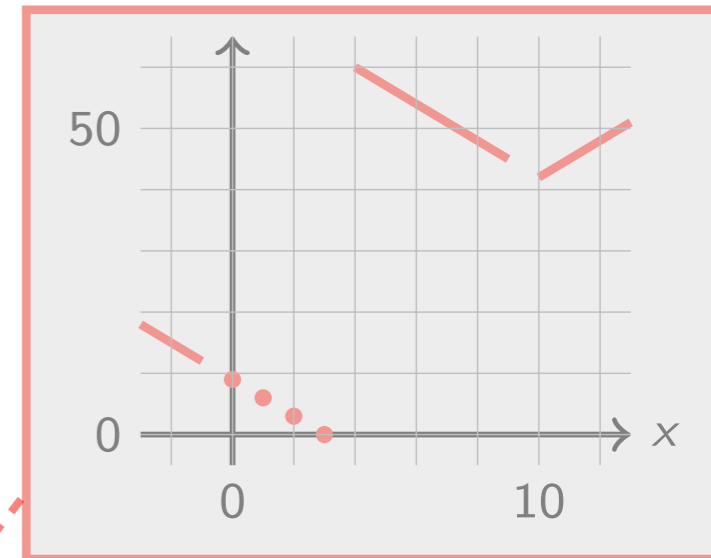
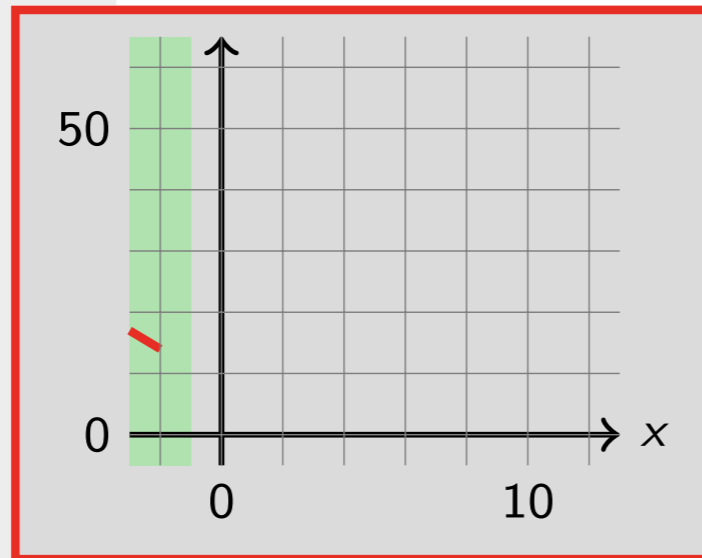
Abstract Recurrence Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



$x := x + 1$

1

$x \geq 0$

$x < 0$

2

false

3

7

true

$x := -x$

4

$x > 10$

$x \leq 10$

5

6

Property

AGAF ($x = 3$)

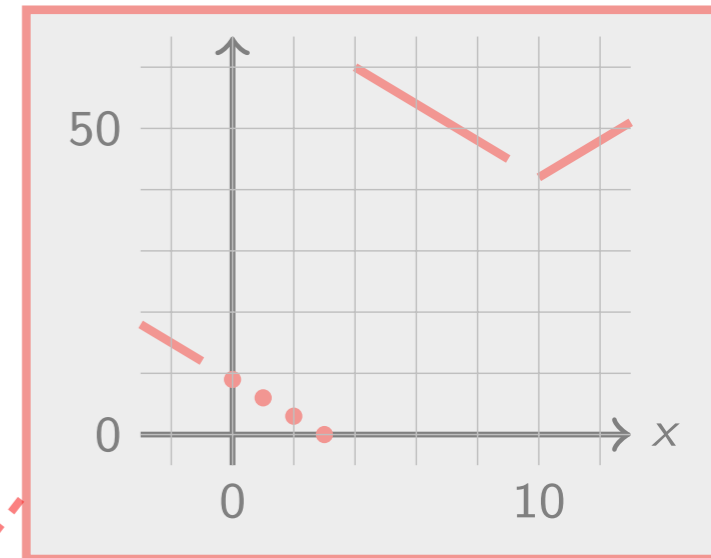
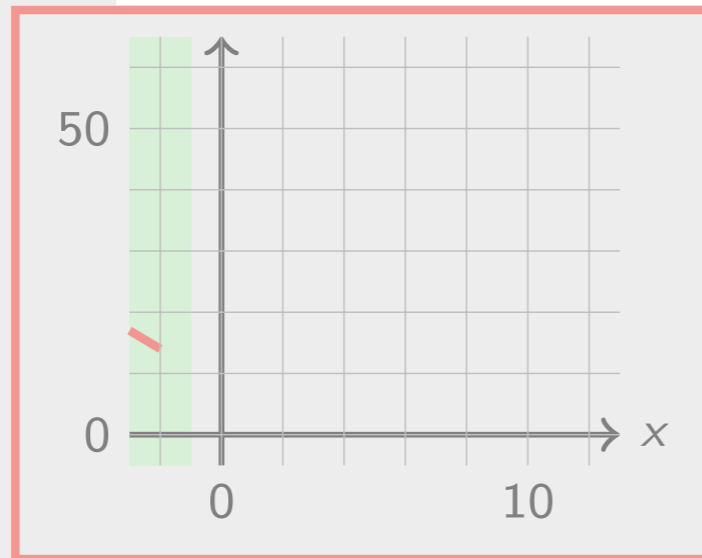
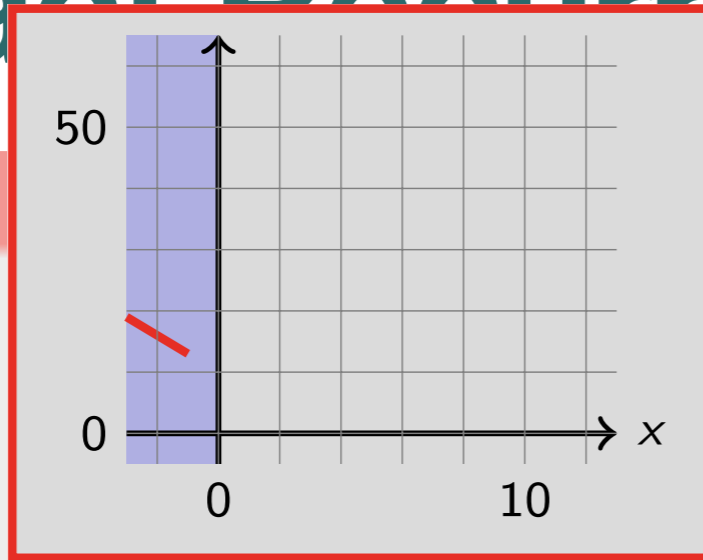
Abstract Recurrence Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



$x := x + 1$

$x \geq 0$

$x < 0$

false

true

$x := -x$

$x \leq 10$

$x > 10$

5

6

Property

AGAF ($x = 3$)

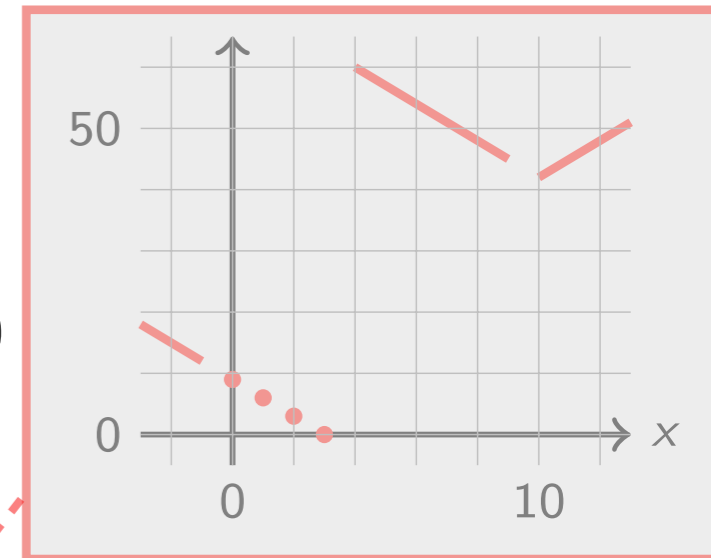
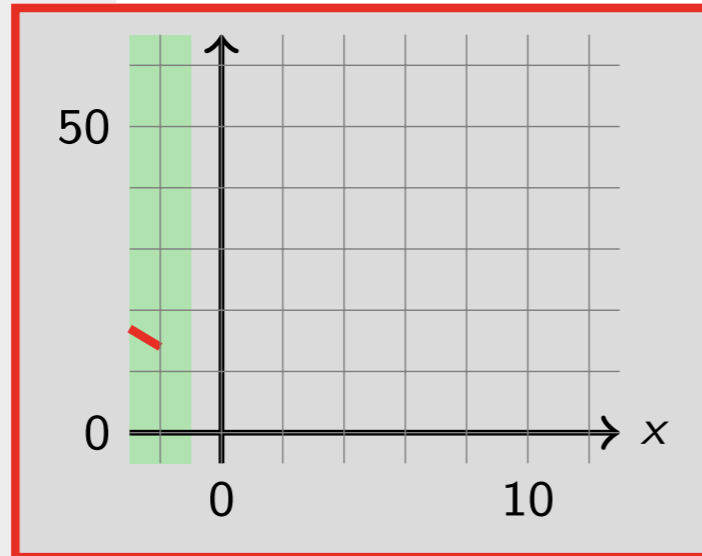
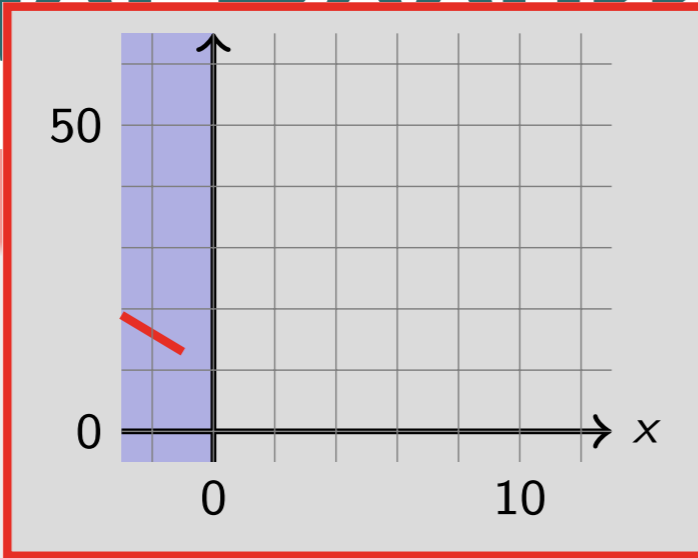
Abstract Recurrence Semantics

Example

```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



$x := x + 1$

1

$x \geq 0$

$x < 0$

2

false

3

7

true

$x := -x$

4

$x > 10$

$x \leq 10$

5

6

Property

$AGAF(x = 3)$

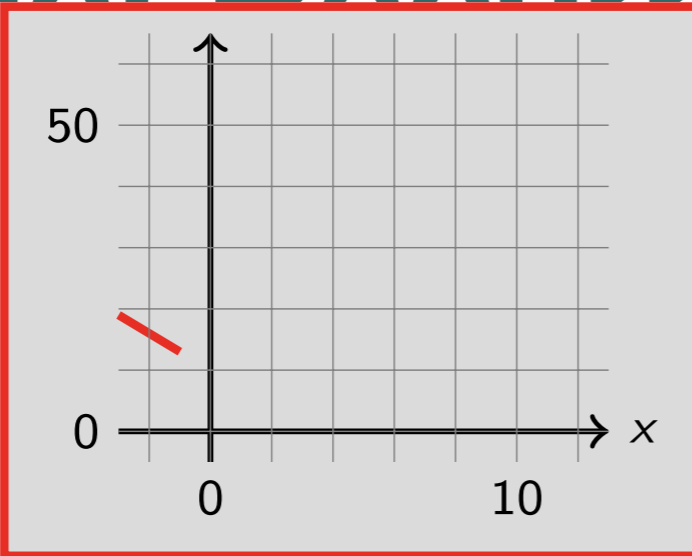
Abstract Recurrence Semantics

Example

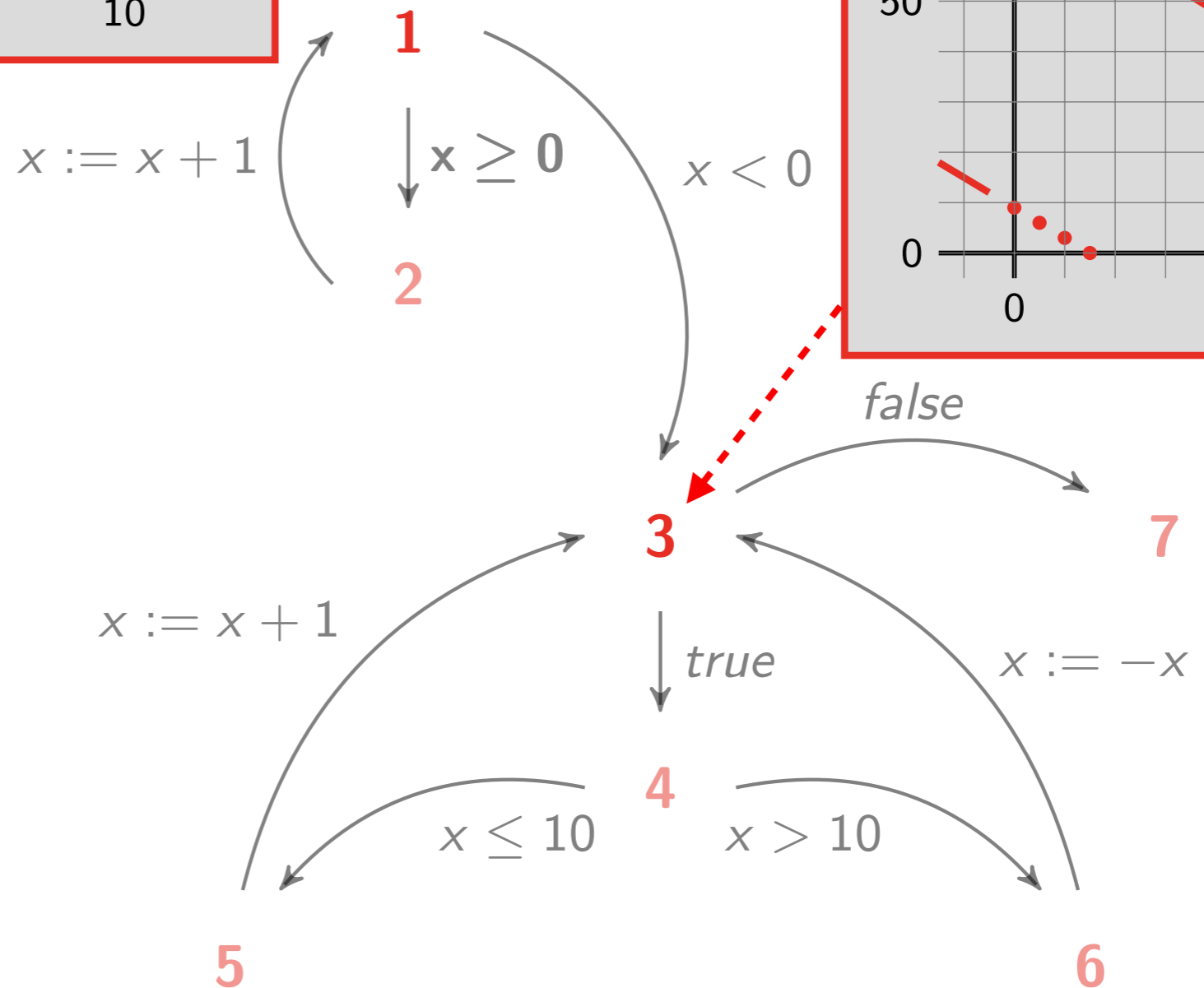
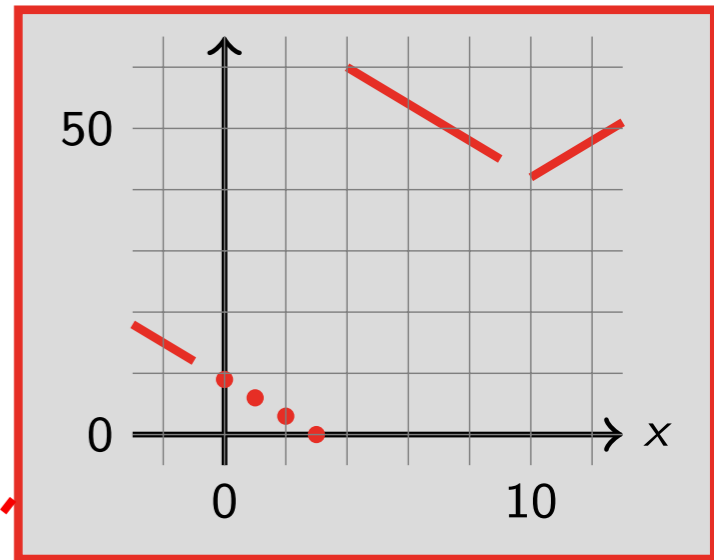
```

int : x, y
while 1(x ≥ 0)
  2x := x + 1
od
while 3( true ) do
  if 4( x ≤ 10 )
    5x := x + 1
  else
    6x := -x
  od7

```



the analysis gives $x < 0$ as **sufficient precondition**



Property

AGAF ($x = 3$)

Abstract Recurrence Semantics

Definition

The **abstract recurrence semantics** $\mathcal{R}_R^{\varphi\#}[\text{stat}^\ell] \in \mathcal{A}$ of a program stat^ℓ is:

$$\mathcal{R}_R^{\varphi\#}[\text{stat}^\ell] \stackrel{\text{def}}{=} \mathcal{R}_R^{\varphi\#}[\text{stat}](\text{LEAF: } \perp_F)$$

where $\mathcal{R}_R^{\varphi\#}[\text{stat}]: \mathcal{A} \rightarrow \mathcal{A}$ is the abstract recurrence semantics of each program instruction stat

Corollary (Soundness)

A program stat^ℓ satisfies a **recurrence property** $\text{AG AF } \varphi$ for traces starting from a set of initial states \mathcal{I} if $\mathcal{I} \subseteq \text{dom}(\gamma_A(\mathcal{R}_R^{\varphi\#}[\text{stat}^\ell]))$

Private < > github.com

Why GitHub? Team Enterprise Explore Marketplace Pricing Search / Sign in Sign up

caterinaurban / function Public Notifications Fork 2 Star 7

<> Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Code

caterinaurban no message bdeee1 on Aug 21, 2018 98 commits

banal	Changes according to feedback in pull-request:	5 years ago
cfgfrontend	- added loop detection to CFG based analysis	5 years ago
domains	no message	4 years ago
frontend	- added loop detection to CFG based analysis	5 years ago
main	added time measurements to CTL analysis	5 years ago
tests	more testcases with nestings of E/A	4 years ago
utils	Moved forward analysis code to distinct module ForwardIterator and	5 years ago
.gitignore	Renamed 'newfrontend' directory to 'cfgfrontend'	5 years ago
.merlin	Renamed 'newfrontend' directory to 'cfgfrontend'	5 years ago
.ocamlinit	added banal abstract domain source code	5 years ago
Makefile	- added loop detection to CFG based analysis	5 years ago
README.md	- added loop detection to CFG based analysis	5 years ago
pretty.py	Added CTL testcases	5 years ago
pretty_cfg.py	Implemented CFG based forward analysis	5 years ago

About
No description or website provided.

c static-analysis ocaml
termination abstract-interpretation
liveness

Readme
7 stars
1 watching
2 forks

Releases
No releases published

Packages
No packages published

Languages

Bibliography

[Urban15] **Caterina Urban**. Static Analysis by Abstract Interpretation of Functional Temporal Properties of Programs. PhD Thesis, École Normale Supérieure, 2015.

[Urban17] **Caterina Urban**, Antoine Miné. Inference of Ranking Functions for Proving Temporal Properties by Abstract Interpretation. In COMLAN, 2017.