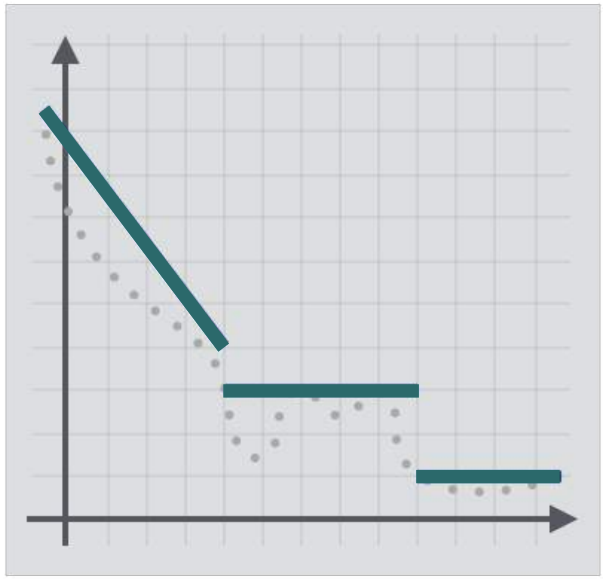
Liveness Analysis

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



Caterina Urban

December 7th, 2023

Year 2023-2024

Lesson 8

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

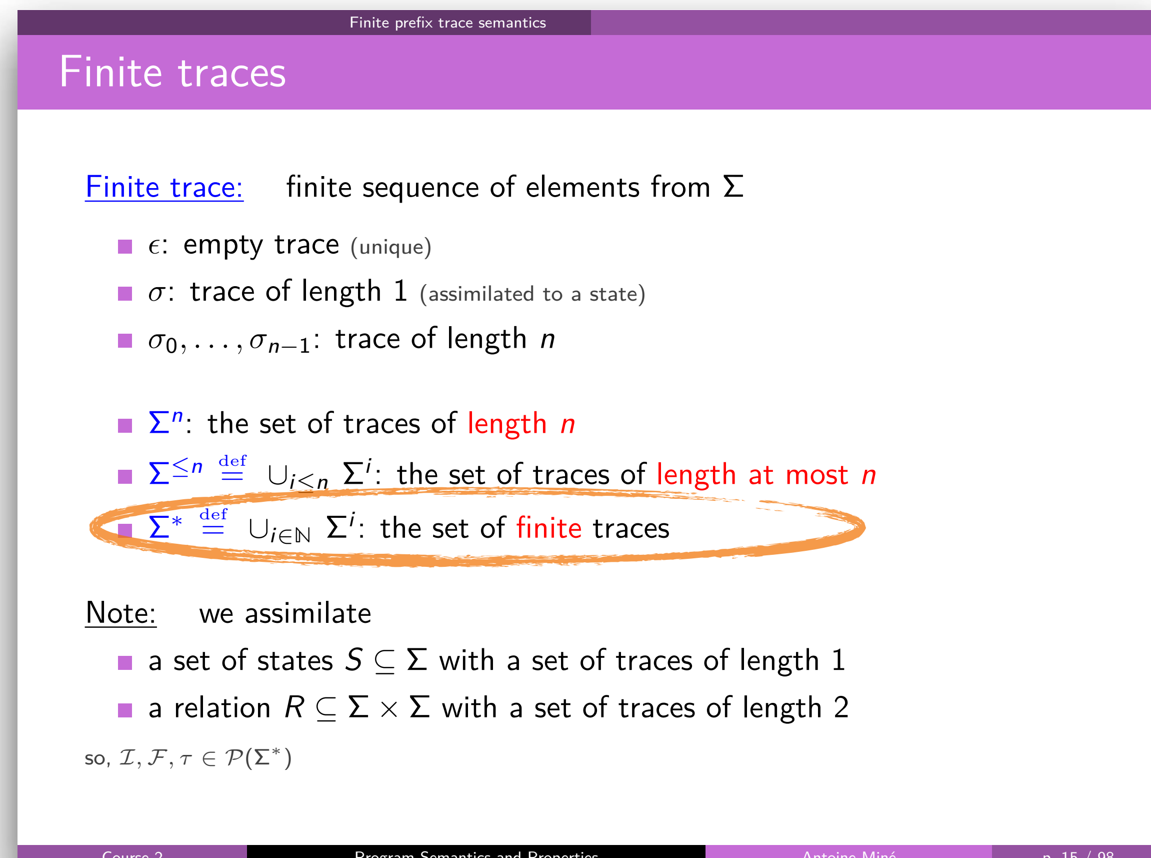
Potential and Definite Termination

Definition

A program with trace semantics $\mathscr{M} \in \mathscr{P}(\Sigma^\infty)$ **may terminate** if and only if $\mathscr{M} \cap \Sigma^* \neq \emptyset$

Definition

A program with trace semantics $\mathscr{M} \in \mathscr{P}(\Sigma^\infty)$ **must terminate** if and only if $\mathscr{M} \subseteq \Sigma^*$


Finite prefix trace semantics

Finite traces

Finite trace: finite sequence of elements from Σ

- ϵ: empty trace (unique)
- σ: trace of length 1 (assimilated to a state)
- $\sigma_0, \ldots, \sigma_{n-1}$: trace of length n

- Σ^n: the set of traces of length n
- $\Sigma^{\leq n} \stackrel{\text{def}}{=} \cup_{i<n} \Sigma^i$: the set of traces of length at most n
- $\Sigma^* \stackrel{\text{def}}{=} \cup_{i \in \mathbb{N}} \Sigma^i$: the set of finite traces

Note: we assimilate

- a set of states $S \subseteq \Sigma$ with a set of traces of length 1
- a relation $R \subseteq \Sigma \times \Sigma$ with a set of traces of length 2

so, $\mathcal{I}, \mathcal{F}, \tau \in \mathcal{P}(\Sigma^*)$

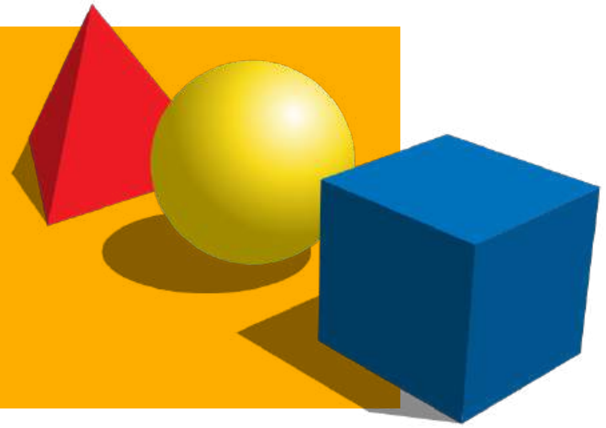
Course 2 | Program Semantics and Properties | Antoine Miné | p. 15 / 98

In *absence of non-determinism*, potential and definite termination coincide

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

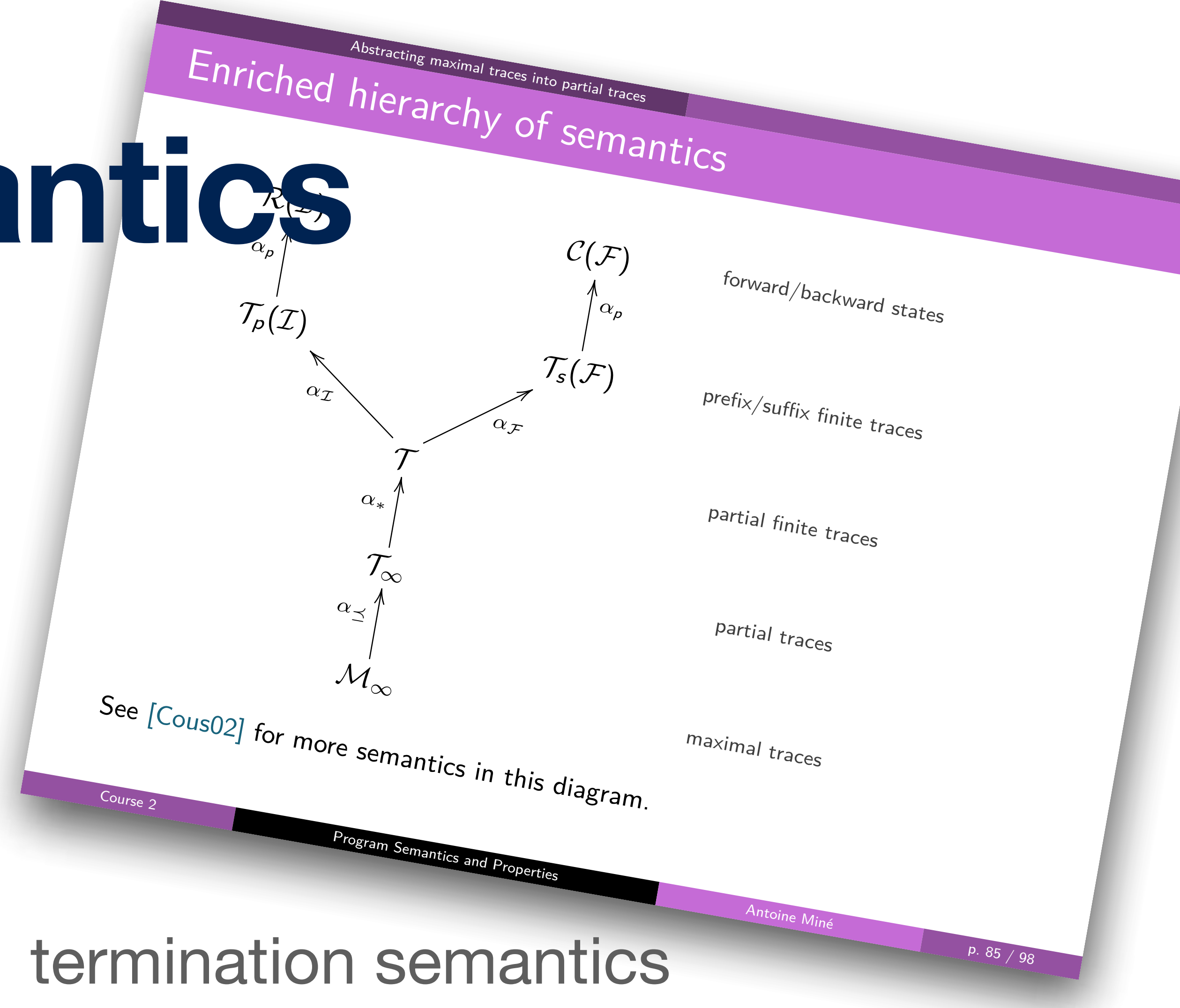
Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Hierarchy of Semantics



$$\begin{tikzcd}
\mathscr{R}_m & & \mathscr{R}_M \\
\mathscr{T}_m \arrow[u, "\alpha_m"] & & \mathscr{T}_M \arrow[u, "\overline{\alpha}_M"'] \\
& \mathscr{M}_\infty \arrow[ul, "\alpha_*"'] \arrow[ur, "\overline{\alpha}_*"] &
\end{tikzcd}$$

termination semantics

termination trace semantics

maximal trace semantics

Definite Termination Trace Semantics

Definite Termination Abstraction

$$\begin{tikzcd} \langle \mathscr{P}(\Sigma^\infty), \sqsubseteq \rangle \arrow[r, bend right, "\overline{\alpha}_*"'] & \langle \mathscr{P}(\Sigma^*), \subseteq \rangle \end{tikzcd}$$

$$\overline{\alpha}_*(T) \stackrel{\text{def}}{=} \{t \in T \cap \Sigma^* \mid \text{nhdb}(t, T \cap \Sigma^\omega) = \varnothing\}$$

$$\text{nhdb}(t, T) \stackrel{\text{def}}{=} \{t' \in T \mid \text{pf}(t) \cap \text{pf}(t') \neq \varnothing\}$$

$$\text{pf}(t) \stackrel{\text{def}}{=} \{t' \in \Sigma^\infty \setminus \{\epsilon\} \mid \exists t'' \in \Sigma^\infty \colon t = t' \cdot t''\}$$

Example:
$\alpha_*(\{ab, aba, bb, ba^\omega\}) = \{ab, aba\}$ since $\text{pf}(bb) \cap \text{pf}(ba^\omega) = \{b\} \neq \varnothing$

Definite Termination Trace Semantics

Tarskian Fixpoint Transfer

- $\langle \mathscr{P}(\Sigma^\infty), \sqsubseteq, \sqcup, \sqcap, \Sigma^\omega, \Sigma^* \rangle$
- $\mathscr{M}_\infty \stackrel{\text{def}}{=} \text{lfp}^{\sqsubseteq} F_s$
  $F_s(T) \stackrel{\text{def}}{=} \mathscr{B} \cup \tau \frown T$
- $\langle \mathscr{P}(\Sigma^*), \subseteq, \cup, \cap, \varnothing, \Sigma^* \rangle$
- $\overline{\alpha}_*: \mathscr{P}(\Sigma^\infty) \rightarrow \mathscr{P}(\Sigma^*)$

$$\mathscr{T}_M \stackrel{\text{def}}{=} \overline{\alpha}_*(\mathscr{M}_\infty) = \text{lfp}^{\subseteq} \overline{F}_*$$

$$\overline{F}_*(T) \stackrel{\text{def}}{=} \mathscr{B} \cup ((\tau \frown T) \cap (\Sigma^+ \setminus (\tau \frown (\Sigma^+ \setminus T))))$$

Theorem

Let $\langle C, \leq, \vee, \wedge, \perp, \top \rangle$ and $\langle A, \sqsubseteq, \sqcup, \sqcap, \perp^\#, \top^\# \rangle$ be complete lattices, let $f: C \rightarrow C$ and $f^\#: A \rightarrow A$ be monotonic functions, and let $\alpha: C \rightarrow A$ be <u>an abstraction function that is a complete $\wedge$-morphism</u> $(\forall S \subseteq C: f(\wedge S) = \sqcap \{f(s) \mid s \in S\})$ and that satisfies $f^\# \circ \alpha \sqsubseteq \alpha \circ f$ and the post-fixpoint correspondence $\forall a^\# \in A: f^\#(a^\#) \sqsubseteq a^\# \Rightarrow \exists a \in C: f(a) \leq d \wedge \alpha(a) = a^\#$ (i.e., each abstract post-fixpoint of $f^\#$ is the abstraction by α of some concrete post-fixpoint of f). Then, we have the fixpoint abstraction $\alpha(\text{lfp}^{\leq} f) = \text{lfp}^{\sqsubseteq} f^\#$.

(see proof in [Cousot02])

Definite Termination Semantics

Ranking Abstraction

count execution steps backwards

$\langle \mathscr{P}(\Sigma^*), \subseteq \rangle \xrightarrow{\alpha_m} \langle \Sigma \rightharpoonup \mathbb{O}, \preceq \rangle$

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

$$\overline{\alpha}_M(T) \overset{\text{def}}{=} \overline{\alpha}_V(\overrightarrow{\alpha}(T))$$

$$\overline{\alpha}_V(\emptyset) \overset{\text{def}}{=} \dot{\emptyset}$$

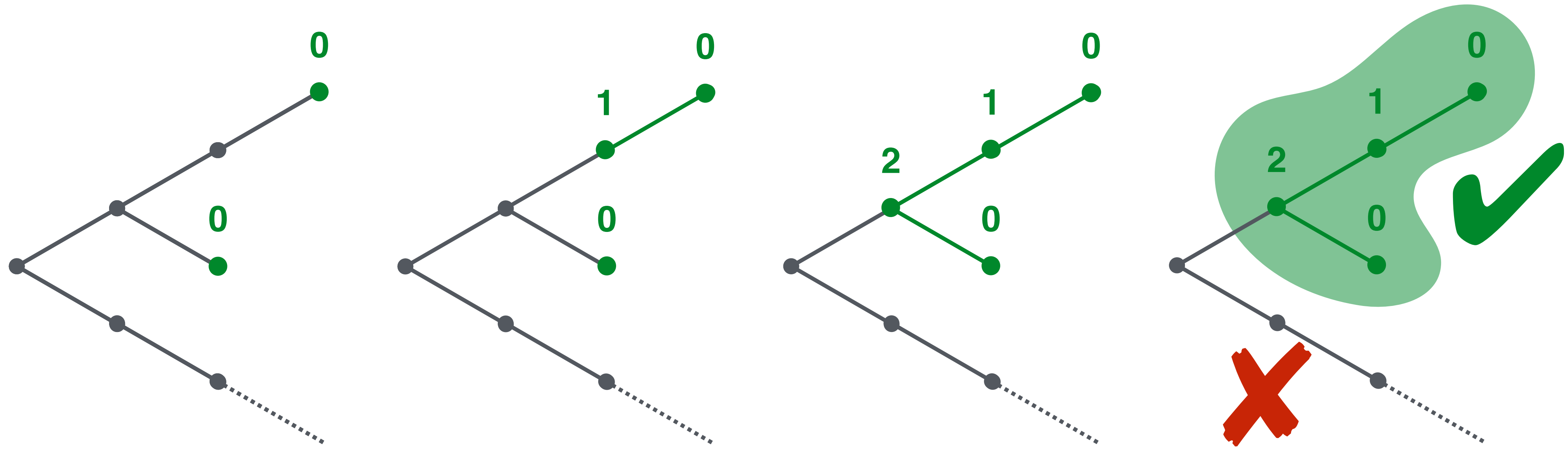
$$\overline{\alpha}_V(r)\sigma \overset{\text{def}}{=} \begin{cases} 0 & \forall \sigma' \in \Sigma: (\sigma, \sigma') \notin r \\ \text{sup}\{\overline{\alpha}_V(r)\sigma' + 1 \mid \sigma' \in \text{dom}(\overline{\alpha}_V(r)) \wedge (\sigma, \sigma') \in r\} & \text{otherwise} \end{cases}$$

$$\overrightarrow{\alpha}(T) \overset{\text{def}}{=} \{(\sigma, \sigma') \in \Sigma \times \Sigma \mid \exists t \in \Sigma^*, t' \in \Sigma^\infty: t\sigma\sigma't' \in T\}$$

Definite Termination Semantics

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

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



Theorem

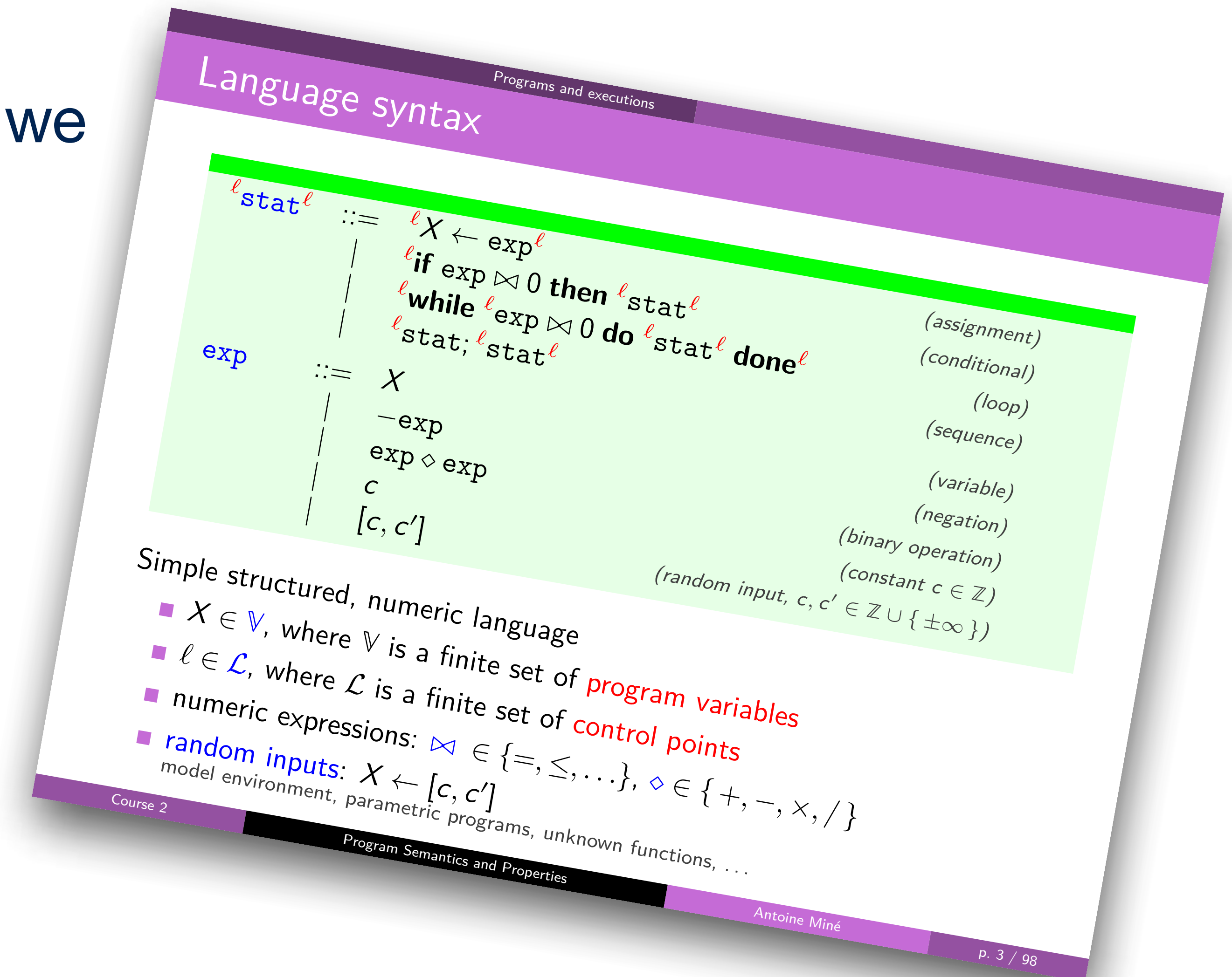
A program **must terminate** for traces starting from a set of initial states $\mathcal{I}$ if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_M)$

Denotational Definite Termination Semantics

We define the definite termination semantics $\mathscr{R}_M \colon \Sigma \rightharpoonup \mathbb{O}$ by partitioning with respect to the program control points, i.e., $\mathscr{R}_M \colon \mathscr{L} \to (\mathscr{E} \rightharpoonup \mathbb{O})$.

Thus, for each program instruction **stat**, we define a transformer $\mathscr{R}_M[\![\mathsf{stat}]\!] \colon (\mathscr{E} \rightharpoonup \mathbb{O}) \to (\mathscr{E} \rightharpoonup \mathbb{O})$:

- $\mathscr{R}_M[\![{}^{\ell}X \leftarrow e]\!]$
- $\mathscr{R}_M[\![\mathbf{if}\ {}^{\ell}\ e \bowtie 0\ \mathbf{then}\ s]\!]$
- $\mathscr{R}_M[\![\mathbf{while}\ {}^{\ell}\ e \bowtie 0\ \mathbf{do}\ s\ \mathbf{done}]\!]$
- $\mathscr{R}_M[\![s_1; s_2]\!]$



Denotational Definite Termination Semantics

Definition

The **definite termination semantics** $\mathcal{R}_M[\![\mathsf{stat}^{\ell}]\!] : \mathcal{E} \rightharpoonup \mathbb{O}$ of a program stat^{ℓ} is:

$$\mathcal{R}_M[\![\mathsf{stat}^{\ell}]\!] \stackrel{\text{def}}{=} \mathcal{R}_M[\![\mathsf{stat}]\!](\lambda\rho.0)$$

where $\mathcal{R}_M[\![\mathsf{stat}]\!] : (\mathcal{E} \rightharpoonup \mathbb{O}) \to (\mathcal{E} \rightharpoonup \mathbb{O})$ is the definite termination semantics of each program instruction **stat**

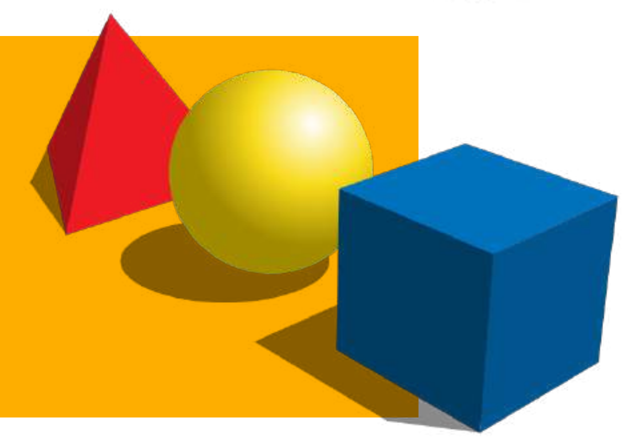
Theorem

A program stat^{ℓ} **must terminate** for traces starting from a set of initial states $\mathcal{I}$ if and only if $\mathcal{I} \subseteq \mathrm{dom}(\mathcal{R}_m[\![\mathsf{stat}^{\ell}]\!])$

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Concretization-Based Piecewise Abstraction

γ_A

$\mathcal{R}^{\#}_{M}[\![\mathsf{stat}^{\ell}]\!] \in \mathcal{A}$

$\langle \mathcal{E} \rightharpoonup \mathbb{O}, \preccurlyeq \rangle \quad \langle \mathcal{A}, \preccurlyeq_A \rangle$

$\mathcal{R}_M[\![\mathsf{stat}^{\ell}]\!] : \mathcal{E} \rightharpoonup \mathbb{O}$

$f_1 \preccurlyeq f_2 \overset{\text{def}}{=} \mathrm{dom}(f_1) \supseteq \mathrm{dom}(f_2) \wedge \forall x \in \mathrm{dom}(f_1) : f_1(x) \leq f_2(x)$

approximation order

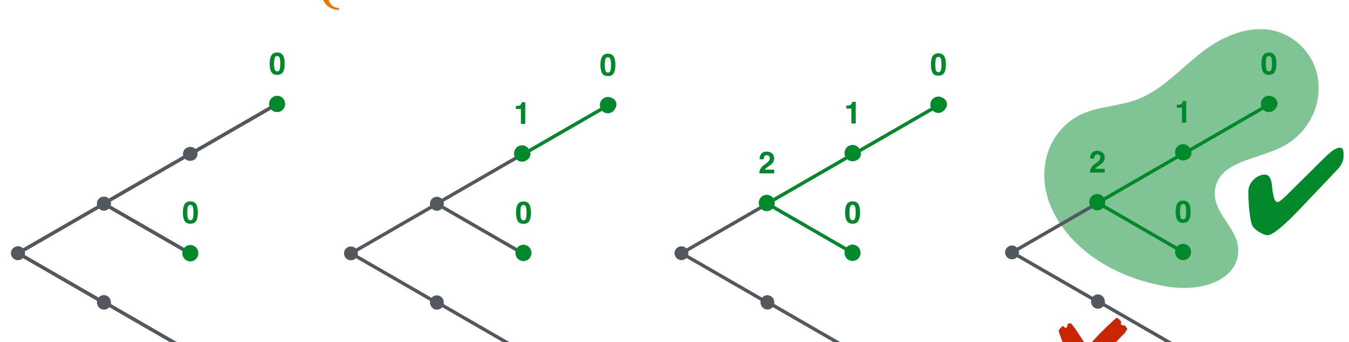
Definite Termination Semantics

$\mathcal{R}_M \overset{\text{def}}{=} \overline{\alpha}_M(\mathcal{T}_M) = \mathrm{lfp}^{\preceq}\, \overline{F}_M$

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

computational order

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



Concretization-Based Piecewise Abstraction

$\mathcal{R}^{\#}_{M}[\![\mathsf{stat}^{\ell}]\!] \in \mathcal{A}$

$$\langle \mathcal{E} \rightharpoonup \mathbb{O}, \preccurlyeq \rangle \xleftarrow{\gamma_A} \langle \mathcal{A}, \preccurlyeq_A \rangle$$

$\mathcal{R}_{M}[\![\mathsf{stat}^{\ell}]\!] \colon \mathcal{E} \rightharpoonup \mathbb{O}$

$f_1 \preccurlyeq f_2 \overset{\text{def}}{=} \text{dom}(f_1) \supseteq \text{dom}(f_2) \wedge \forall x \in \text{dom}(f_1) \colon f_1(x) \leq f_2(x)$

approximation order

By *pointwise lifiting* we obtain an abstraction $\mathcal{R}^{\#}_{M}$ of $\mathcal{R}_{M}$:

$$\langle \mathcal{L} \rightarrow (\mathcal{E} \rightharpoonup \mathbb{O}), \dot{\preccurlyeq} \rangle \xleftarrow{\dot{\gamma}_A} \langle \mathcal{L} \rightarrow \mathcal{A}, \dot{\preccurlyeq}_A \rangle$$

$\mathcal{R}_{M} \colon \mathcal{L} \rightarrow (\mathcal{E} \rightharpoonup \mathbb{O})$

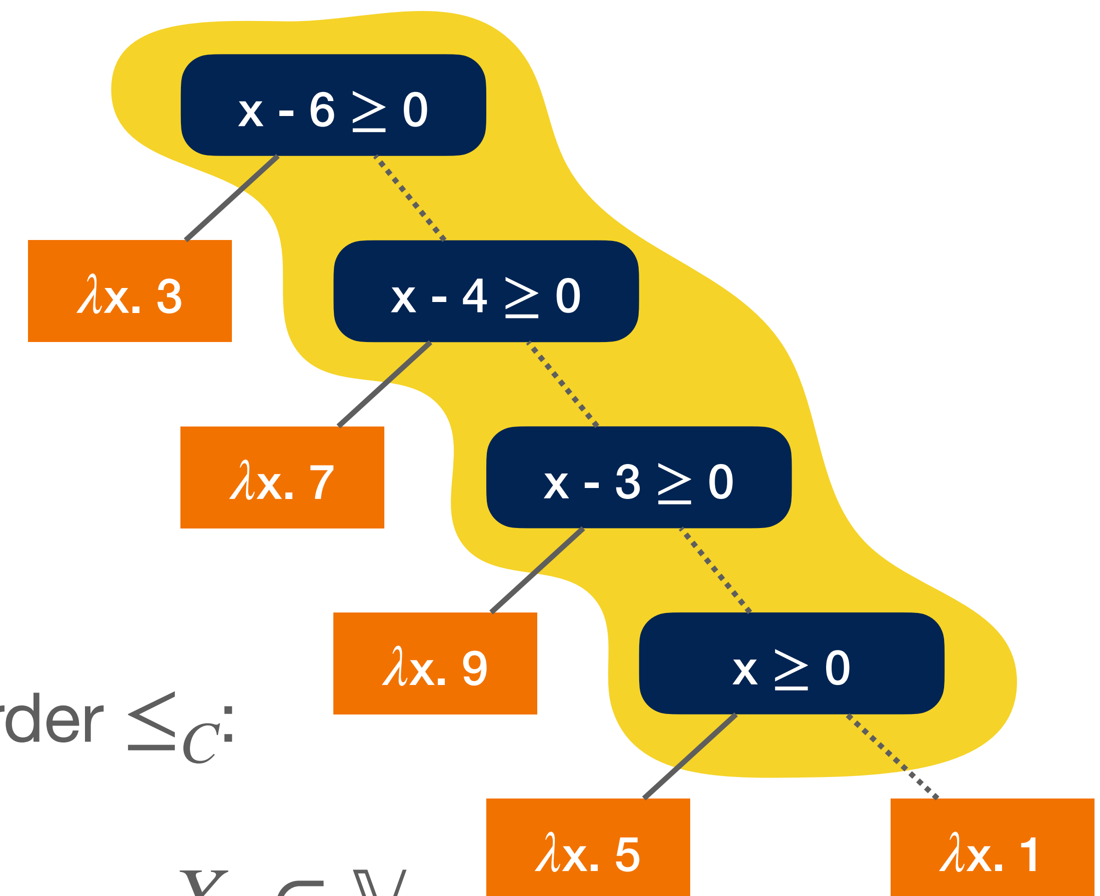
$\mathcal{R}^{\#}_{M} \colon \mathcal{L} \rightarrow \mathcal{A}$

Piecewise-Defined Ranking Functions Abstract Domain

Linear Constraints Auxiliary Abstract Domain

- Parameterized by an ***underlying numerical abstract domain*** $\langle \mathcal{D}, \sqsubseteq_D \rangle$ (i.e., intervals, octagons, or polyhedra):

$$\langle \mathcal{P}(\mathcal{C}/\equiv_C), \sqsubseteq_D \rangle \underset{\alpha_C}{\overset{\gamma_C}{\leftrightarrows}} \langle \mathcal{D}, \sqsubseteq_D \rangle$$



- $\mathcal{C}$ is a set of linear constraints *in canonical form*, equipped with a total order $\leq_C$:

$$\mathcal{C} \stackrel{\text{def}}{=} \{c_1 \cdot X_1 + c_k \cdot X_k + c_{k+1} \geq 0 \mid X_1, \ldots, X_k \in \mathbb{V} \\ \wedge c_1, \ldots, c_{k+1} \in \mathbb{Z} \wedge \text{gcd}(|c_1|, \ldots, |c_{k+1}|) = 1\}$$

Piecewise-Defined Ranking Functions Abstract Domain

Functions Auxiliary Abstract Domain

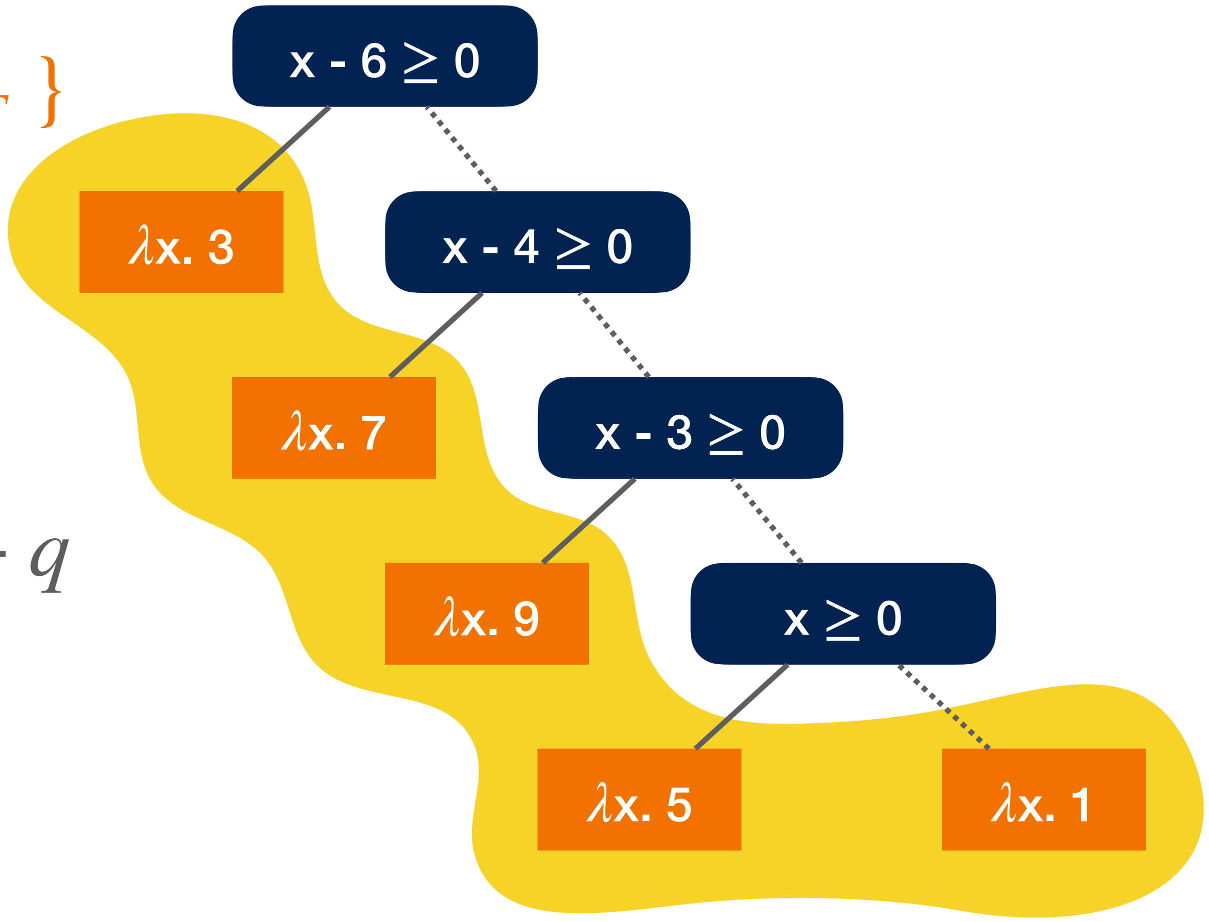
- Parameterized by an ***underlying numerical abstract domain*** $\langle \mathscr{D}, \sqsubseteq_D \rangle$
- $\mathscr{F} \stackrel{\text{def}}{=} \{ \perp_F \} \cup (\mathbb{Z}^{|\mathbb{V}|} \to \mathbb{N}) \cup \{ \top_F \}$

We consider **affine functions**:

$\mathscr{F}_A \stackrel{\text{def}}{=} \{ \perp_F \} \cup \{ f \colon \mathbb{Z}^{|\mathbb{V}|} \to \mathbb{N} \mid$

$$f(X_1, \ldots, X_k) = \sum_{i=1}^{k} m_i \cdot X_i + q$$

$\} \cup \{ \top_F \}$



Piecewise-Defined Ranking Functions Abstract Domain

- $\mathscr{A} \stackrel{\text{def}}{=} \{\text{LEAF}: f \mid f \in \mathscr{F}\} \cup \{\text{NODE}\{c\}: t_1; t_2 \mid c \in \mathscr{C} \wedge t_1, t_2 \in \mathscr{A}\}$

- **concretization function** $\gamma_A : \mathscr{A} \to (\mathscr{E} \rightharpoonup \mathbb{O})$:

$$\gamma_A(t) \stackrel{\text{def}}{=} \bar{\gamma}_A[\varnothing](t)$$

where $\bar{\gamma}_A : \mathscr{P}(\mathscr{C}/\equiv_C) \to \mathscr{A} \to (\mathscr{E} \rightharpoonup \mathbb{O})$:

$$\bar{\gamma}_A[C](\text{LEAF}: f) \stackrel{\text{def}}{=} \gamma_F[\alpha_C(C)](f)$$
$$\bar{\gamma}_A[C](\text{NODE}\{c\}: t_1; t_2) \stackrel{\text{def}}{=} \bar{\gamma}_A[C \cup \{c\}](t_1) \,\dot{\cup}\, \bar{\gamma}_A[C \cup \{\neg c\}](t_2)$$

and $\gamma_F : \mathscr{D} \to \mathscr{F} \to (\mathscr{E} \rightharpoonup \mathbb{O})$:

$$\gamma_F[D](\bot_F) \stackrel{\text{def}}{=} \dot{\varnothing}$$
$$\gamma_F[D](f) \stackrel{\text{def}}{=} \lambda\rho \in \gamma_D(D): f(\ldots, \rho(X_i), \ldots)$$
$$\gamma_F[D](\top_F) \stackrel{\text{def}}{=} \dot{\varnothing}$$

Piecewise-Defined Ranking Functions Abstract Domain

Abstract Domain Operators

- They manipulate elements in $\mathscr{A}_{\text{NIL}} \stackrel{\text{def}}{=} \{\text{NIL}\} \cup \mathscr{A}$
- The **binary operators** rely on a tree unification algorithm
  - approximation order $\preccurlyeq_A$ and computational order $\sqsubseteq_A$
  - approximation join $\curlyvee_A$ and computational join $\sqcup_A$
  - meet $\curlywedge_A$
  - widening $\triangledown_A$
- The **unary operators** rely on a tree pruning algorithm
  - assignment $\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]$
  - test $\text{FILTER}_A[\![e]\!]$

Piecewise-Defined Ranking Functions Abstract Domain

Widening

Goal: try to **predict** a valid ranking function

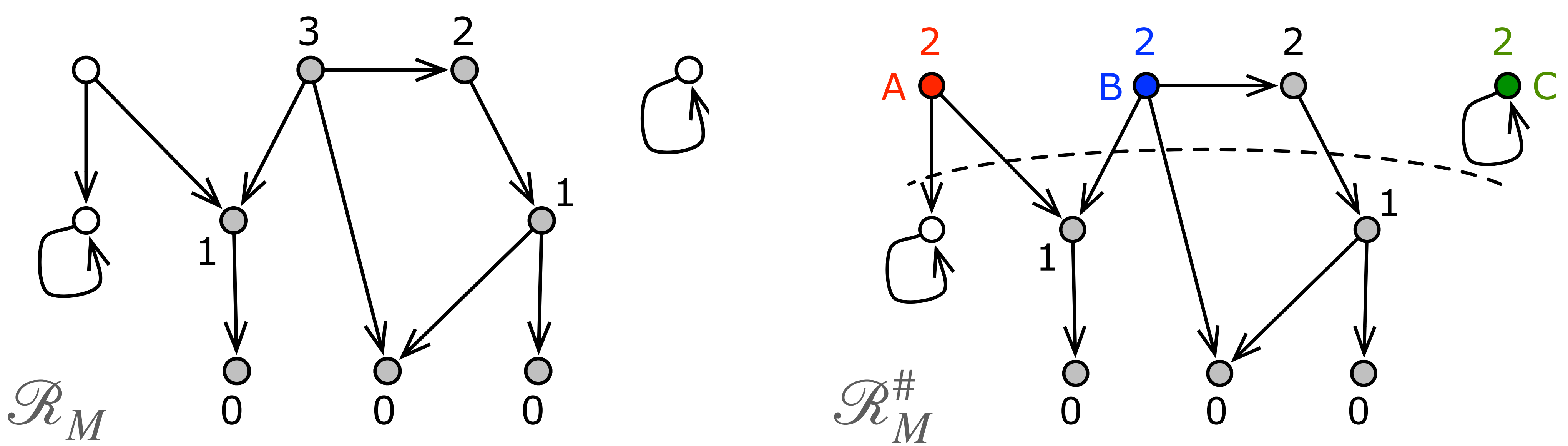
The prediction can (temporarily) be wrong!, i.e.,

- *under-approximates* the value of $\mathscr{R}_M$

and/or

- *over-approximates* the domain $\mathrm{dom}(\mathscr{R}_M)$ of $\mathscr{R}_M$

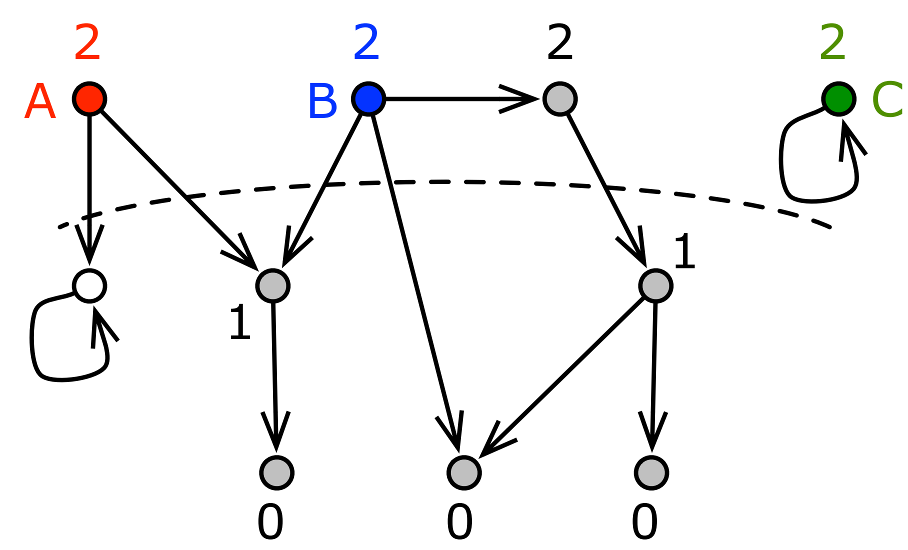
Example



Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

1. Check for **case A** (i.e., wrong domain predictions)
2. Perform **domain widening**
3. Check for **case B or C** (i.e., wrong value predictions)
4. Perform **value widening**



Abstract Definite Termination Semantics

Example

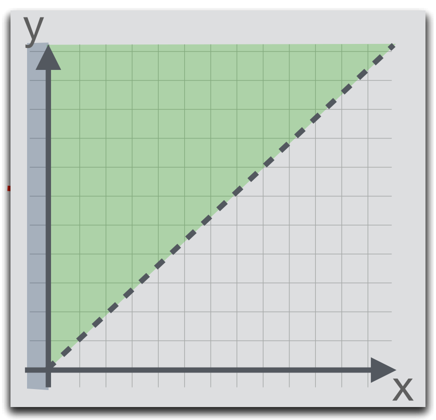
$^{1}x \leftarrow [-\infty, +\infty]$

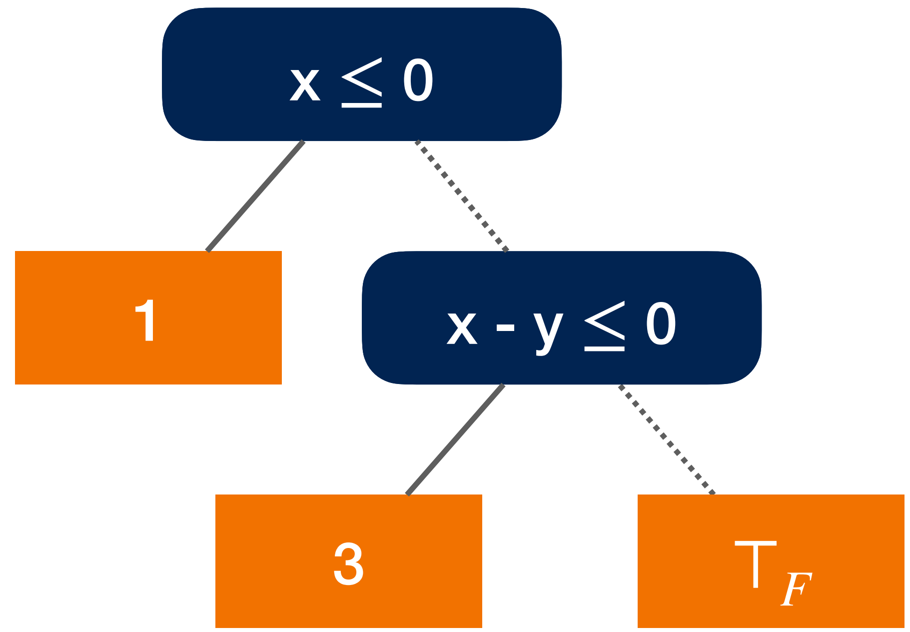
$^{2}y \leftarrow [-\infty, +\infty]$

while $^{3}(x > 0)$ **do**

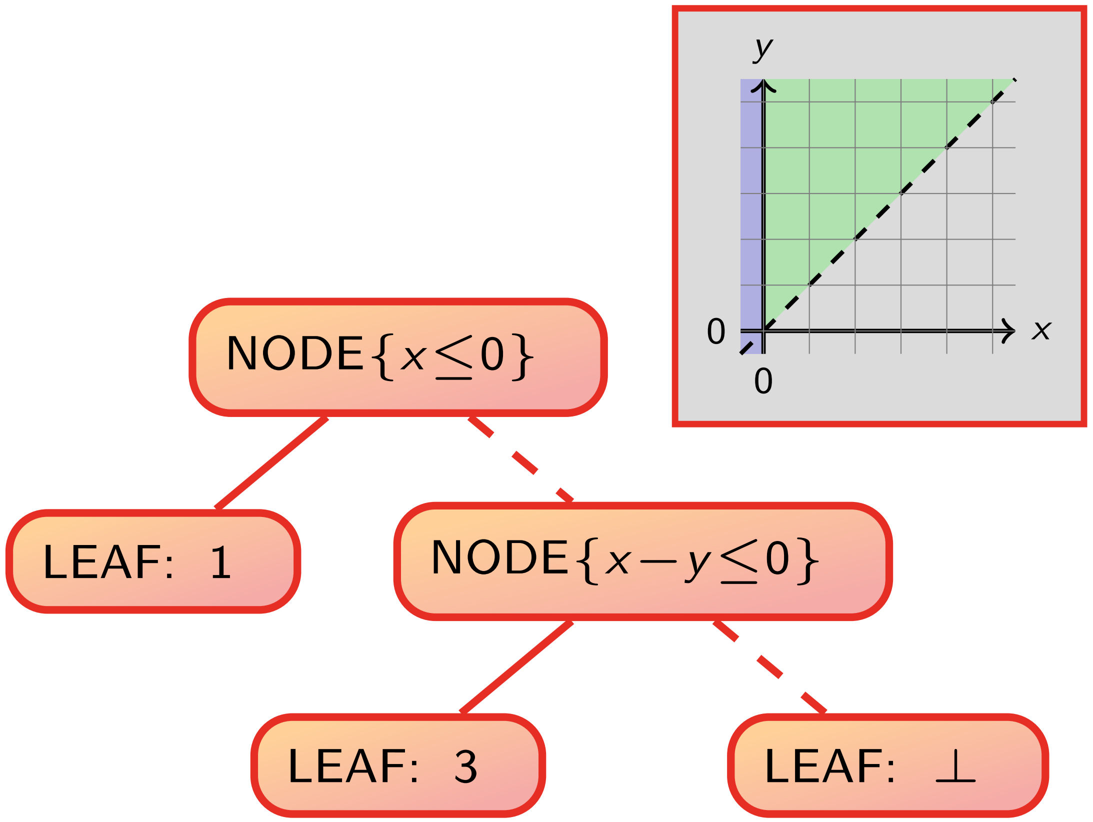
$\quad ^{4}x \leftarrow x - y$

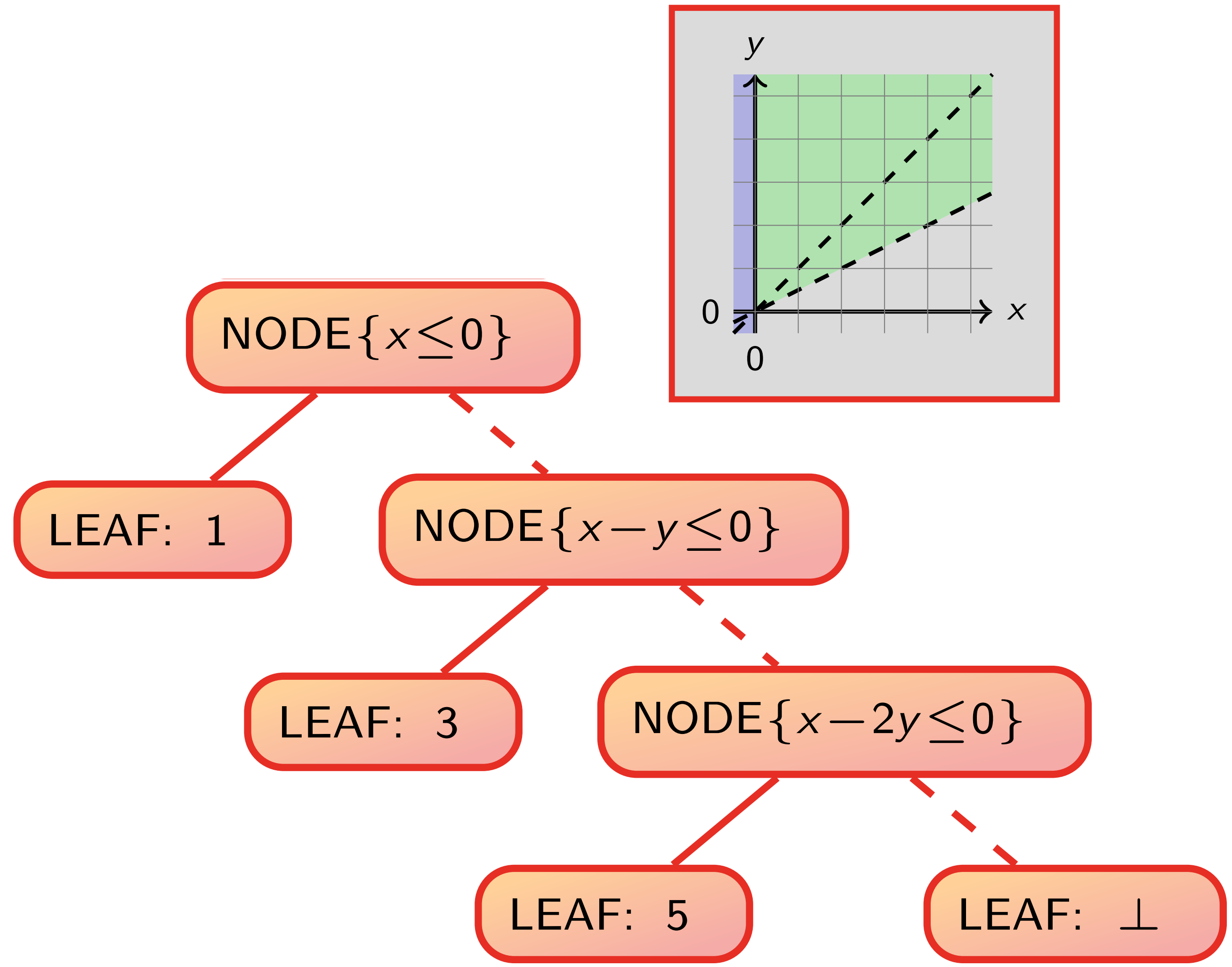
od^{5}

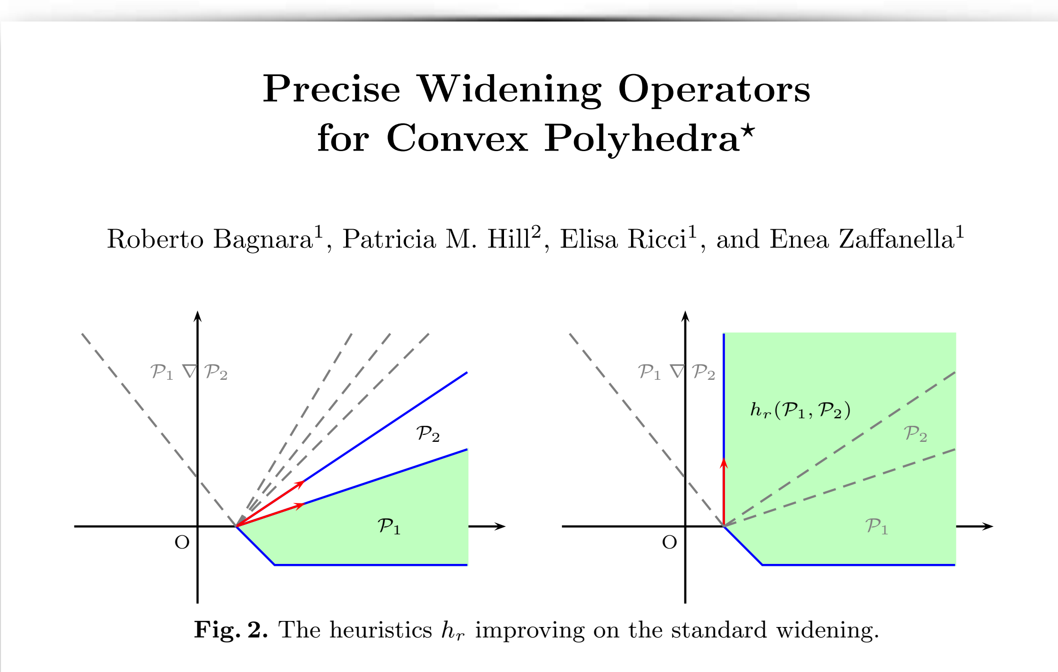




Better Widening



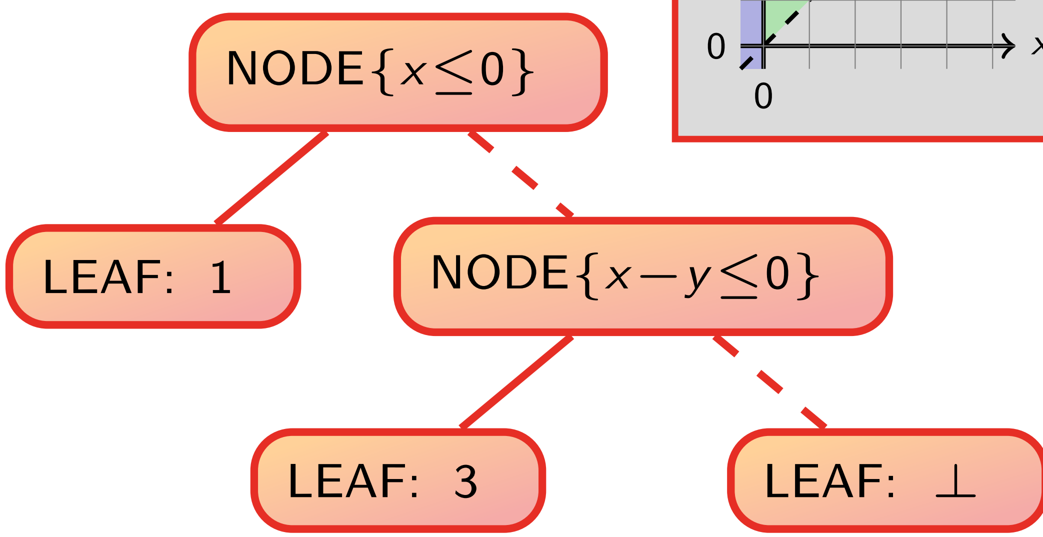


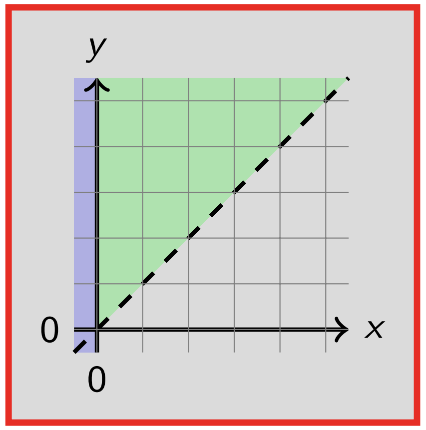

Precise Widening Operators
for Convex Polyhedra*

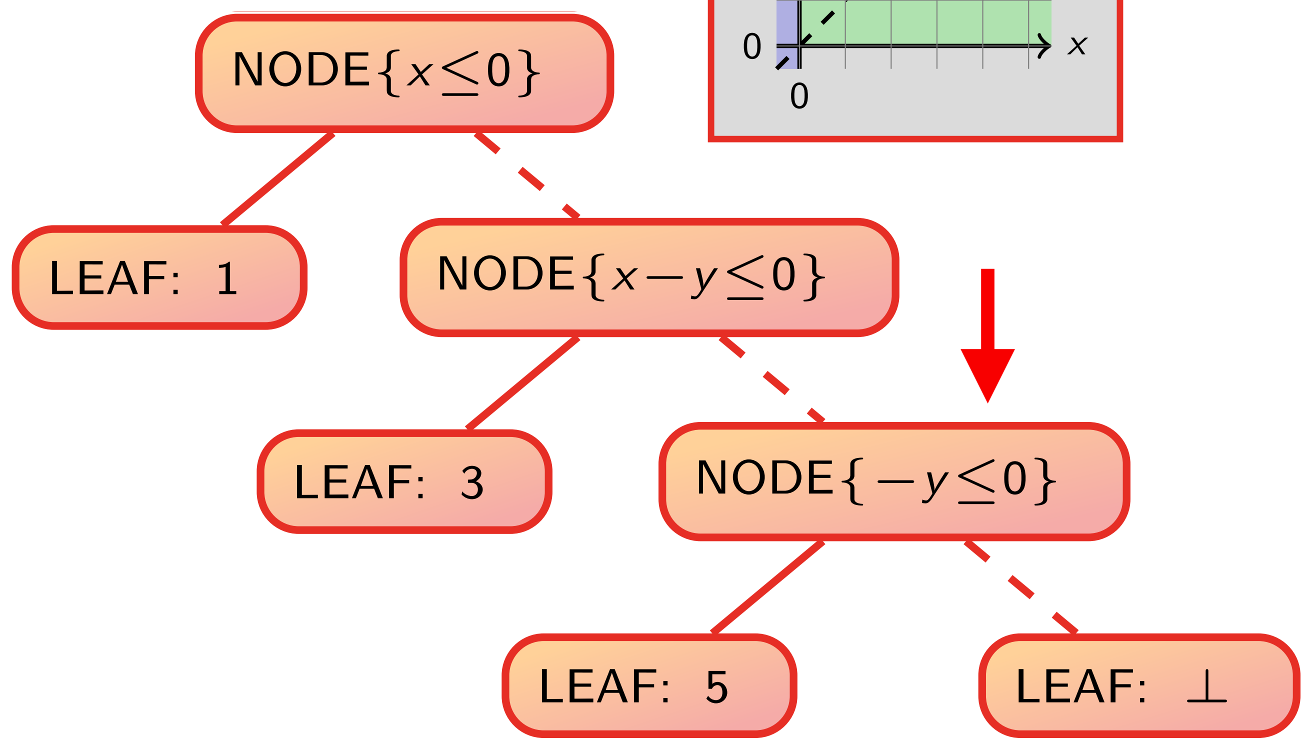
Roberto Bagnara[1], Patricia M. Hill[2], Elisa Ricci[1], and Enea Zaffanella[1]

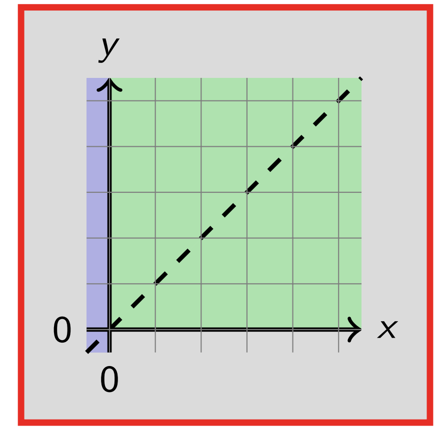
Fig. 2. The heuristics h_r improving on the standard widening.

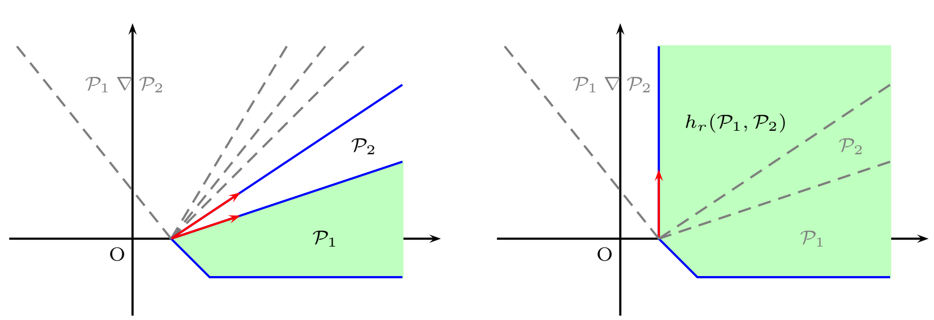
Better Widening









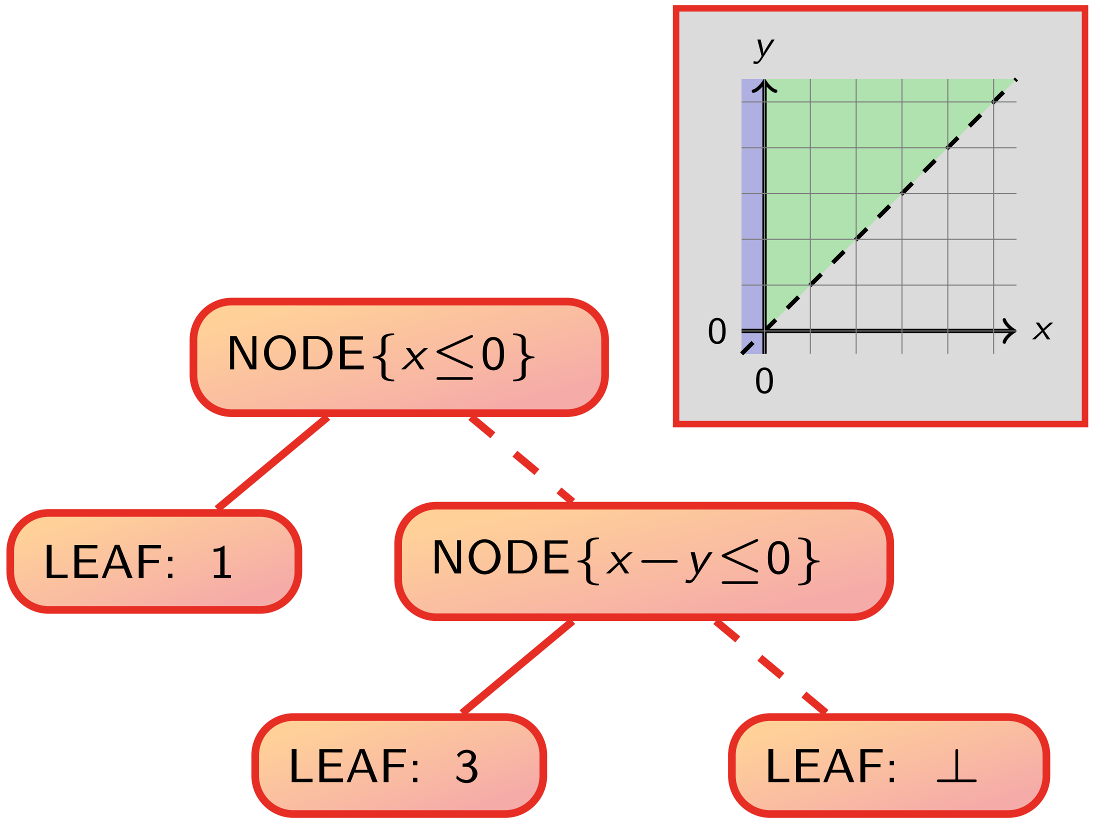


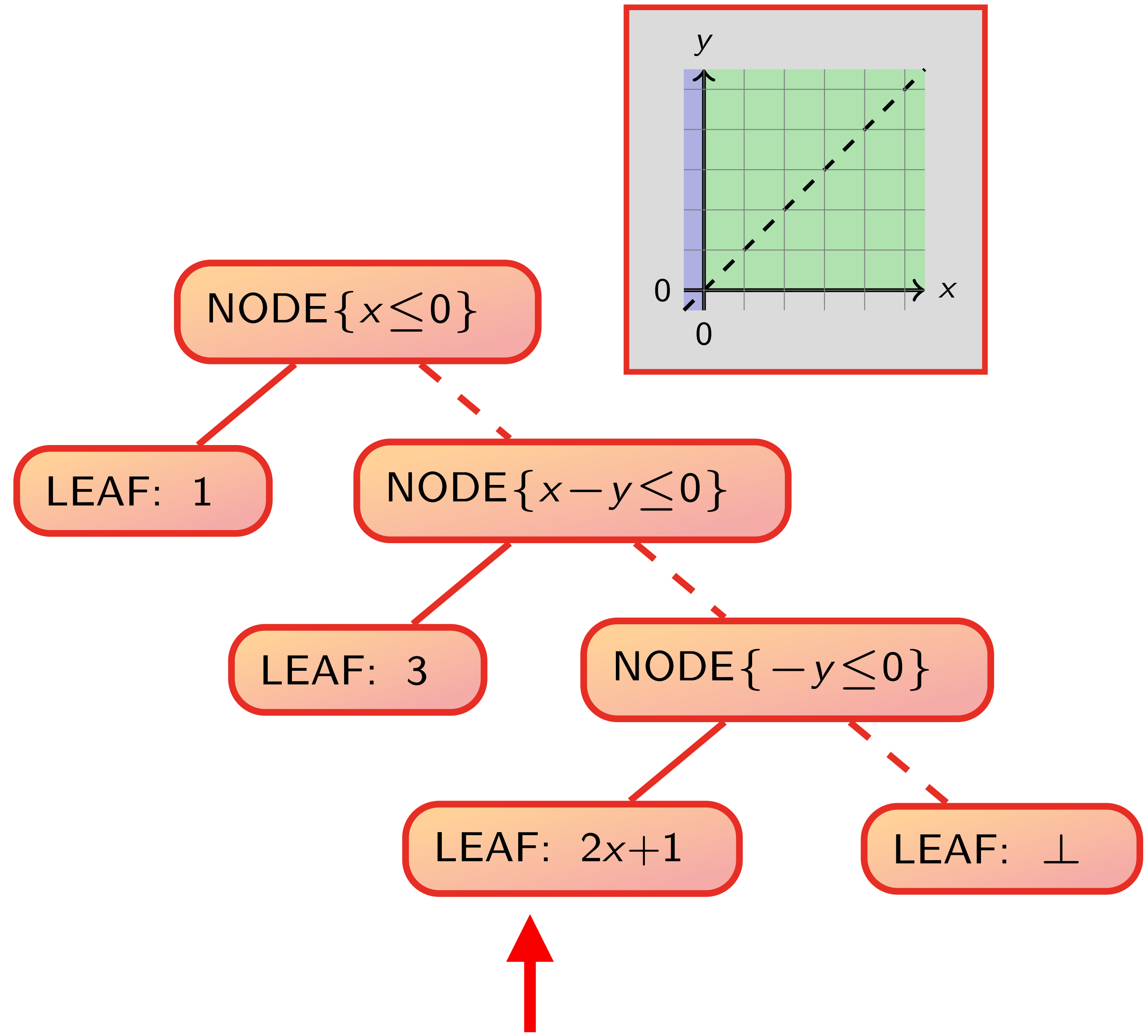
Precise Widening Operators
for Convex Polyhedra*

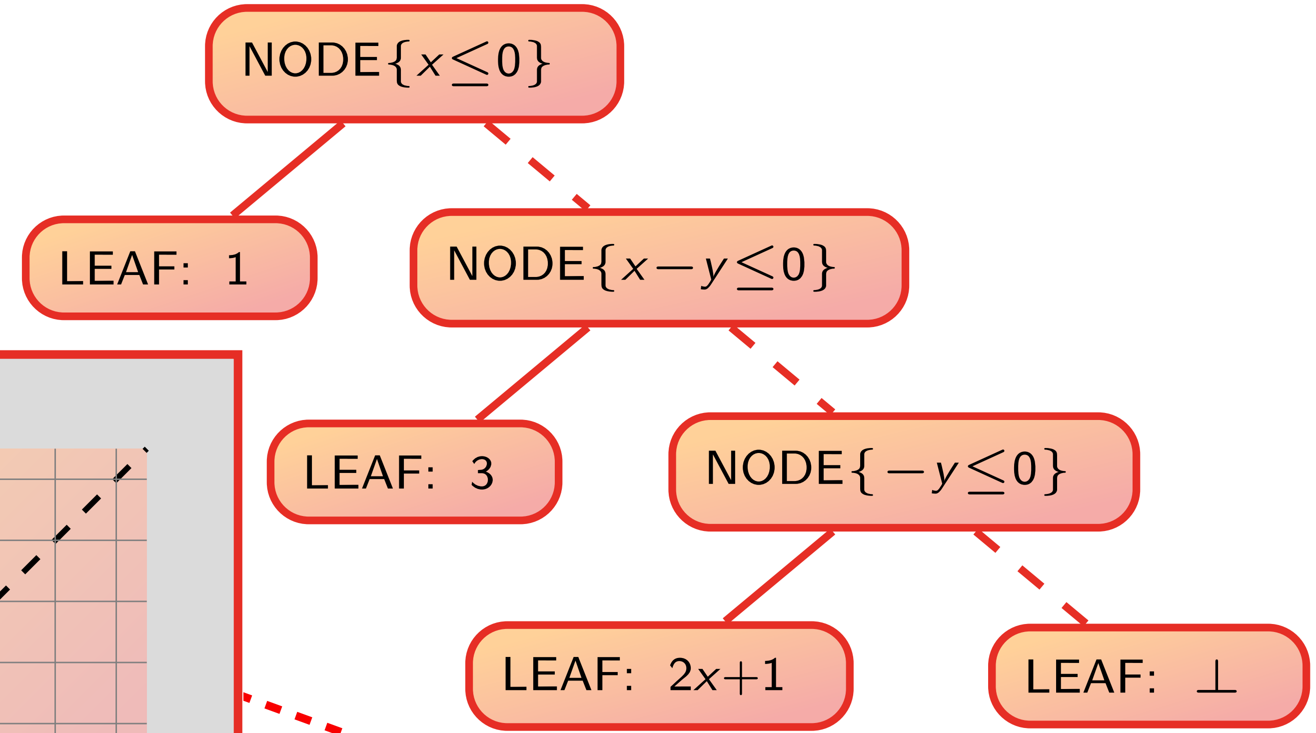
Roberto Bagnara[1], Patricia M. Hill[2], Elisa Ricci[1], and Enea Zaffanella[1]

Fig. 2. The heuristics h_r improving on the standard widening.

Better Widening







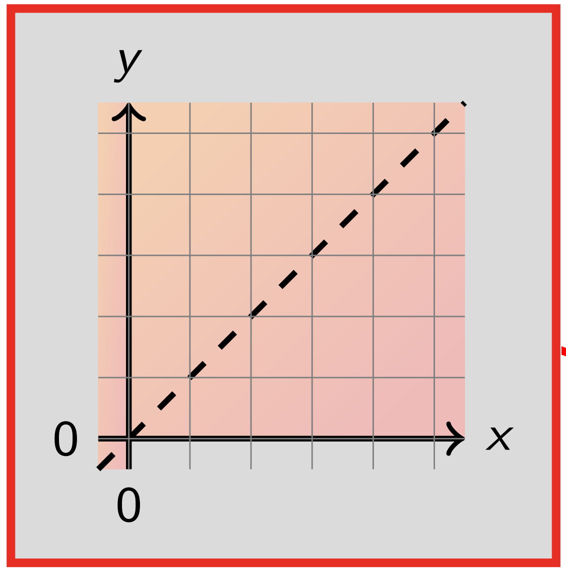
Example

int : x, y

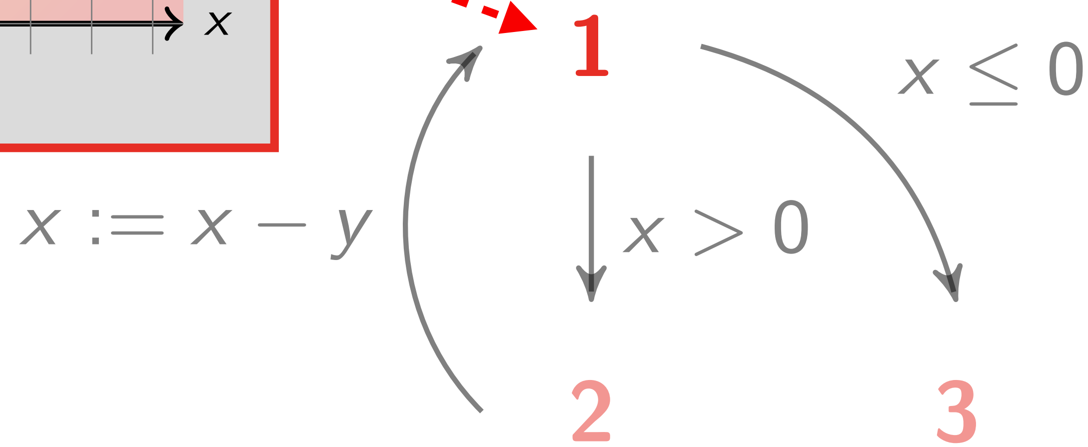
while [1]$(x > 0)$ do

[2]$x := x - y$

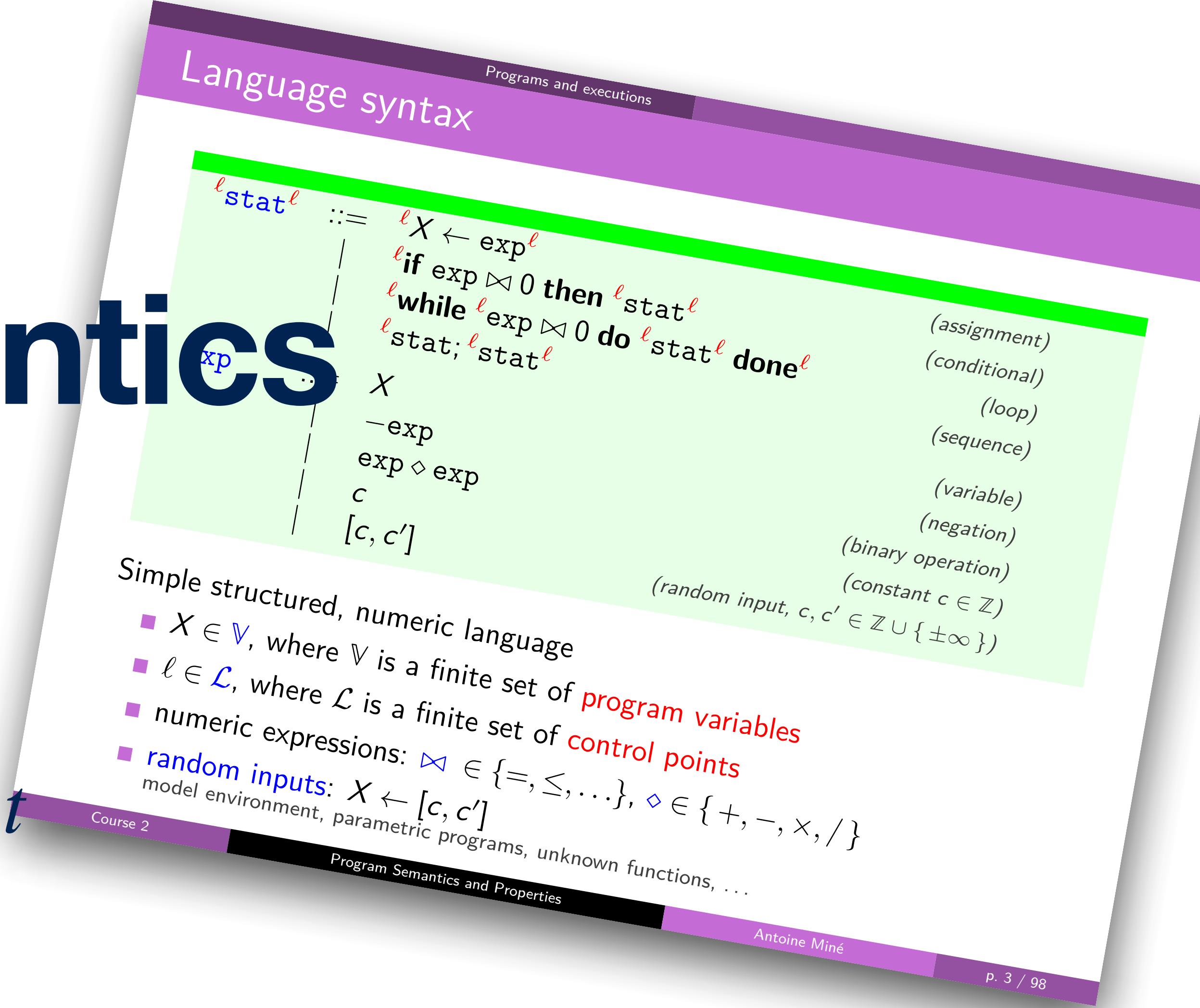
od[3]



the analysis gives the **weakest precondition** $x \leq 0 \vee y > 0$



Abstract Definite Termination Semantics



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

- $\mathscr{R}^\#_M[\![^\ell X \leftarrow e]\!]t \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]t$

- $\mathscr{R}^\#_M[\![\textbf{if } {}^\ell\, e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=}$
  $\text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^\#_M[\![s]\!]t) \curlyvee_T \text{FILTER}_A[\![e \not\bowtie 0]\!]t$

- $\mathscr{R}^\#_M[\![\textbf{while } {}^\ell\, e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]t \stackrel{\text{def}}{=} \text{lfp}^\# \overline{F}^\#_M$
  where $\overline{F}^\#_M(x) \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^\#_M[\![s]\!]x) \curlyvee_T \text{FILTER}_A[\![e \not\bowtie 0]\!](t)$

- $\mathscr{R}^\#_M[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathscr{R}^\#_M[\![s_1]\!](\mathscr{R}^\#_M[\![s_2]\!]t)$

Abstract Definite Termination Semantics

Definition

The **abstract definite termination semantics** $\mathcal{R}^{\#}_{M}[\![\text{stat}^{\ell}]\!] \in \mathcal{A}$ of a program stat^{ℓ} is:

$$\mathcal{R}^{\#}_{M}[\![\text{stat}^{\ell}]\!] \stackrel{\text{def}}{=} \mathcal{R}^{\#}_{M}[\![\text{stat}]\!](\text{LEAF}\colon \lambda X_1, \ldots, X_k.0)$$

where $\mathcal{R}^{\#}_{M}[\![\text{stat}]\!] : \mathcal{A} \to \mathcal{A}$ is the abstract definite termination semantics of each program instruction stat

Theorem (Soundness)

$$\mathcal{R}_{M}[\![\text{stat}^{\ell}]\!] \preccurlyeq \gamma_A(\mathcal{R}^{\#}_{M}[\![\text{stat}^{\ell}]\!])$$

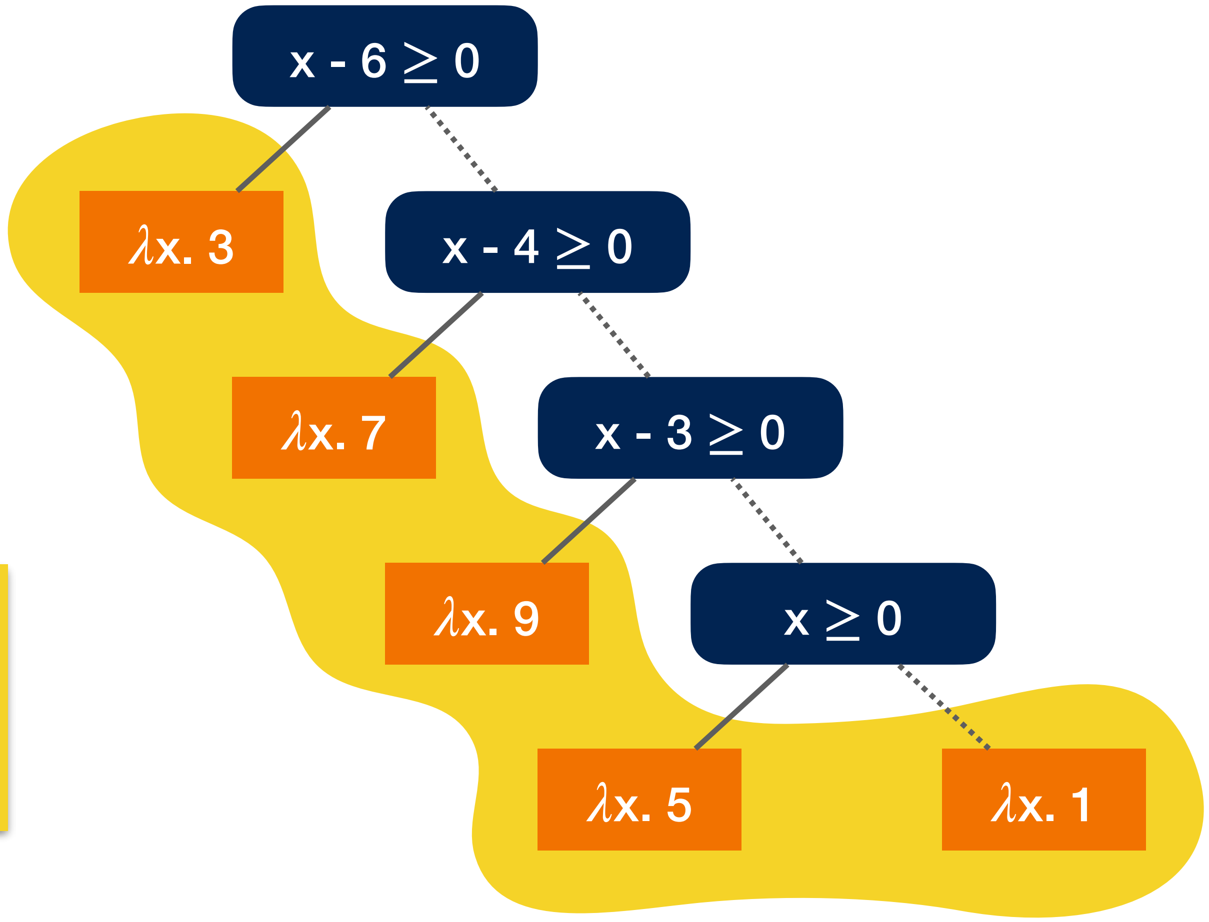
Corollary (Soundness)

A program stat^{ℓ} **must terminate** for traces starting from a set of initial states $\mathcal{I}$ if $\mathcal{I} \subseteq \text{dom}(\gamma_A(\mathcal{R}^{\#}_{M}[\![\text{stat}^{\ell}]\!]))$

Piecewise-Defined Ranking Functions Abstract Domain

Ordinal-Valued Functions Auxiliary Domain

- Parameterized by the ***underlying functions auxiliary domain*** $\langle \mathcal{F}, \sqsubseteq_F \rangle$
- $\mathcal{W} \stackrel{\text{def}}{=} \{ \perp_W \} \cup \{ \sum_i \omega^i \cdot f_i \mid f_i \in \mathcal{F} \setminus \{ \perp_F, \top_F \} \} \cup \{ \top_W \}$



Cantor Normal Form

$\omega^{\beta_1} \cdot n_1 + \ldots + \omega^{\beta_k} \cdot n_k$

Piecewise-Defined Ranking Functions Abstract Domain

Abstract Domain Operators

- They manipulate elements in $\mathscr{A}_{\text{NIL}} \stackrel{\text{def}}{=} \{\text{NIL}\} \cup \mathscr{A}$

- The **binary operators** rely on a tree unification algorithm
  - approximation order $\preccurlyeq_A$ and computational order $\sqsubseteq_A$
  - approximation join $\curlyvee_A$ and computational join $\sqcup_A$
  - meet $\curlywedge_A$
  - widening $\triangledown_A$

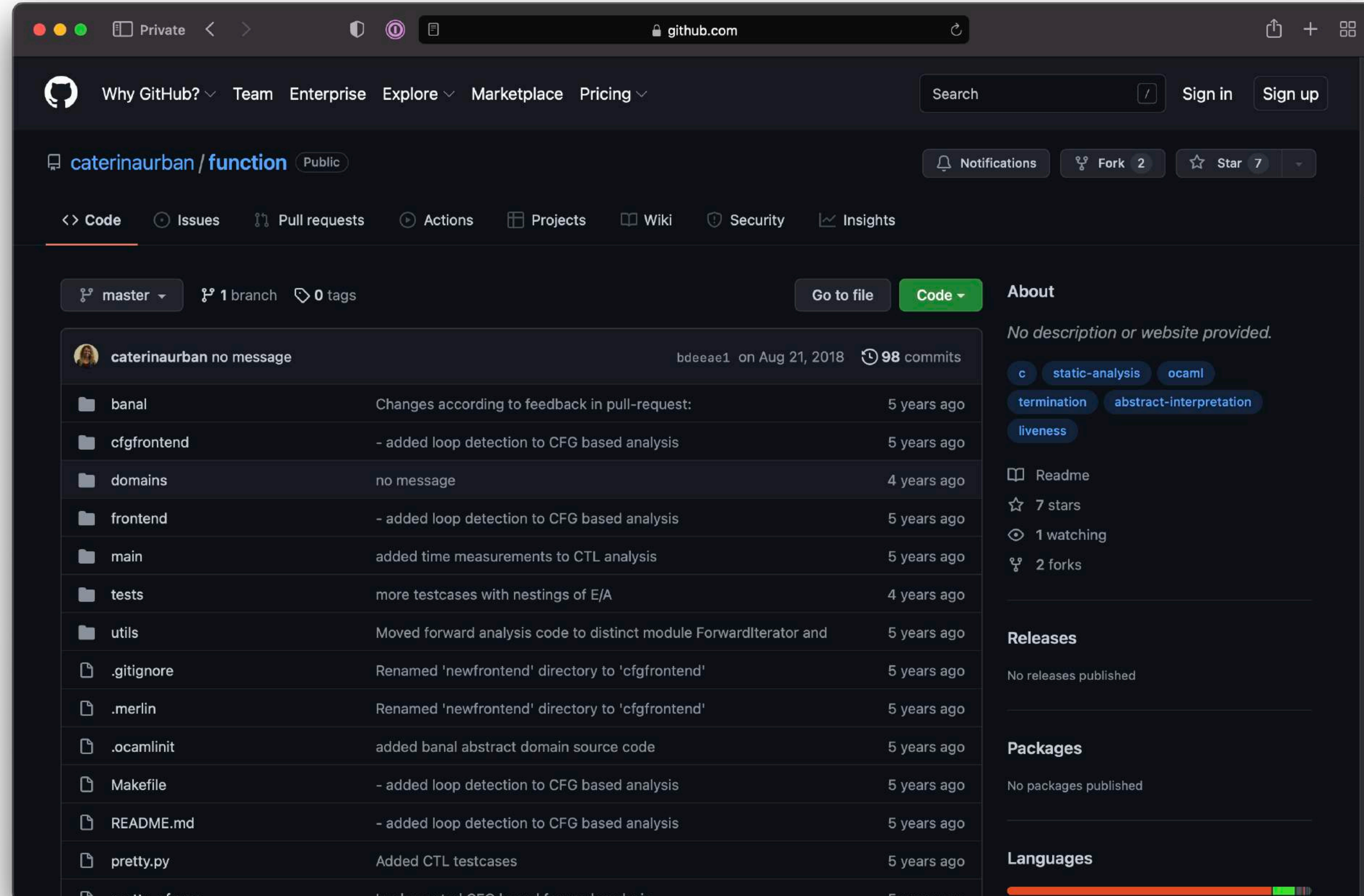
- The **unary operators** rely on a tree pruning algorithm
  - assignment $\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]$
  - test $\text{FILTER}_A[\![e]\!]$

Abstract Interpretation Recipe

practical tools
targeting specific programs

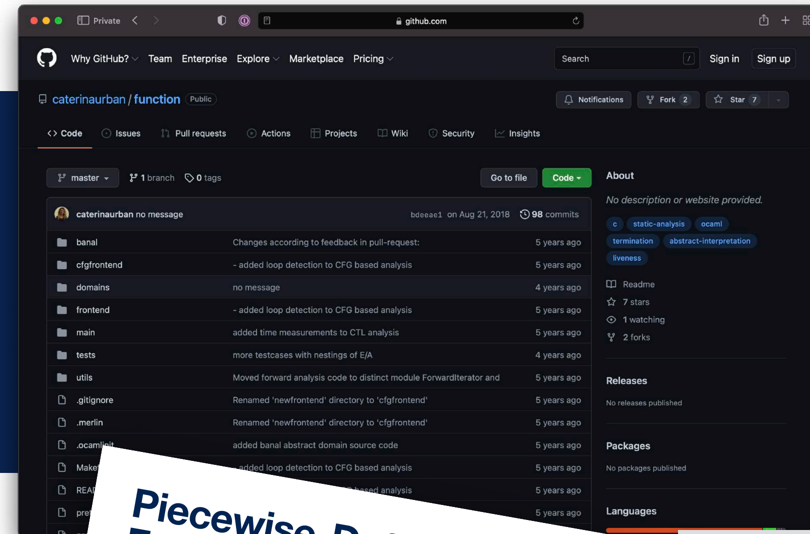
algorithmic approaches
to decide program properties

mathematical models
of the program behavior

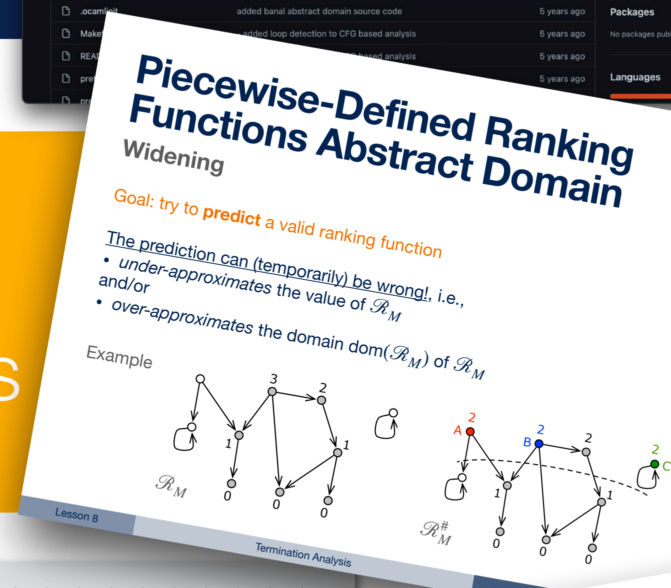


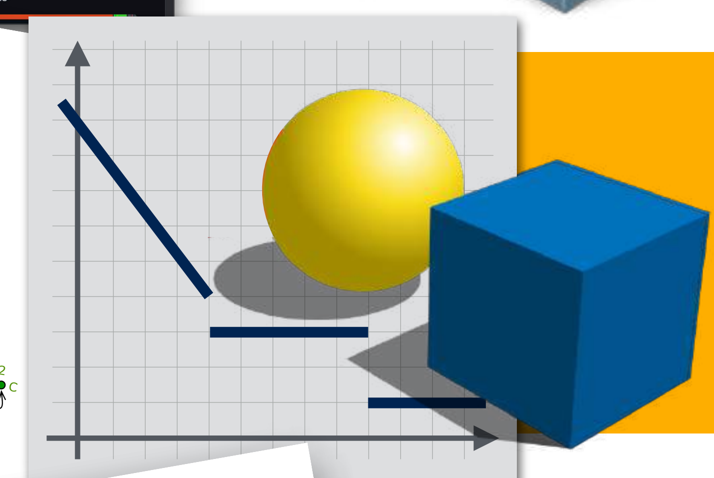
Private
github.com
Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search
Sign in
Sign up
caterinaurban / function
Public
Notifications
Fork 2
Star 7
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
master
1 branch
0 tags
Go to file
Code
caterinaurban no message
bdeeae1 on Aug 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

Abstract Interpretation Recipe

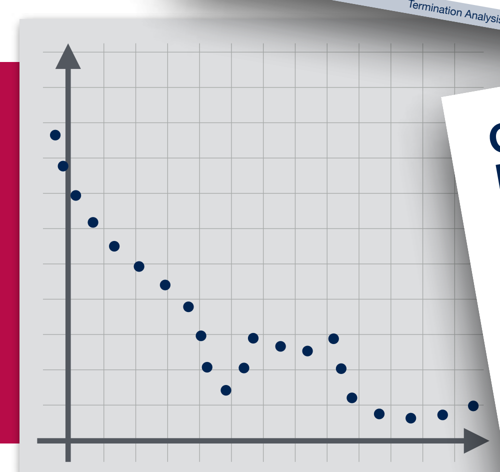


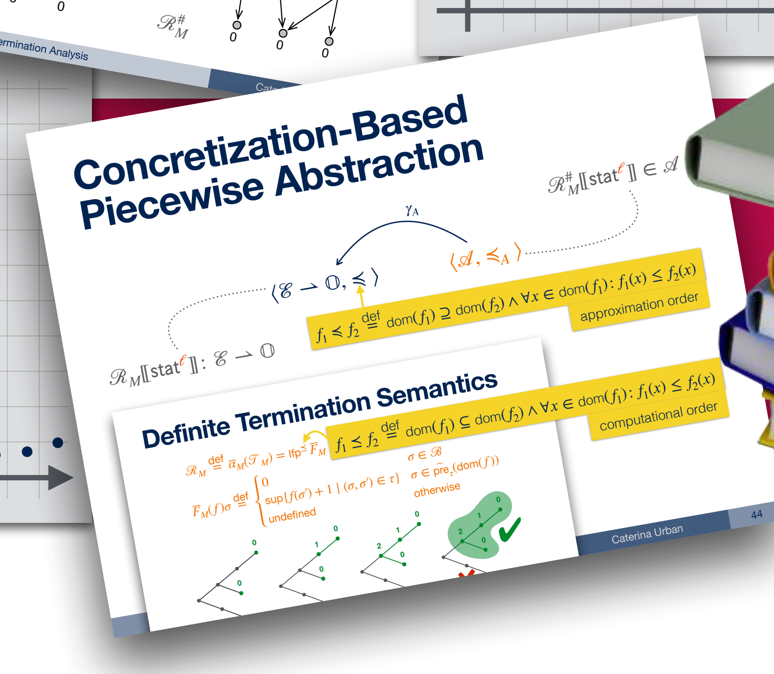
practical tools
targeting specific programs





algorithmic approaches
to decide program properties





mathematical models
of the program behavior

Today

Liveness Properties

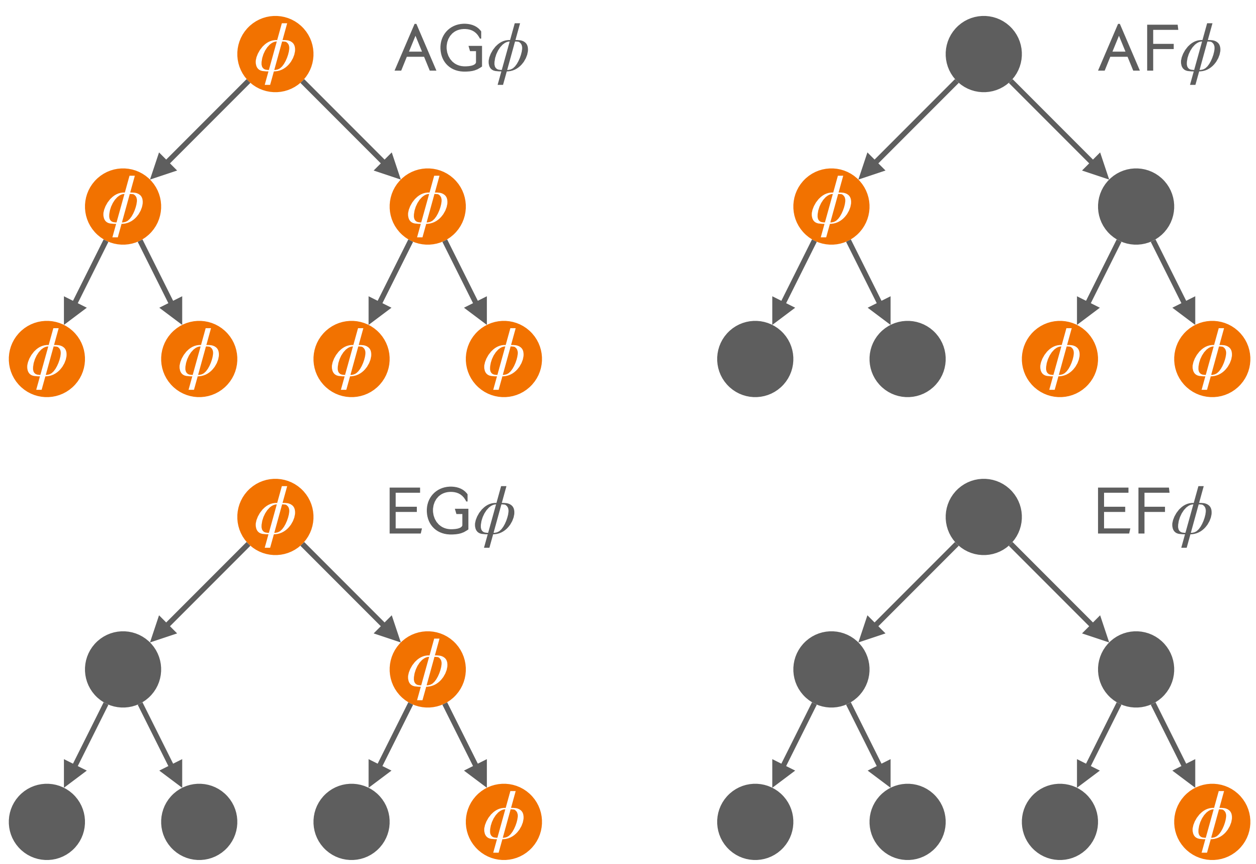
- **Guarantee Properties**
  "something good eventually happens at least once"
  - Example: Program Termination
- **Recurrence Properties**
  "something good eventually happens infinitely often"
  - Example: Starvation Freedom

Computation Tree Logic (CTL)

Branching Temporal Logic

$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$

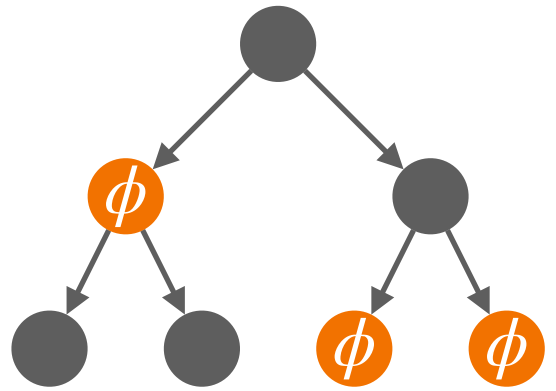
$$\mathsf{AF}\phi \equiv \mathsf{A}(\mathbf{true}\ \mathsf{U}\ \phi) \qquad \mathsf{EF}\phi \equiv \mathsf{E}(\mathbf{true}\ \mathsf{U}\ \phi)$$



Guarantee Properties

Guarantee Properties

"something good eventually happens at least once"



$$\mathsf{AF}\ \phi$$

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

Example:

^{1}x ← [-∞, +∞]
 while ^{2}(x ≥ 0) **do**
      ^{3}x ← x + 1
od^{4}
 while ^{5}(0 ≥ 0) **do**
      if ^{6}(x ≤ 10) **do**
            ^{7}x ← x + 1
      else
            ^{8}x ← -x
od^{9}

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

Abstract Interpretation Recipe

practical tools
targeting specific programs

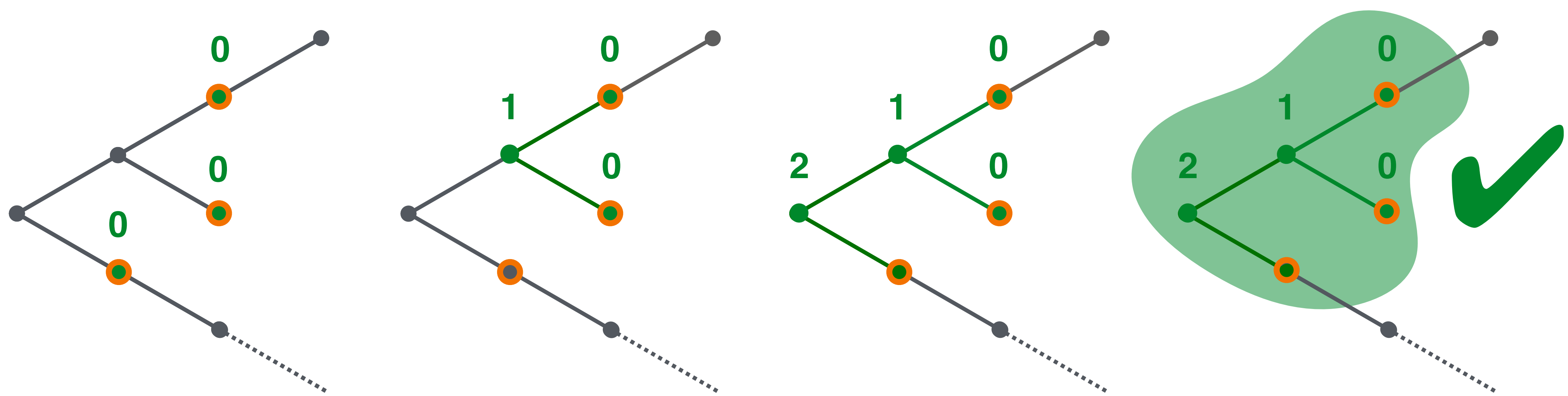
algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Guarantee Semantics

$$\mathcal{R}_G^{\varphi} \stackrel{\text{def}}{=} \mathsf{lfp}^{\preceq}\, \overline{F}_G[\{\sigma \in \Sigma \mid \sigma \vDash \varphi\}]$$

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



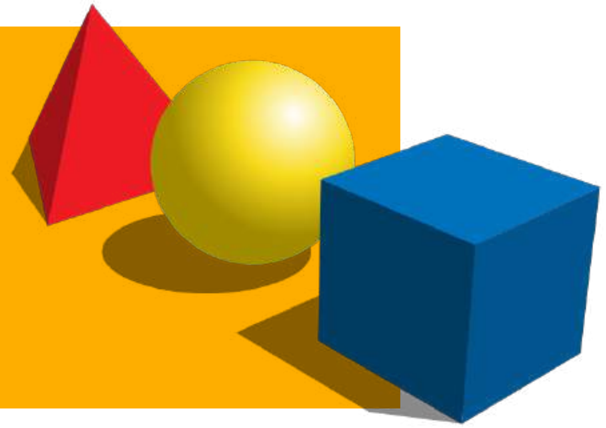
Theorem

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

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Abstract Guarantee Semantics

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

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

- $\mathscr{R}_G^{\varphi\#}[\![\mathbf{if}\ {}^{\ell}e \bowtie 0\ \mathbf{then}\ s]\!]t \stackrel{\text{def}}{=} \mathsf{RESET}_A^G[\![\varphi]\!](X)$
  where $X \stackrel{\text{def}}{=} \mathsf{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}_G^{\varphi\#}[\![s]\!]t) \curlyvee_T \mathsf{FILTER}_A[\![e \not\bowtie 0]\!]t)$

- $\mathscr{R}_G^{\varphi\#}[\![\mathbf{while}\ {}^{\ell}e \bowtie 0\ \mathbf{do}\ s\ \mathbf{done}]\!]t \stackrel{\text{def}}{=} \mathsf{lfp}^{\#}\overline{F}_G^{\varphi\#}$
  where $\overline{F}_G^{\varphi\#}(x) \stackrel{\text{def}}{=} \mathsf{RESET}_A^G[\![\varphi]\!](X)$
  $X \stackrel{\text{def}}{=} \mathsf{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}_G^{\varphi\#}[\![s]\!]x) \curlyvee_T \mathsf{FILTER}_A[\![e \not\bowtie 0]\!](t))$

- $\mathscr{R}_G^{\varphi\#}[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathscr{R}_G^{\varphi\#}[\![s_1]\!](\mathscr{R}_G^{\varphi\#}[\![s_2]\!]t)$

Abstract Guarantee Semantics

Example

int : x, y

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

  $^2x := x + 1$

od

while $^3($ true $)$ do

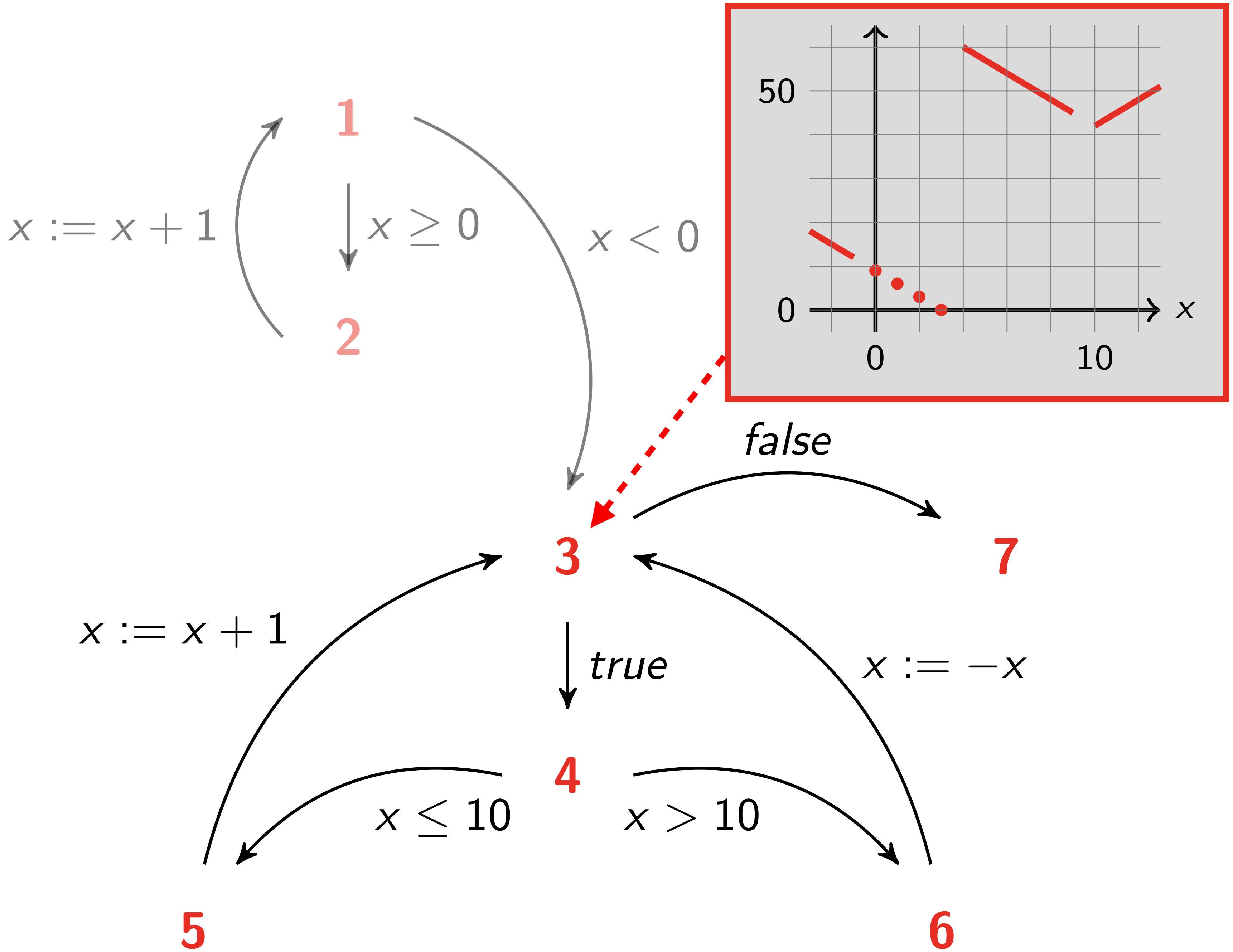
  if $^4($ $x \leq 10$ $)$

    $^5x := x + 1$

  else

    $^6x := -x$

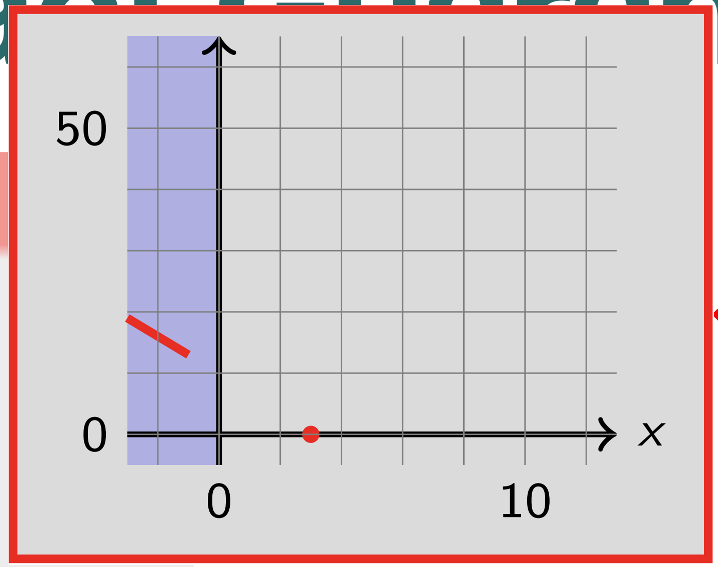
od^7



Property

$$\text{AF}\,(x = 3)$$

Abstract Guarantee Semantics



Example

int : $x, \; y$

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

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

od

while $^{3}($ true $)$ do

$\quad$ if $^{4}(\; x \leq 10 \;)$

$\quad\quad ^{5}x := x + 1$

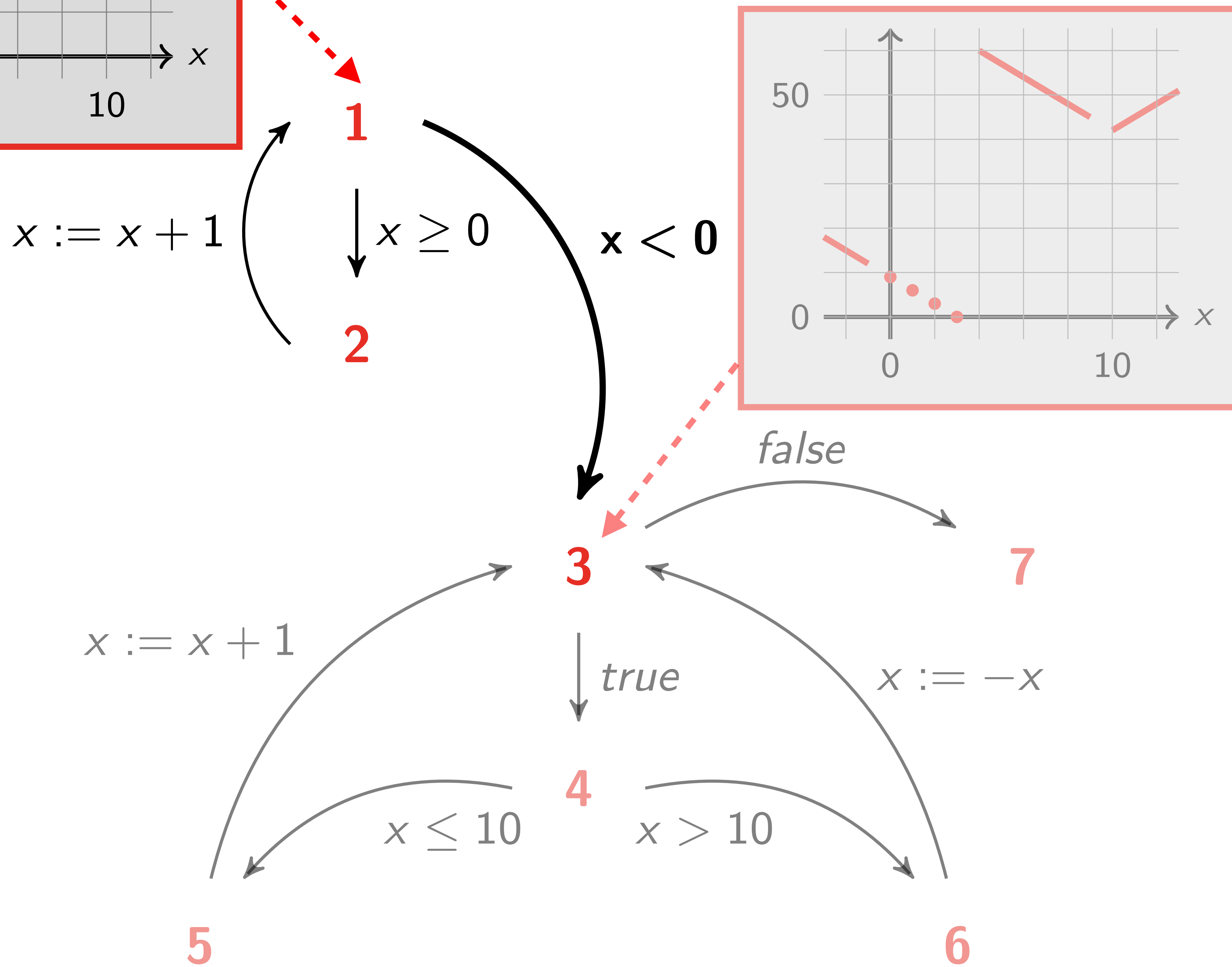
$\quad$ else

$\quad\quad ^{6}x := -x$

od^{7}

Property

$\mathsf{AF}\,(x = 3)$



Abstract Guarantee Semantics

Example

int : x, y

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

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

od

while $^{3}($ true $)$ do

  if $^{4}(\ x \leq 10\ )$

    $^{5}x := x + 1$

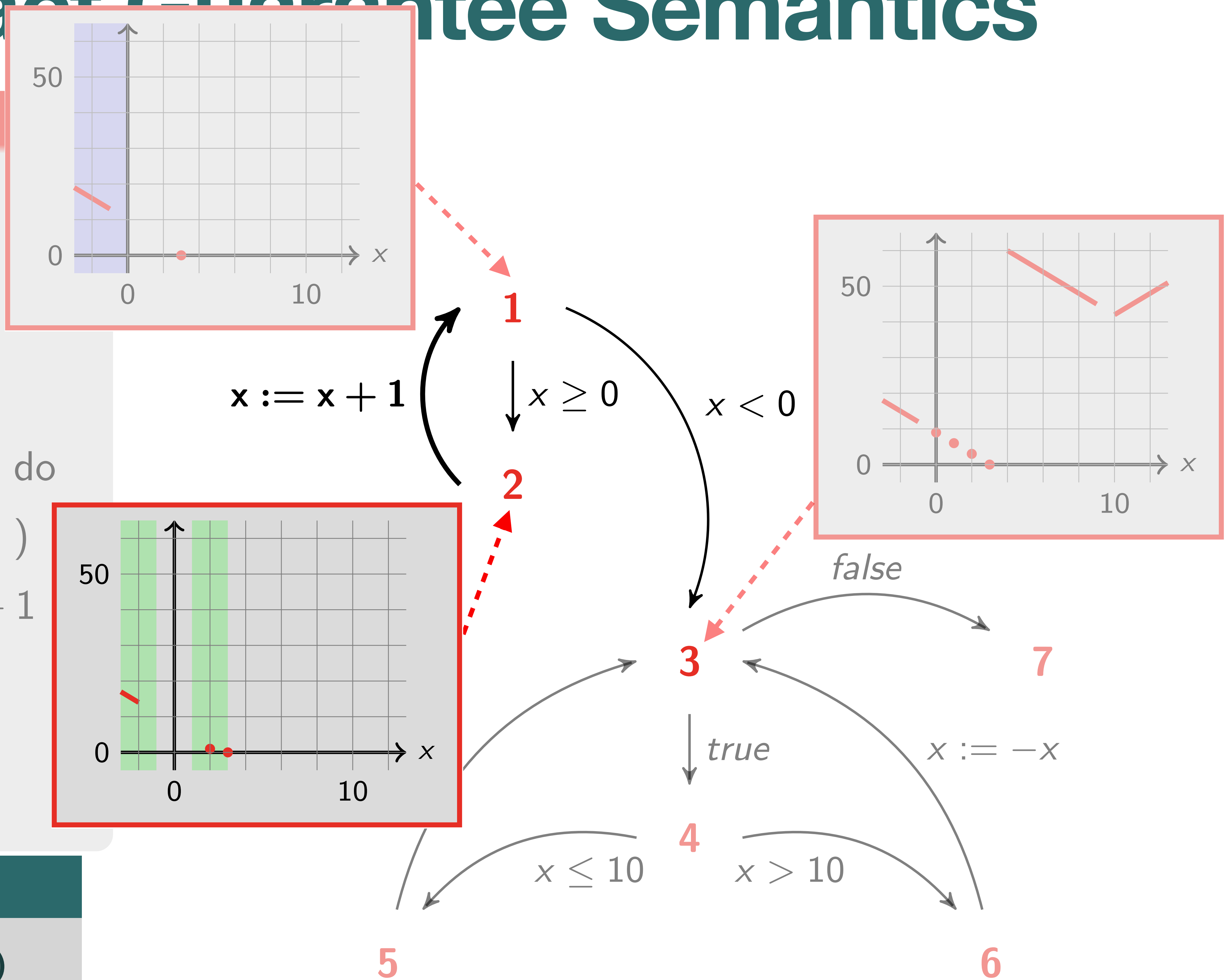
  else

    $^{6}x := -x$

od^{7}

Property

$$\mathsf{AF}\,(x = 3)$$



Abstract Guarantee Semantics

Example

int : x, y

while [1]$(x \geq 0)$

[2]$x := x + 1$

od

while [3](true) do

if [4]($x \leq 10$)

[5]$x := x + 1$

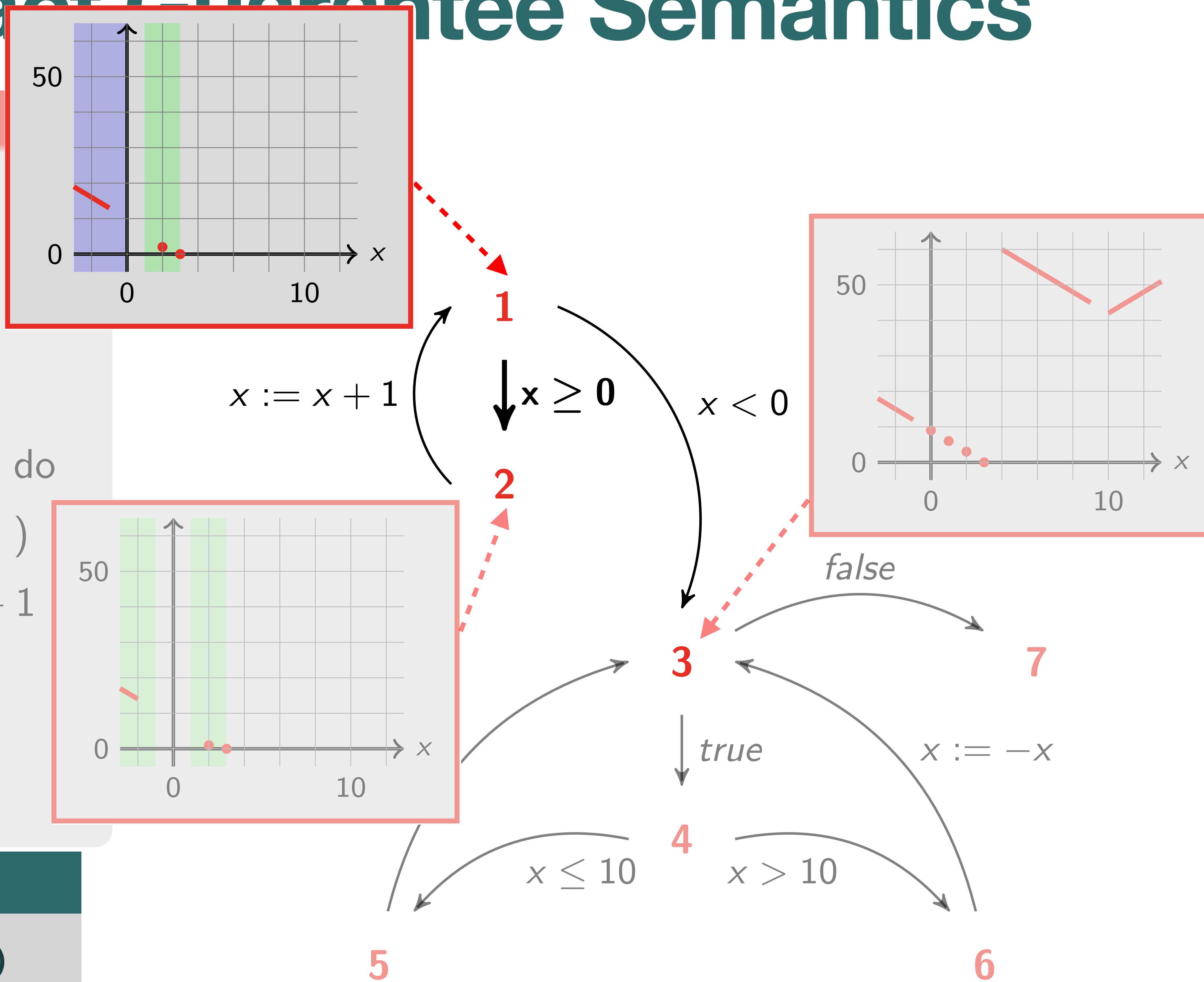
else

[6]$x := -x$

od[7]

Property

$$\mathsf{AF}\,(x = 3)$$



Abstract Guarantee Semantics

Example

int : x, y

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

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

od

while ${}^3($ true $)$ do

if ${}^4(\ x \leq 10\ )$

${}^5x := x + 1$

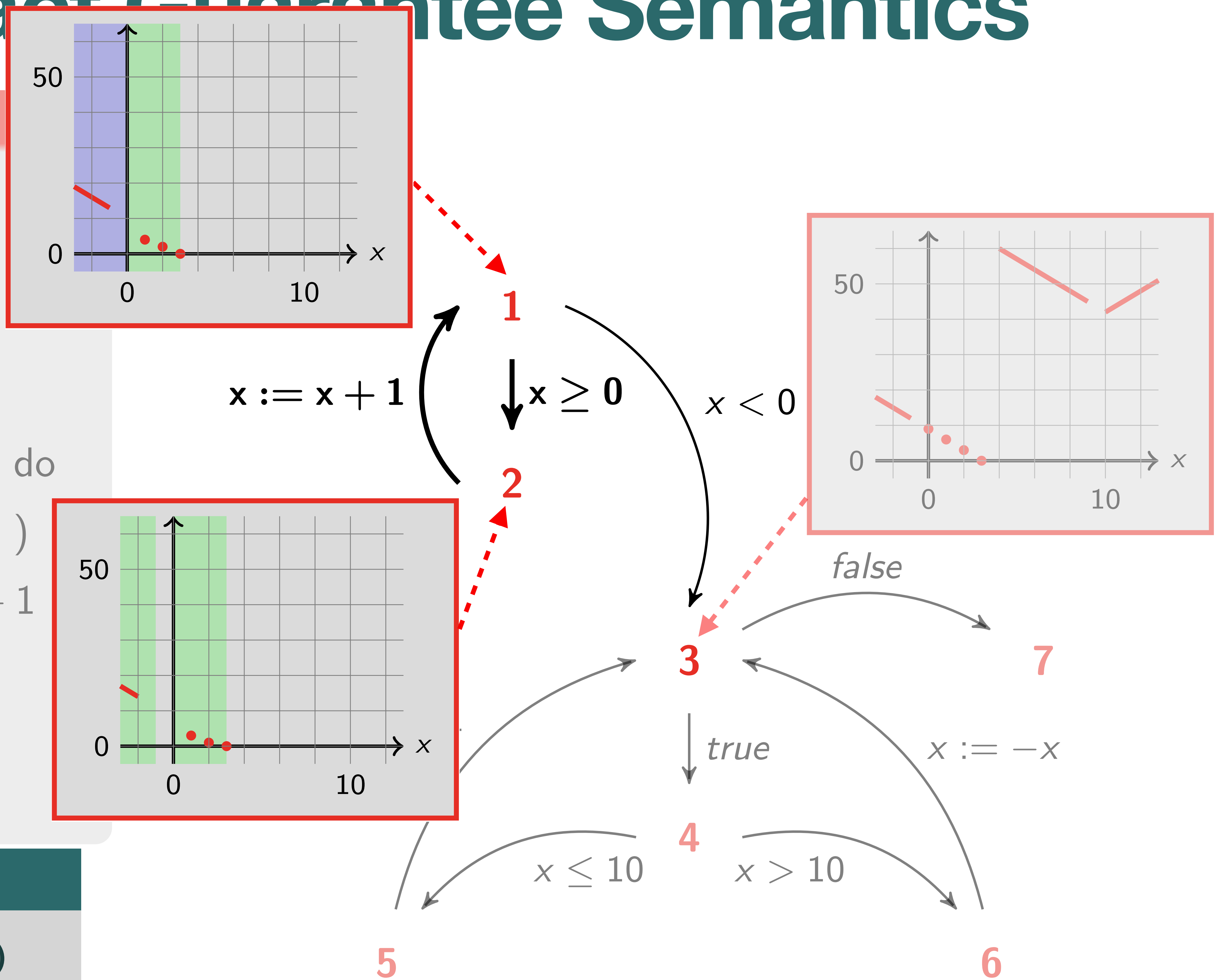
else

${}^6x := -x$

od^7

Property

$$\mathrm{AF}\,(x = 3)$$



Abstract Guarantee Semantics

Example

int : x, y

while [1]$(x \geq 0)$

  [2]$x := x + 1$

od

while [3](true) do

  if [4]($x \leq 10$)

    [5]$x := x + 1$

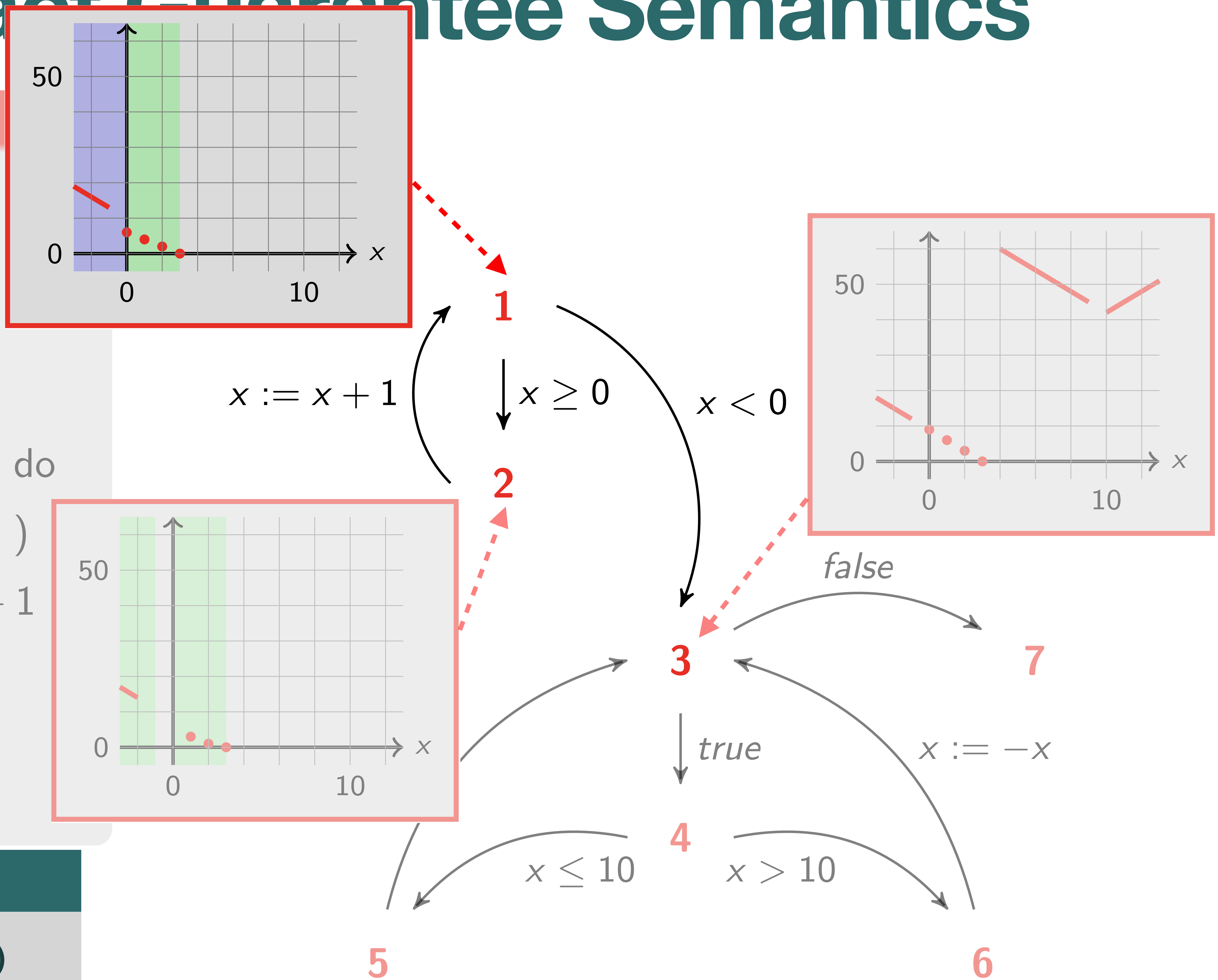
  else

    [6]$x := -x$

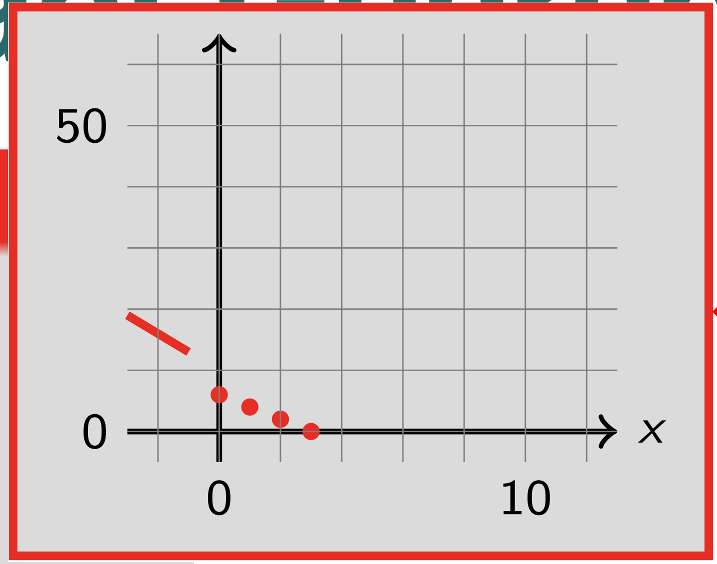
od[7]

Property

$$\mathrm{AF}\,(x = 3)$$



Abstract Guarantee Semantics



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

Example

int : x, y

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

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

od

while ^{3}(true) do

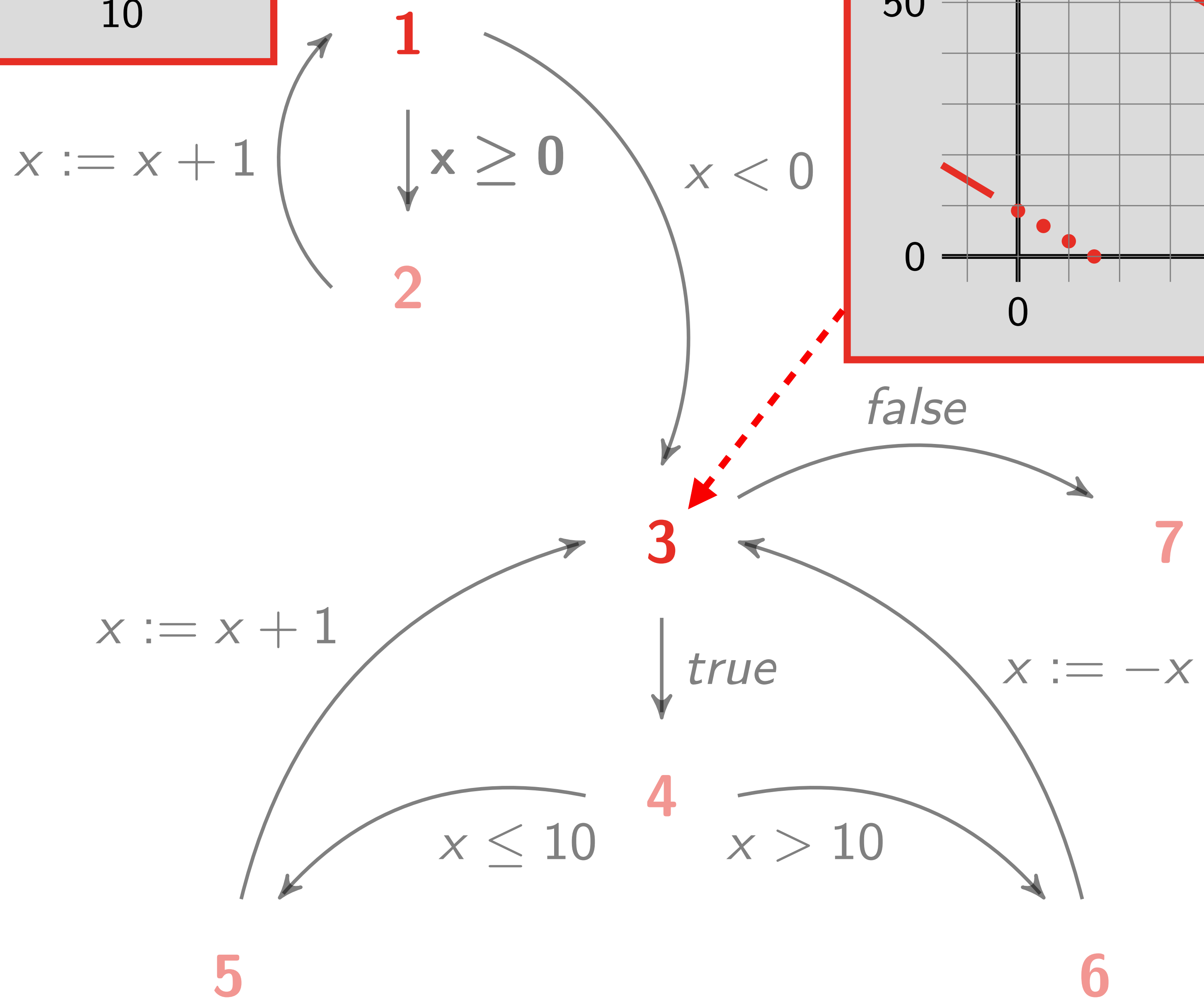
    if ${}^{4}(\ x \leq 10\ )$

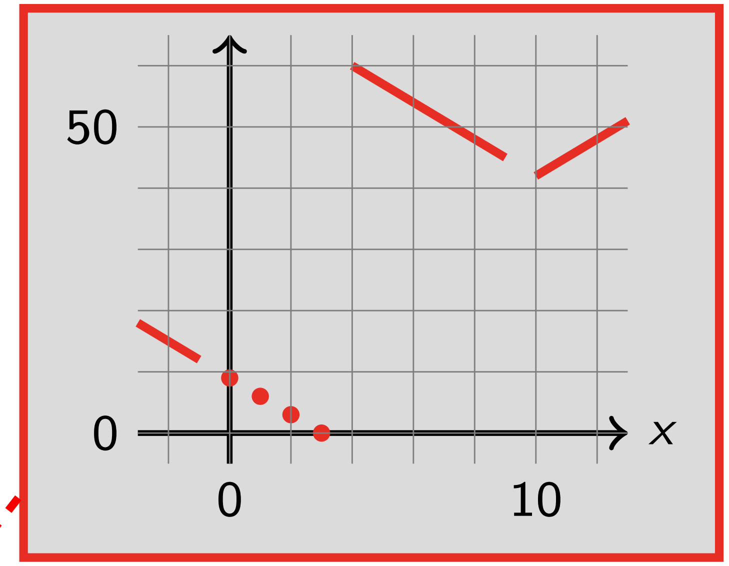
        ${}^{5}x := x + 1$

    else

        ${}^{6}x := -x$

od^{7}





Property

$$\mathsf{AF}\,(x = 3)$$

Abstract Guarantee Semantics

Definition

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

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

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** AF φ 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"

$$\text{AG AF } \phi$$

$\phi ::= e \bowtie 0 \;\mid\; \ell : e \bowtie 0 \;\mid\; \phi \wedge \phi \mid \phi \vee \phi$ $\qquad \ell \in \mathscr{L}$

Example:

AG AF $(x = 3)$ is satisfied for $\mathscr{I} \stackrel{\text{def}}{=} \{(1,\rho) \in \Sigma \mid \rho(x) < 0\}$

^{1}x ← [-∞, +∞]
 while ^{2}(x ≥ 0) **do**
  ^{3}x ← x + 1
od^4
 while ^{5}(0 ≥ 0) **do**
  if ^{6}(x ≤ 10) **do**
   ^{7}x ← x + 1
  else
   ^{8}x ← -x
od^9

Abstract Interpretation Recipe

practical tools
targeting specific programs

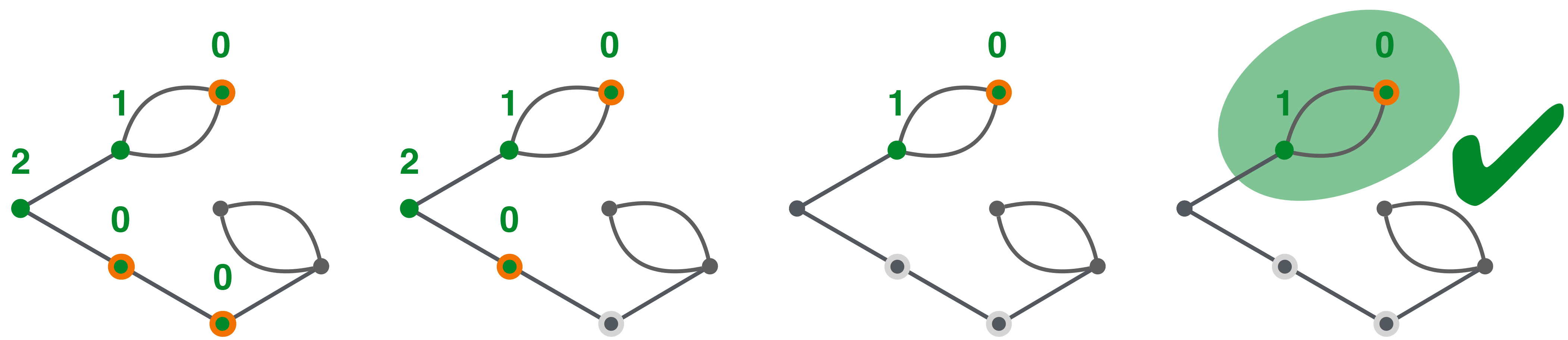
algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Recurrence Semantics

$$\mathcal{R}_R^{\varphi} \stackrel{\text{def}}{=} \textcolor{orange}{\mathsf{gfp}_{\mathcal{R}_G^{\varphi}}^{\preceq}} \overline{F}_R$$

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



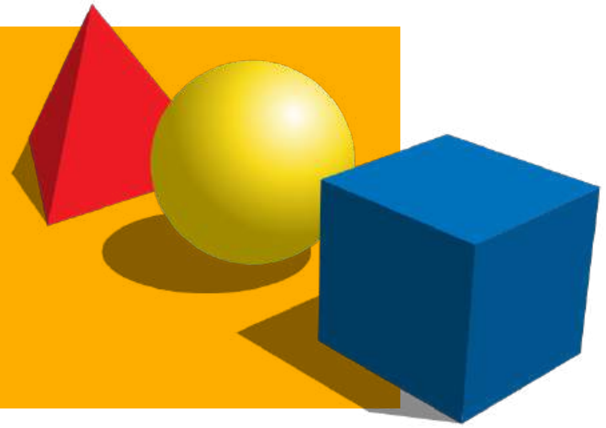
Theorem

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

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Abstract Recurrence Semantics

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

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

- $\mathcal{R}_R^{\varphi\#}[\![\textbf{if } {}^{\ell} e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=} \text{RESET}_A^R[\![\varphi]\!](X)$
  where $X \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathcal{R}_G^{\varphi\#}[\![s]\!]t) \curlyvee_T \text{FILTER}_A[\![e \not\bowtie 0]\!]t)$

- $\mathcal{R}_R^{\varphi\#}[\![\textbf{while } {}^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]t \stackrel{\text{def}}{=} \text{gfp}_{G(t)}^{\#} \overline{F}_R^{\varphi\#}$
  where $G \stackrel{\text{def}}{=} \mathcal{R}_G^{\varphi\#}[\![\textbf{while } {}^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]$
  $\overline{F}_R^{\varphi\#}(x) \stackrel{\text{def}}{=} \text{RESET}_A^R[\![\varphi]\!](X)$
  $X \stackrel{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathcal{R}_R^{\varphi\#}[\![s]\!]x) \curlyvee_T \text{FILTER}_A[\![e \not\bowtie 0]\!](t))$

- $\mathcal{R}_R^{\varphi\#}[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathcal{R}_R^{\varphi\#}[\![s_1]\!](\mathcal{R}_R^{\varphi\#}[\![s_2]\!]t)$

Dual Widening

Definition

Let $\langle \mathcal{D}, \sqsubseteq \rangle$ be a poset. A *dual widening* $\bar{\nabla} : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ obeys:

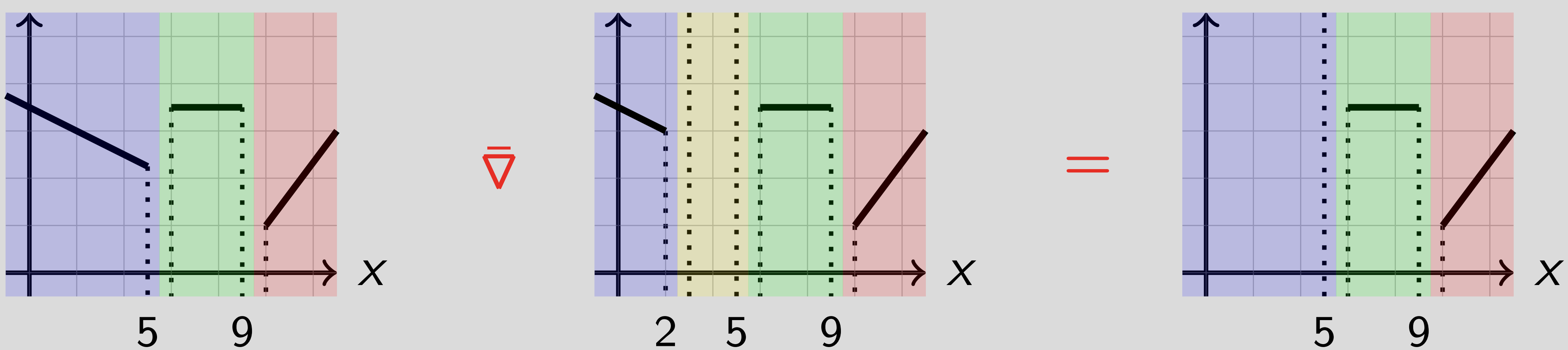
(1) for all element $x, y \in \mathcal{D}$, we have $x \sqsupseteq x \mathbin{\bar{\nabla}} y$ and $y \sqsupseteq x \mathbin{\bar{\nabla}} y$

(2) for all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq \cdots \sqsupseteq x_n \sqsupseteq \cdots$, the chain

$$y_0 \stackrel{\text{def}}{=} x_0 \qquad\qquad y_{n+1} \stackrel{\text{def}}{=} y_n \mathbin{\bar{\nabla}} x_{n+1}$$

is ultimately stationary

Example



Abstract Recurrence Semantics

Example

int : x, y

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

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

od

while ${}^{3}(\text{ true })$ do

  if ${}^{4}(\ x \leq 10\ )$

    ${}^{5}x := x + 1$

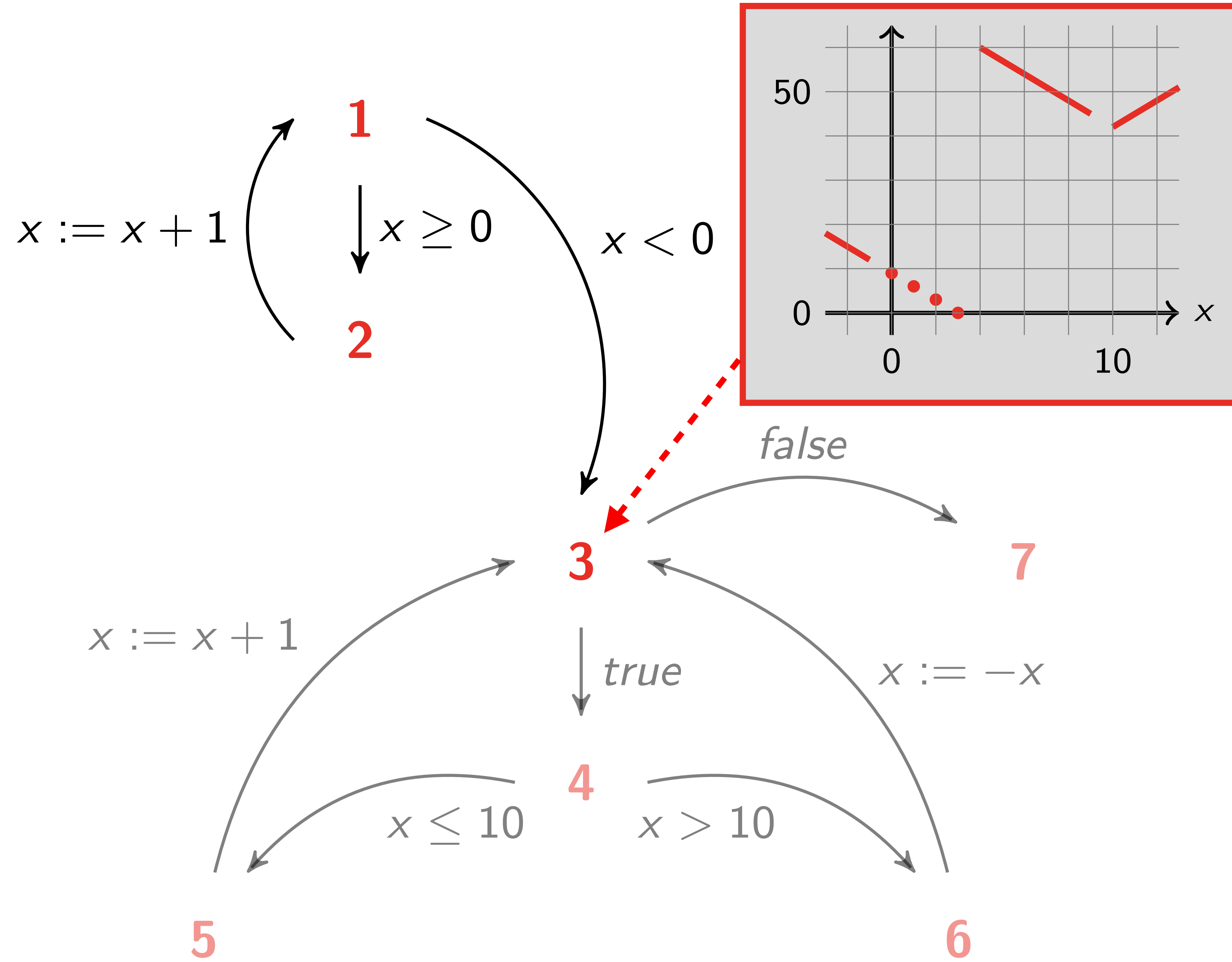
  else

    ${}^{6}x := -x$

od^{7}

Property

$\mathrm{AGAF}\,(x = 3)$



Abstract Recurrence Semantics

Example

int : x, y

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

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

od

while ^{3}(true) do

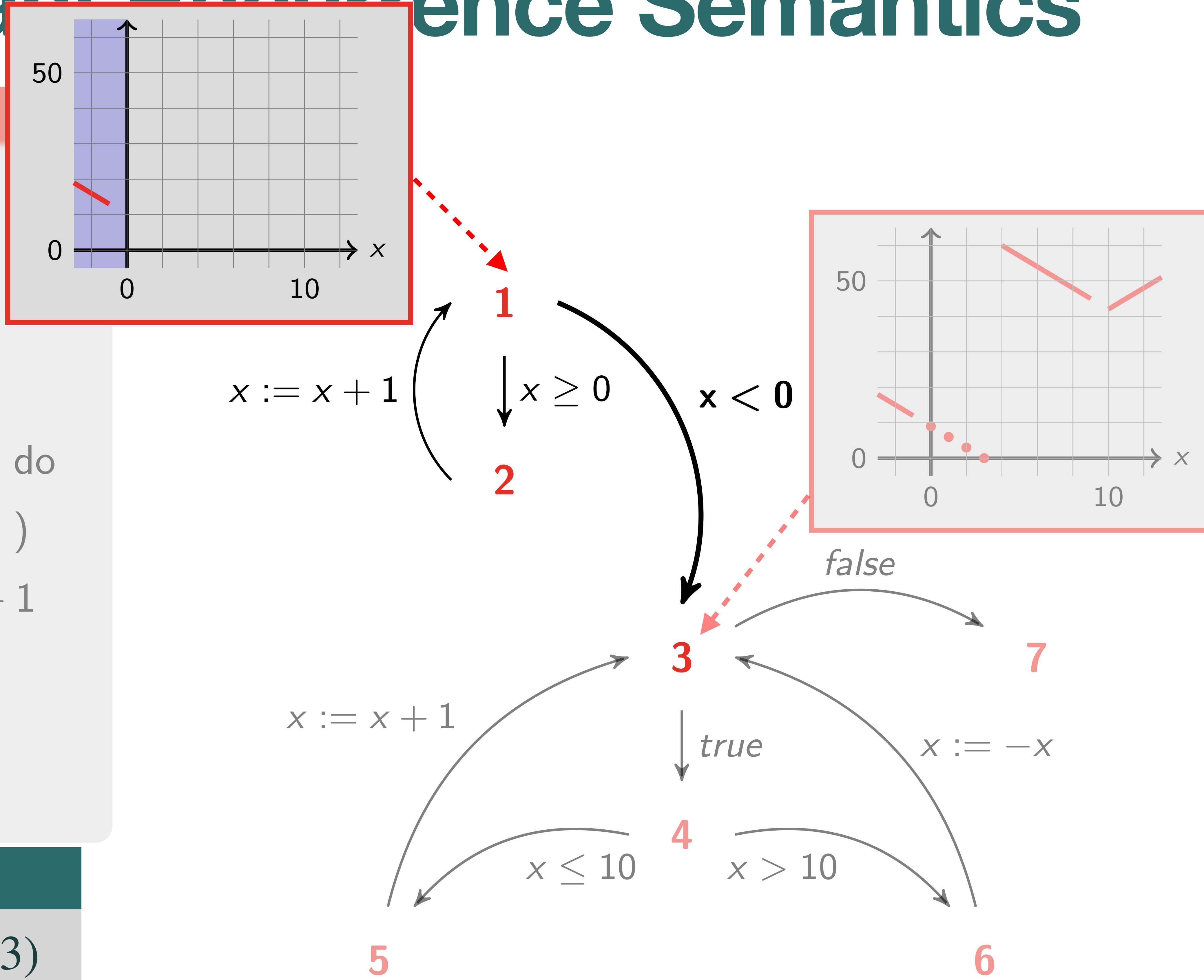
  if ^{4}($x \leq 10$)

    $^{5}x := x + 1$

  else

    $^{6}x := -x$

od^{7}



Property

$$\mathrm{AGAF}\,(x = 3)$$

Abstract Recurrence Semantics

Example

int : x, y

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

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

od

while $^{3}($ true $)$ do

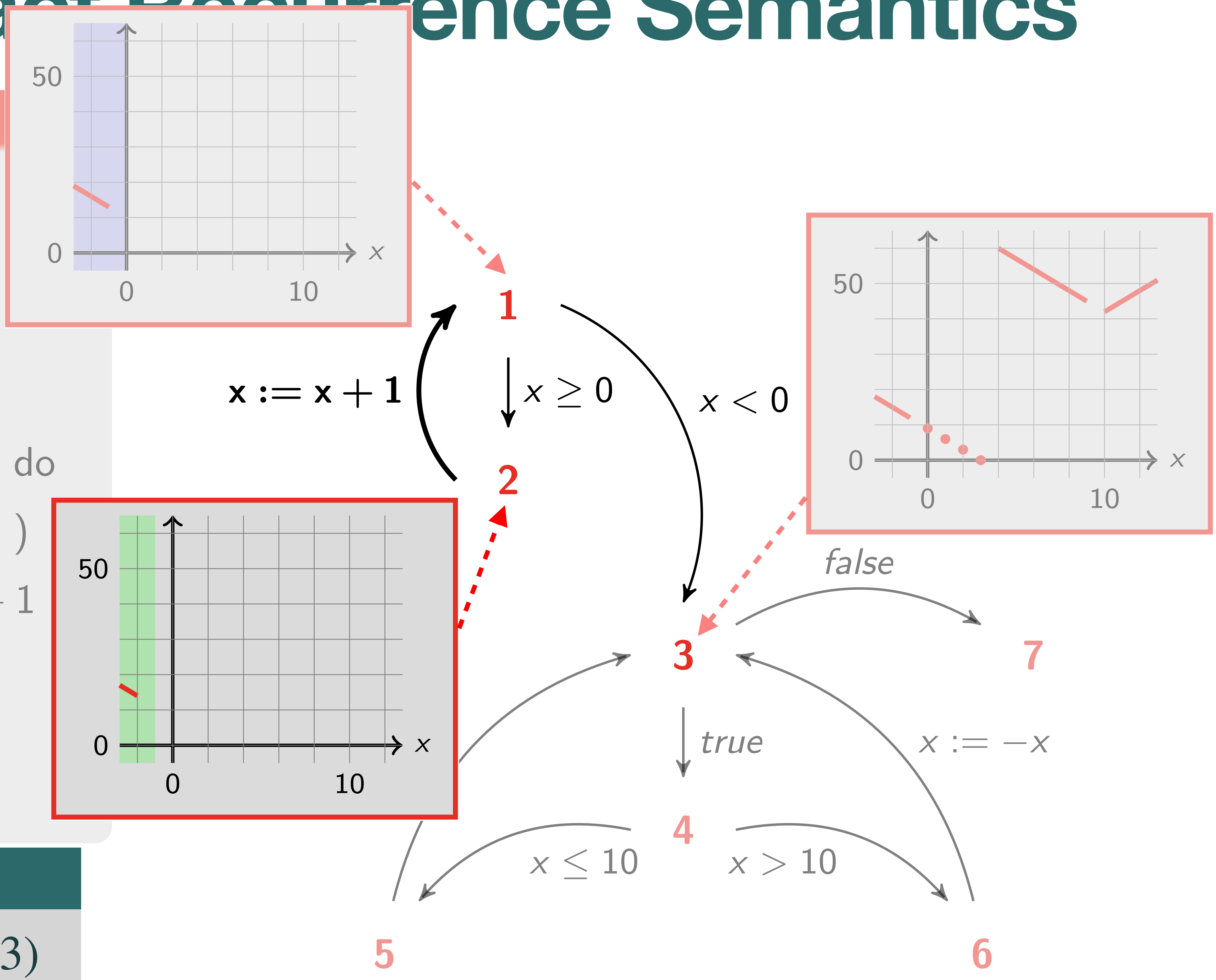
$\quad$ if $^{4}(\ x \leq 10\ )$

$\quad\quad ^{5}x := x + 1$

$\quad$ else

$\quad\quad ^{6}x := -x$

od^{7}



Property

$\mathrm{AGAF}\,(x = 3)$

Abstract Recurrence Semantics

Example

int : x, y

while $^1(x \geq 0)$

  $^2x := x + 1$

od

while $^3($ true $)$ do

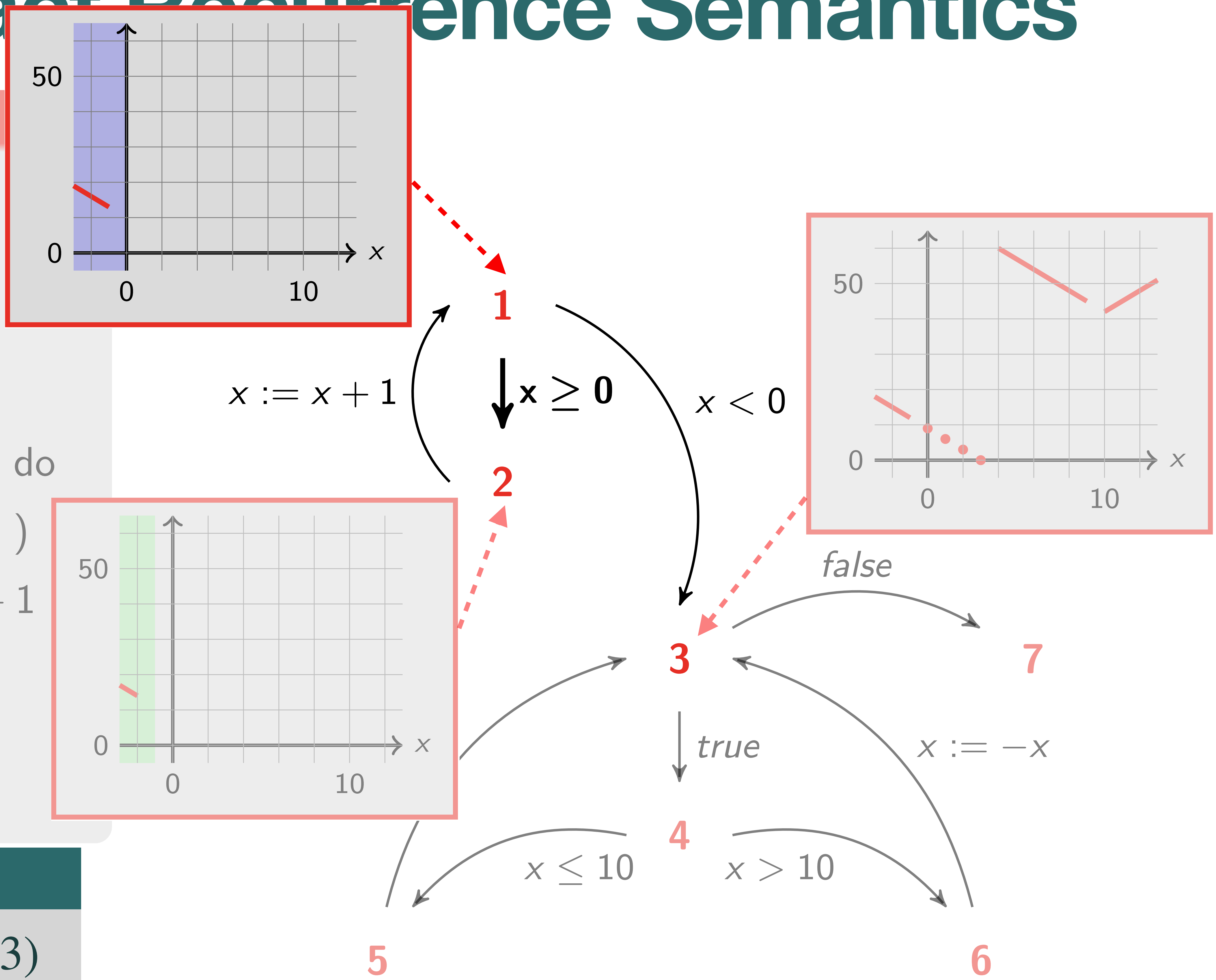
  if $^4(\ x \leq 10\ )$

    $^5x := x + 1$

  else

    $^6x := -x$

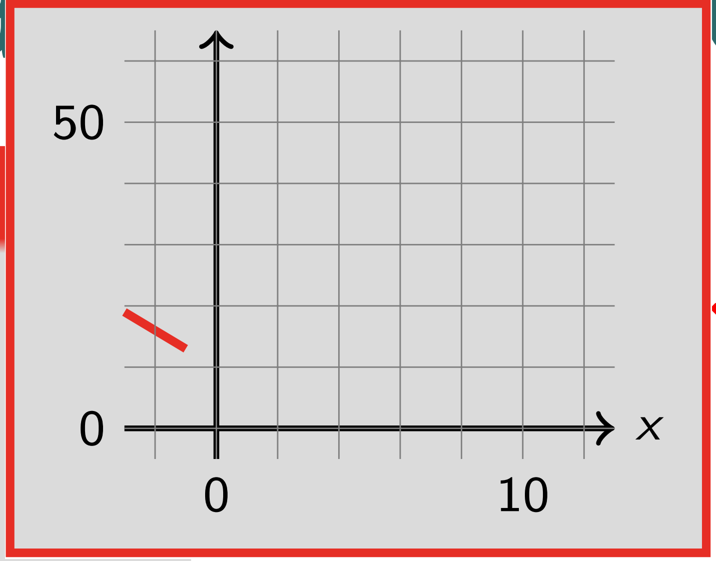
od^7



Property

$\mathrm{AGAF}\,(x = 3)$

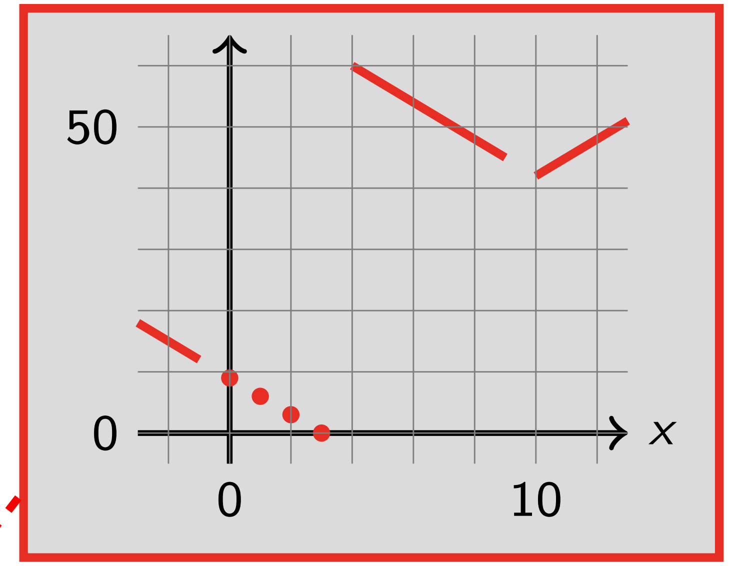
Abstract Recurrence Semantics

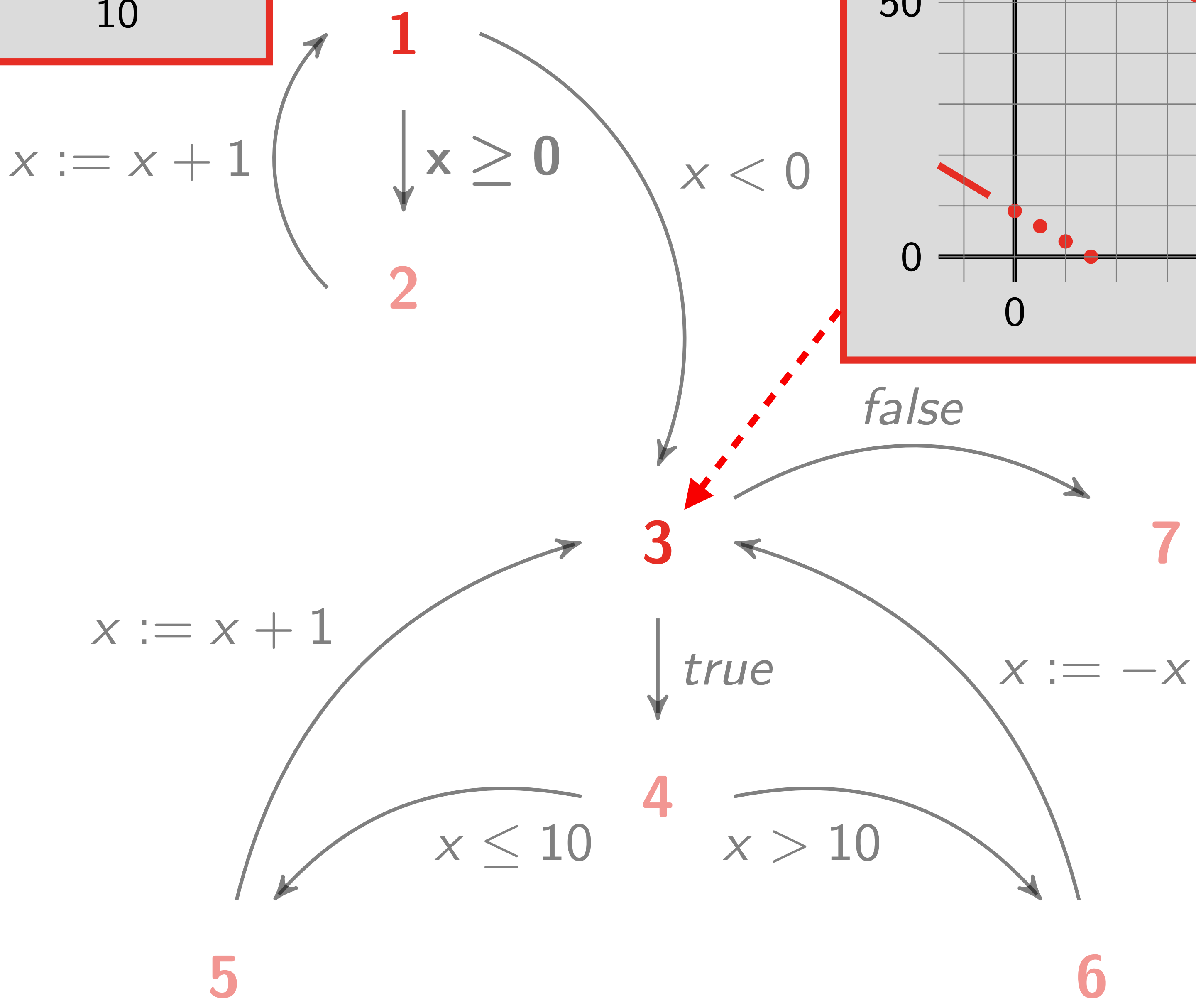


the analysis gives $x < 0$ as **sufficient precondition**

Example

$\text{int} : x,\ y$
while [1]$(x \geq 0)$
  [2]$x := x + 1$
od
while [3](true) do
  if [4]($x \leq 10$)
    [5]$x := x + 1$
  else
    [6]$x := -x$
od[7]





Property

$AGAF\,(x = 3)$

Abstract Recurrence Semantics

Definition

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

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

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

Corollary (Soundness)

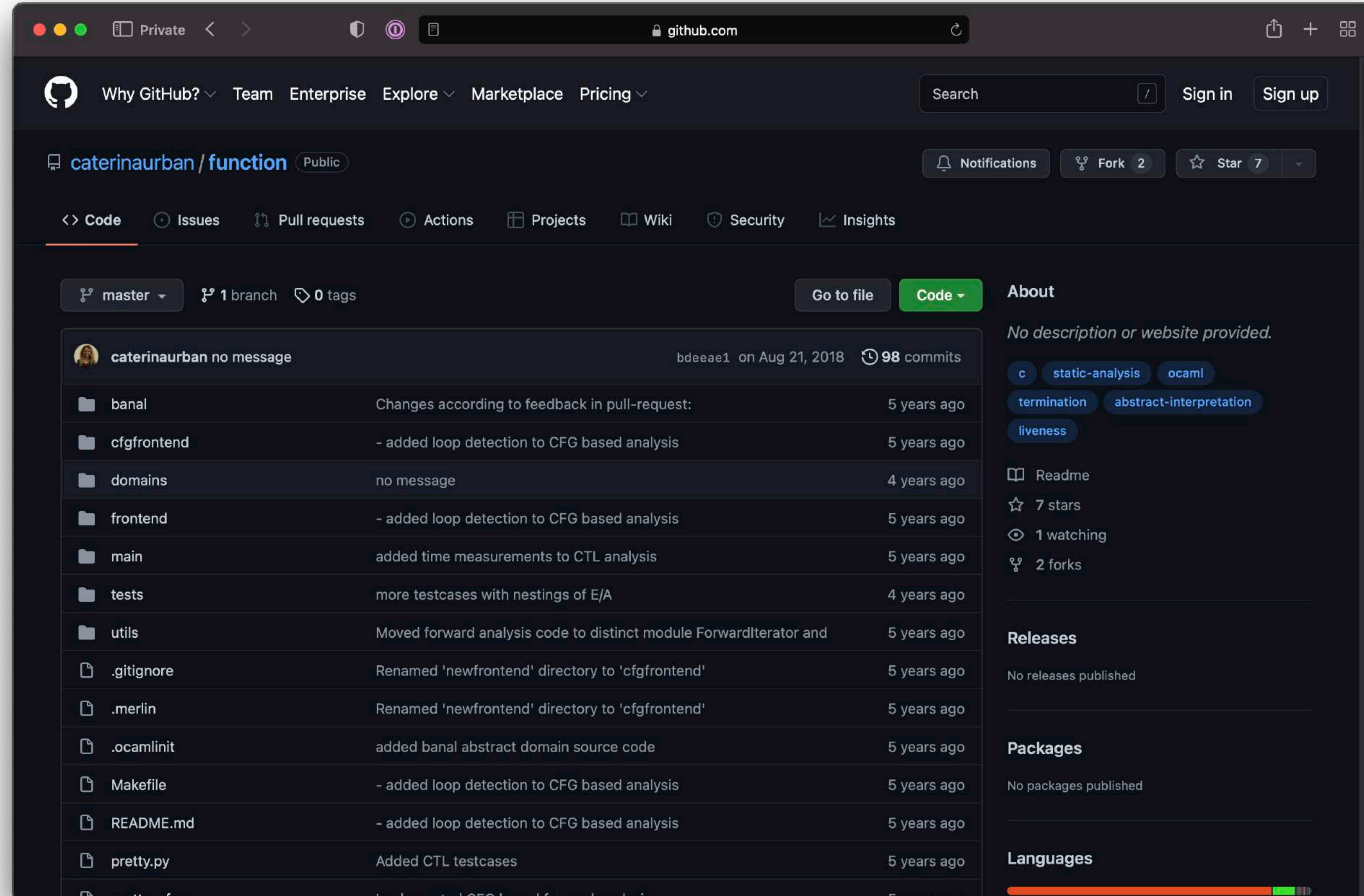
A program stat^{ℓ} satisfies a **recurrence property** AG AF φ for traces starting from a set of initial states $\mathscr{I}$ if $\mathscr{I} \subseteq \text{dom}(\gamma_A(\mathscr{R}_R^{\varphi\#}[\![\text{stat}^{\ell}]\!]))$

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior



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
bdeeae1 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
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

CTL Properties

Computation Tree Logic (CTL)

Branching Temporal Logic

$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$

Recurrence Properties

"something good eventually happens infinitely often"

$\mathsf{AG}\ \mathsf{AF}\ \phi$

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

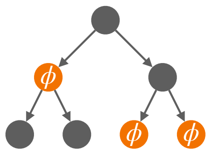
Example:

```
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
```

AG AF $(x = 3)$ is satisfied for $\mathscr{I} \stackrel{\text{def}}{=} \{(1,\rho) \in \Sigma \mid \rho(x) < 0\}$

Lesson 9 | Analysis of Liveness and CTL Properties | Caterina Urban | 55

Guarantee Properties



"something good eventually happens at least once"

$\mathsf{AF}\ \phi$

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

Example:

```
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
```

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

Lesson 9 | Analysis of Liveness and CTL Properties | Caterina Urban | 37

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

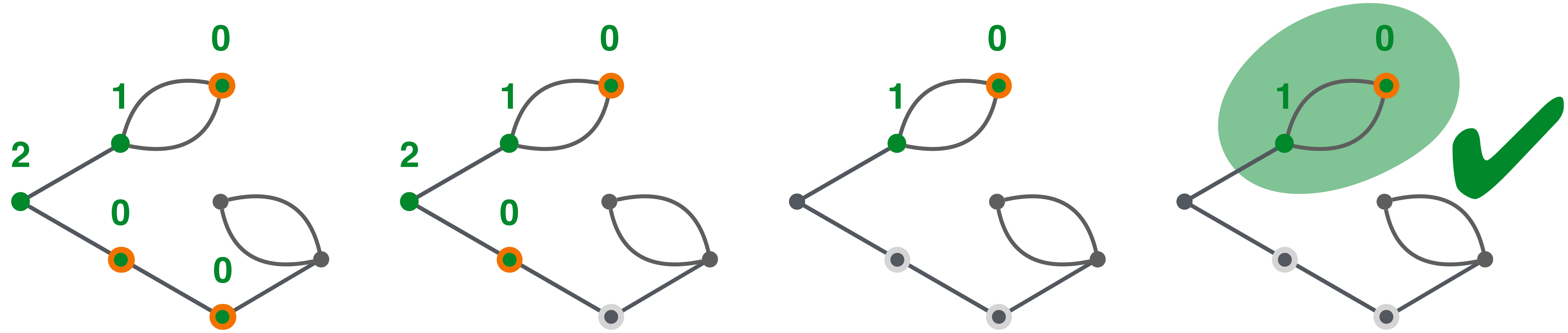
mathematical models
of the program behavior

Recurrence Semantics

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

build upon the semantics of sub-formulas

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



Theorem

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

CTL Abstraction

Atomic Propositions

$$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi\mathsf{U}\phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi\mathsf{U}\phi)$$

$$\alpha_a(T) \stackrel{\text{def}}{=} \lambda s \in st(T) \, . \begin{cases} 0 & s \models a \\ \text{undefined} & \text{otherwise} \end{cases}$$

CTL Abstraction

Negation Formulas

$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$

$$\alpha_{\neg\phi}(T) \overset{\text{def}}{=} \lambda s \in st(T)\,.\begin{cases} 0 & s \notin \mathsf{dom}(\alpha_{\phi}(T)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

CTL Abstraction

Conjunction Formulas

$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi\mathsf{U}\phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi\mathsf{U}\phi)$

$$\alpha_{\phi_1 \wedge \phi_2}(T) \overset{\text{def}}{=} \lambda s \in st(T)\,.\begin{cases} \sup\{f_1(s), f_2(s)\} & s \in \mathrm{dom}(f_1) \cap \mathrm{dom}(f_2) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$f_1 \overset{\text{def}}{=} \alpha_{\phi_1}(T)$$

$$f_2 \overset{\text{def}}{=} \alpha_{\phi_2}(T)$$

CTL Abstraction

Disjunction Formulas

$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$

$$\alpha_{\phi_1 \wedge \phi_2}(T) \stackrel{\text{def}}{=} \lambda s \in st(T)\,.\begin{cases} \sup\{f_1(s), f_2(s)\} & s \in \mathrm{dom}(f_1) \cap \mathrm{dom}(f_2) \\ f_1(s) & s \in \mathrm{dom}(f_1) \backslash \mathrm{dom}(f_2) \\ f_2(s) & s \in \mathrm{dom}(f_2) \backslash \mathrm{dom}(f_1) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$f_1 \stackrel{\text{def}}{=} \alpha_{\phi_1}(T)$$
$$f_2 \stackrel{\text{def}}{=} \alpha_{\phi_2}(T)$$

CTL Abstraction

Next Formulas

$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$

$$\alpha_{\mathsf{AX}\phi}(T) \stackrel{\mathsf{def}}{=} \lambda s \in st(T)\,.\begin{cases} 0 & s \in \widetilde{\mathsf{pre}}(\mathsf{dom}(\alpha_\phi(T))) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\alpha_{\mathsf{EX}\phi}(T) \stackrel{\mathsf{def}}{=} \lambda s \in st(T)\,.\begin{cases} 0 & s \in \mathsf{pre}(\mathsf{dom}(\alpha_\phi(T))) \\ \text{undefined} & \text{otherwise} \end{cases}$$

CTL Abstraction

Globally Formulas

$$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$$

$$\alpha_{\mathsf{AG}\phi}(T) \stackrel{\text{def}}{=} \mathsf{gfp}^{\preceq}_{\alpha_\phi(T)} \overline{F}_{\mathsf{AG}\phi}$$

$$\overline{F}_{\mathsf{AG}\phi}(f) \stackrel{\text{def}}{=} \lambda s\,.\begin{cases} f(s) & s \in \mathsf{dom}(f) \cap \widetilde{\mathsf{pre}}(\mathsf{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

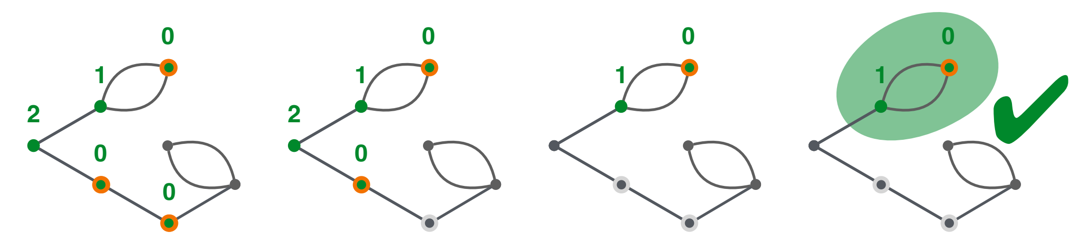
$$\alpha_{\mathsf{EG}\phi}(T) \stackrel{\text{def}}{=} \mathsf{gfp}^{\preceq}_{\alpha_\phi(T)} F_{\mathsf{EG}\phi}$$

$$F_{\mathsf{EG}\phi}(f) \stackrel{\text{def}}{=} \lambda s\,.\begin{cases} f(s) & s \in \mathsf{dom}(f) \cap \mathsf{pre}(\mathsf{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Recurrence Semantics

$$\mathscr{R}_R^{\varphi} \stackrel{\text{def}}{=} \mathsf{gfp}^{\preceq}_{\mathscr{R}_G^{\varphi}} \overline{F}_R$$

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



CTL Abstraction

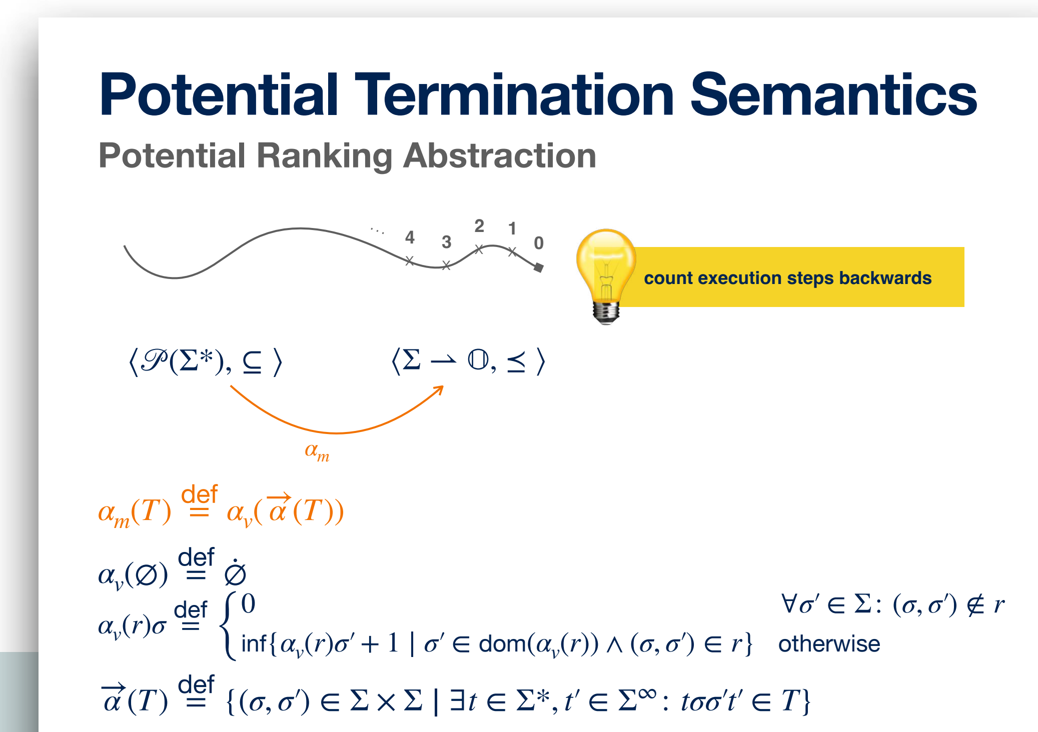
Until Formulas (1)

$$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$$

$$\alpha_{\mathsf{E}(\phi_1 \mathsf{U} \phi_2)}(T) \stackrel{\text{def}}{=} \alpha_v(\overrightarrow{\alpha}(\alpha^{sq}_{\mathsf{E}(\phi_1 \mathsf{U} \phi_2)}(T)))$$

$$\alpha^{sq}_{\mathsf{E}(\phi_1 \mathsf{U} \phi_2)}(T) \stackrel{\text{def}}{=} \overline{\alpha}_{\mathsf{E}(\phi_1 \mathsf{U} \phi_2)}[\mathsf{dom}(\alpha_{\phi_1}(T))][\mathsf{dom}(\alpha_{\phi_2}(T))]T$$

$$\overline{\alpha}_{\mathsf{E}(\phi_1 \mathsf{U} \phi_2)}[S_1][S_2]T \stackrel{\text{def}}{=} \left\{ \sigma s \in \mathsf{sq}(T) \;\middle|\; \sigma \in (S_1 \setminus S_2)^*, s \in S_2 \right\}$$



CTL Abstraction

Until Formulas (2)

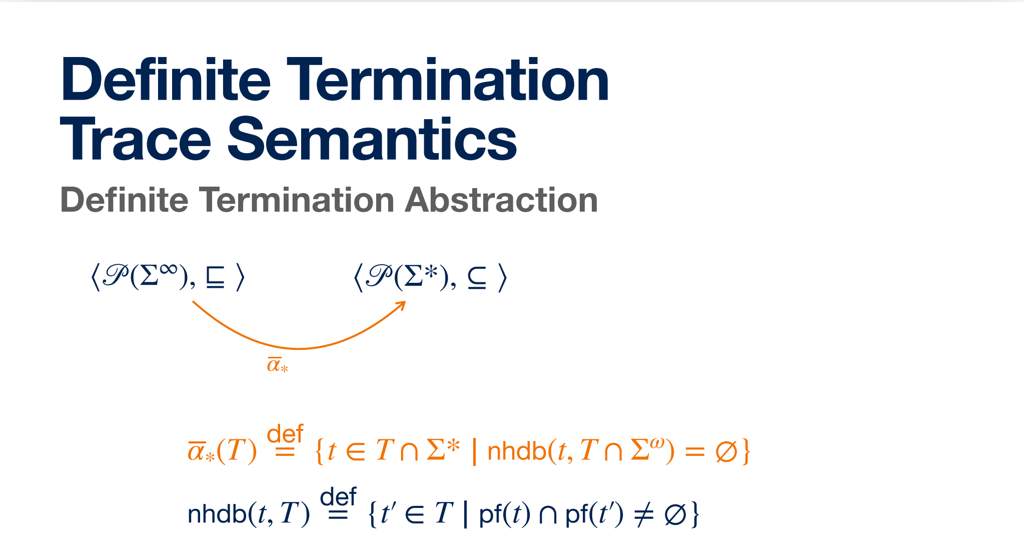
$$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\phi \mid \mathsf{AG}\phi \mid \mathsf{A}(\phi \mathsf{U} \phi) \mid \mathsf{EX}\phi \mid \mathsf{EG}\phi \mid \mathsf{E}(\phi \mathsf{U} \phi)$$

$$\alpha_{\mathsf{A}(\phi_1 \mathsf{U} \phi_2)}(T) \stackrel{\text{def}}{=} \overline{\alpha}_V(\overrightarrow{\alpha}(\alpha^{sq}_{\mathsf{A}(\phi_1 \mathsf{U} \phi_2)}(T)))$$

$$\alpha^{sq}_{\mathsf{A}(\phi_1 \mathsf{U} \phi_2)}(T) \stackrel{\text{def}}{=} \overline{\alpha}_{\mathsf{A}(\phi_1 \mathsf{U} \phi_2)}[\mathsf{dom}(\alpha_{\phi_1}(T))][\mathsf{dom}(\alpha_{\phi_2}(T))]T$$

$$\overline{\alpha}_{\mathsf{A}(\phi_1 \mathsf{U} \phi_2)}[S_1][S_2]T \stackrel{\text{def}}{=} \left\{ \sigma s \in \mathsf{sq}(T) \;\middle|\; \begin{array}{c} \sigma \in (S_1 \setminus S_2)^*, s \in S_2, \\ \mathsf{nbhd}(\sigma, \mathsf{sf}(T) \cap \overline{S_2}^{+\infty}) = \varnothing, \\ \mathsf{nbhd}(\sigma, \mathsf{sf}(T) \cap Z) = \varnothing \end{array} \right\}$$

$$Z \stackrel{\text{def}}{=} \{\sigma s \sigma' \in \Sigma^{+\infty} \mid \sigma \in \Sigma^* \wedge s \in \overline{S_1 \cup S_2} \wedge \sigma' \in \Sigma^{+\infty}\}$$



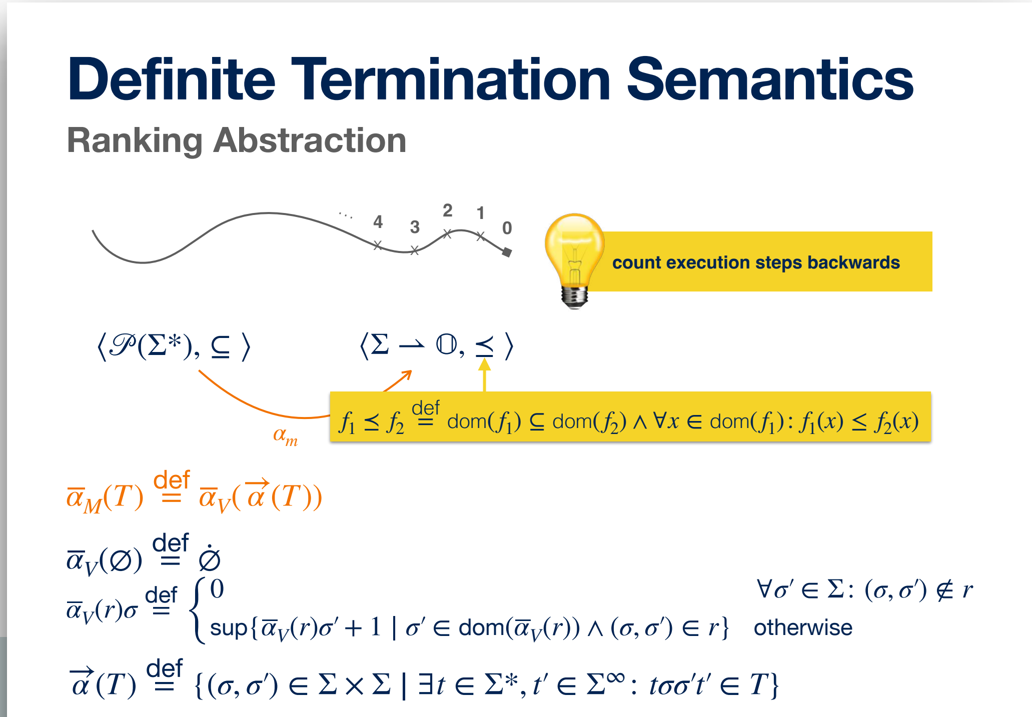
Definite Termination Trace Semantics

Definite Termination Abstraction

$$\langle \mathscr{P}(\Sigma^\infty), \sqsubseteq \rangle \xrightarrow{\overline{\alpha}_*} \langle \mathscr{P}(\Sigma^*), \subseteq \rangle$$

$$\overline{\alpha}_*(T) \stackrel{\text{def}}{=} \{t \in T \cap \Sigma^* \mid \mathsf{nhdb}(t, T \cap \Sigma^\omega) = \varnothing\}$$

$$\mathsf{nhdb}(t, T) \stackrel{\text{def}}{=} \{t' \in T \mid \mathsf{pf}(t) \cap \mathsf{pf}(t') \neq \varnothing\}$$



Definite Termination Semantics

Ranking Abstraction

$$\langle \mathscr{P}(\Sigma^*), \subseteq \rangle \xrightarrow{\alpha_m} \langle \Sigma \rightharpoonup \mathbb{O}, \preceq \rangle$$

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

$$\overline{\alpha}_M(T) \stackrel{\text{def}}{=} \overline{\alpha}_V(\overrightarrow{\alpha}(T))$$

$$\overline{\alpha}_V(\varnothing) \stackrel{\text{def}}{=} \dot{\varnothing}$$

$$\overline{\alpha}_V(r)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \forall \sigma' \in \Sigma \colon (\sigma, \sigma') \notin r \\ \sup\{\overline{\alpha}_V(r)\sigma' + 1 \mid \sigma' \in \mathsf{dom}(\overline{\alpha}_V(r)) \wedge (\sigma, \sigma') \in r\} & \text{otherwise} \end{cases}$$

$$\overrightarrow{\alpha}(T) \stackrel{\text{def}}{=} \{(\sigma, \sigma') \in \Sigma \times \Sigma \mid \exists t \in \Sigma^*, t' \in \Sigma^\infty \colon t\sigma\sigma't' \in T\}$$

CTL Program Semantics

Definition

Given a CTL formula ϕ
and the corresponding CTL abstraction $\alpha_\phi \colon \mathscr{P}(\Sigma^\infty) \to (\Sigma \rightharpoonup \mathbb{O})$,
the **program semantics** $\mathscr{R}_\phi \colon \Sigma \rightharpoonup \mathbb{O}$ **for** ϕ is defined as:

$$\mathscr{R}\phi \stackrel{\text{def}}{=} \alpha_\phi(\mathscr{M}_\infty)$$

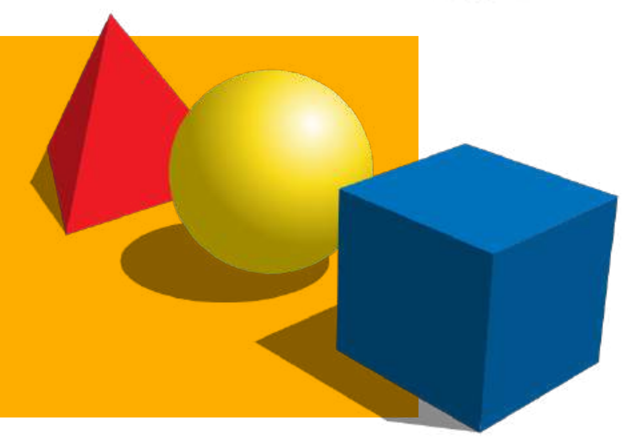
Theorem

A program satisfies a **CTL property** ϕ for traces starting
from a set of initial states $\mathscr{I}$ if and only if $\mathscr{I} \subseteq \mathrm{dom}(\mathscr{R}_\phi)$

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Abstract Interpretation of CTL Properties

Caterina Urban, Samuel Ueltschi, and Peter Müller

Department of Computer Science
ETH Zurich, Switzerland

Abstract. CTL is a temporal logic commonly used to express program properties. Most of the existing approaches for proving CTL properties only support certain classes of programs, limit their scope to a subset of CTL, or do not directly support certain existential CTL formulas. This paper presents an abstract interpretation framework for proving CTL properties that does not suffer from these limitations. Our approach automatically infers sufficient preconditions, and thus provides useful information even when a program satisfies a property only for some inputs.
We systematically derive a program semantics that precisely captures CTL properties by abstraction of the operational trace semantics of a program. We then leverage existing abstract domains based on piecewise-defined functions to derive decidable abstractions that are suitable for static program analysis. To handle existential CTL properties, we augment these abstract domains with under-approximating operators.
We implemented our approach in a prototype static analyzer. Our experimental evaluation demonstrates that the analysis is effective, even for CTL formulas with non-trivial nesting of universal and existential path quantifiers, and performs well on a wide variety of benchmarks.

1 Introduction

Computation tree logic (CTL) [6] is a temporal logic introduced by Clarke and Emerson to overcome certain limitations of linear temporal logic (LTL) [33] for program specification purposes. Most of the existing approaches for proving program properties expressed in CTL have limitations that restrict their applicability: they are limited to finite-state programs [7] or to certain classes of infinite-state programs (e.g., pushdown systems [36]), they limit their scope to a subset of CTL (e.g., the universal fragment of CTL [11]), or support existential path quantifiers only indirectly by considering their universal dual [8].

In this paper, we propose a new static analysis method for proving CTL properties that does not suffer from any of these limitations. We set our work in the framework of *abstract interpretation* [16], a general theory of semantic approximation that provides a basis for various successful industrial-scale tools (e.g., Astrée [3]). We generalize an existing abstract interpretation framework for proving termination [18] and other liveness properties [41].

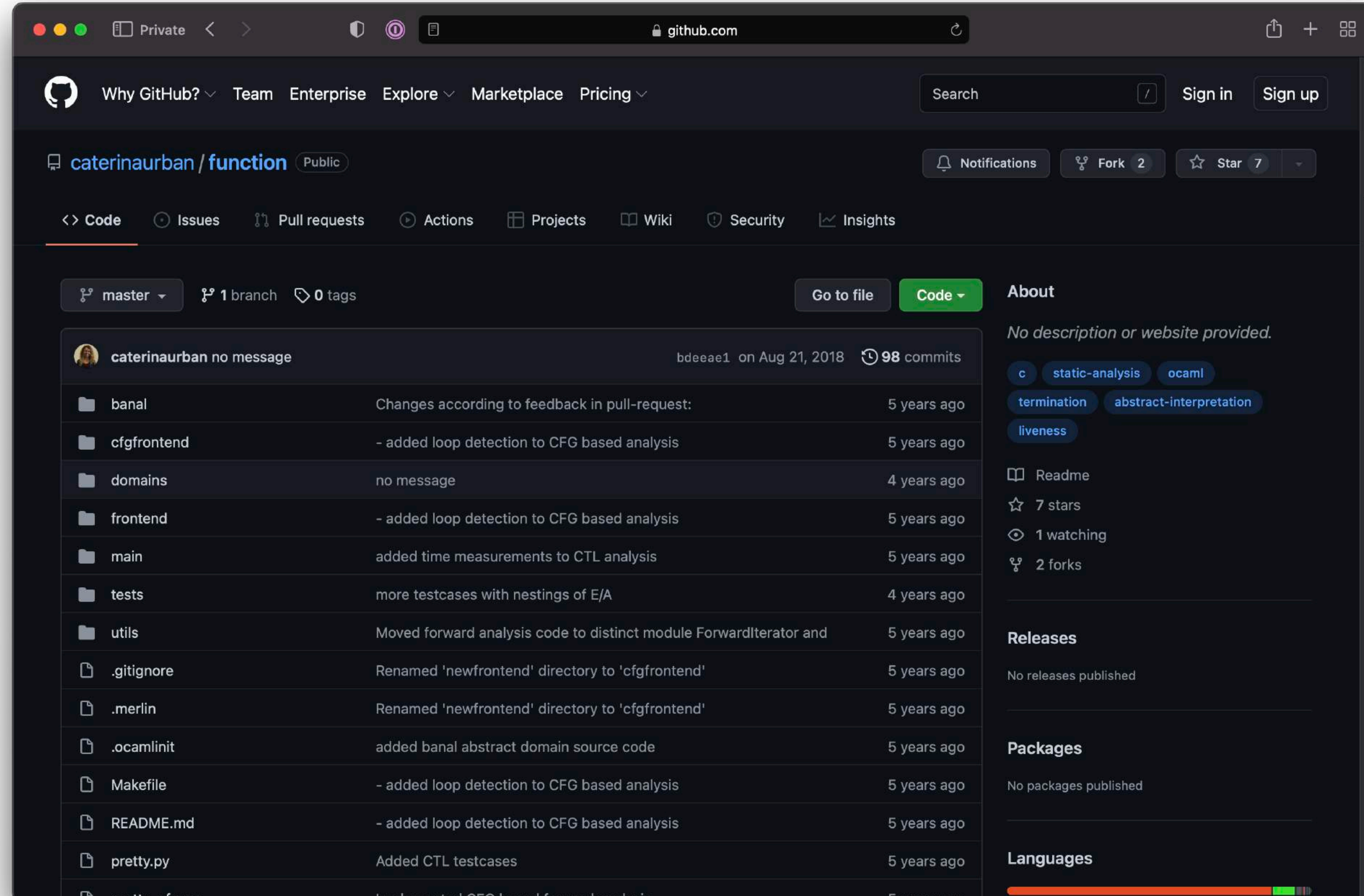
Following the theory of abstract interpretation [14], we abstract away from irrelevant details about the execution of a program and systematically derive a program semantics that is *sound and complete* for proving a CTL property.

Abstract Interpretation Recipe

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior



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
bdeeae1 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.