Analysis of CTL Properties

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

Caterina Urban

November 18th, 2024

Year 2024-2025

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

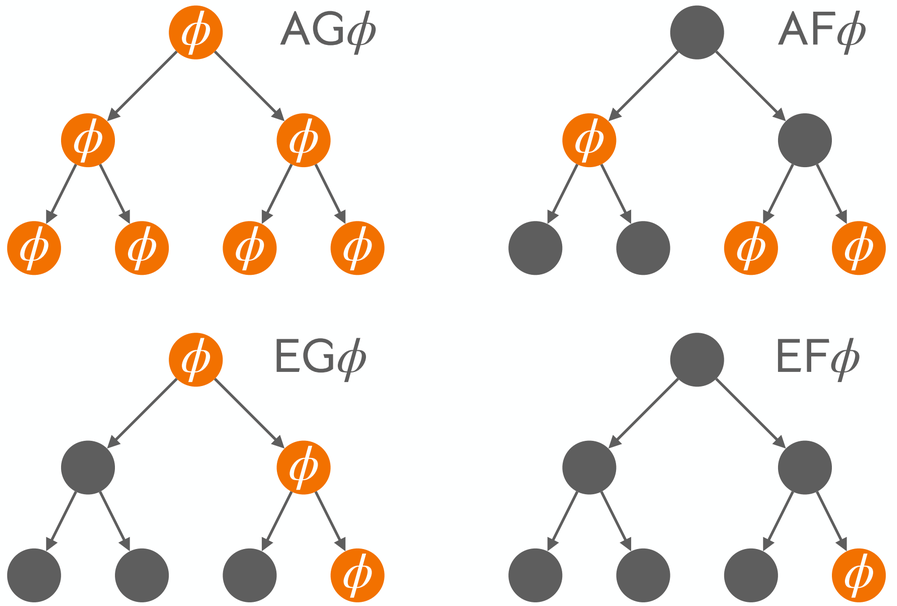

Amir Pnueli

Computation Tree Logic (CTL)

Branching Temporal Logic

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

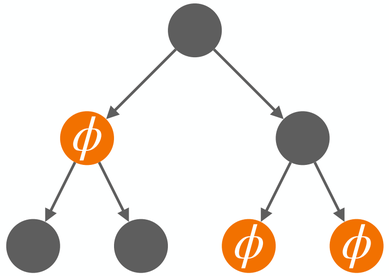
$$\mathsf{AF}\phi \equiv \mathsf{A}(\textbf{true}\ \mathsf{U}\ \phi) \qquad \mathsf{EF}\phi \equiv \mathsf{E}(\textbf{true}\ \mathsf{U}\ \phi)$$



Guarantee Properties

Guarantee Properties

"something good eventually happens <u>at least once</u>"



AF ϕ

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

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

Guarantee Semantics

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Guarantee Semantics

Definite Termination Semantics

$$\mathscr{R}_M = \mathsf{lfp}^{\preceq} F_M$$

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

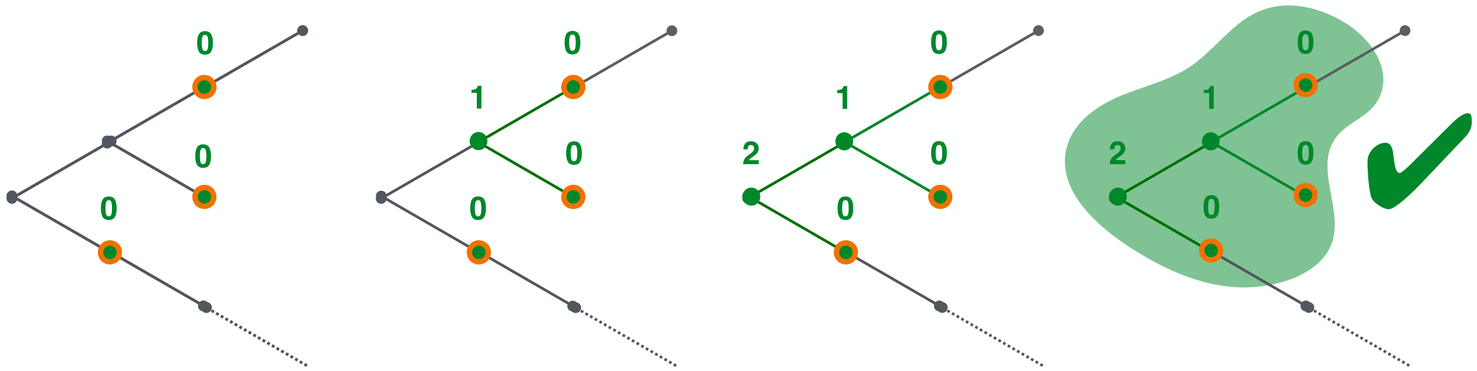
$$\mathscr{R}^{\varphi}_G \stackrel{\mathsf{def}}{=} \mathsf{lfp}^{\preceq} \overline{F}_G \left[\{\sigma \in \Sigma \mid \sigma \vDash \varphi\}\right]$$

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

Guarantee Semantics

$$\mathscr{R}^{\varphi}_{G} \stackrel{\text{def}}{=} \text{lfp}^{\leq} \overline{F}_{G}\left[\{\sigma \in \Sigma \mid \sigma \vDash \varphi\}\right]$$

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



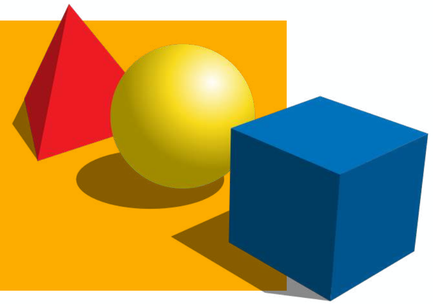
Theorem (Soudness and Completeness)

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

Abstract Guarantee Semantics

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Abstract Guarantee Semantics

Piecewise-Defined Ranking Functions Abstract Domain

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

- $\mathscr{R}^{\varphi\#}_G[\![{}^{\ell} X \leftarrow e]\!]t \stackrel{\text{def}}{=} \text{RESET}^G_A[\![\varphi]\!](\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]t)$
- $\mathscr{R}^{\varphi\#}_G[\![\textbf{if } {}^{\ell} e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=} \text{RESET}^G_A[\![\varphi]\!](\text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_G[\![s]\!]t) \curlyvee_T \text{FILTER}_A[\![e \not\bowtie 0]\!]t))$
- $\mathscr{R}^{\varphi\#}_G[\![\textbf{while } {}^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]t \stackrel{\text{def}}{=} \text{lfp}^{\#}\overline{F}^{\varphi\#}_G$
  where $\overline{F}^{\varphi\#}_G(x) \stackrel{\text{def}}{=} \text{RESET}^G_A[\![\varphi]\!](\text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_G[\![s]\!]x) \curlyvee_T \text{FILTER}_A[\![e \not\bowtie 0]\!](t)))$
- $\mathscr{R}^{\varphi\#}_G[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathscr{R}^{\varphi\#}_G[\![s_1]\!](\mathscr{R}^{\varphi\#}_G[\![s_2]\!]t)$

Abstract Guarantee Semantics

Piecewise-Defined Ranking Functions Abstract Domain

Definition

The **abstract guaranteee semantics** $\mathscr{R}^{\varphi\#}_{G}[\![\mathsf{stat}^{\ell}]\!] \in \mathscr{A}$ of a program stat^{ℓ} is:

$$\mathscr{R}^{\varphi\#}_{G}[\![\mathsf{stat}^{\ell}]\!] \stackrel{\text{def}}{=} \mathscr{R}^{\varphi\#}_{G}[\![\mathsf{stat}]\!](\mathsf{RESET}^{G}_{A}[\![\varphi]\!](\mathsf{LEAF}: \bot_{F}))$$

where $\mathscr{R}^{\varphi\#}_{G}[\![\mathsf{stat}]\!] : \mathscr{A} \to \mathscr{A}$ is the abstract guarantee semantics of each program instruction stat

Theorem (Soundness)

$$\mathscr{R}_{G}[\![\mathsf{stat}^{\ell}]\!] \preccurlyeq \gamma_{A}(\mathscr{R}^{\#}_{G}[\![\mathsf{stat}^{\ell}]\!])$$

Corollary (Soundness)

A program stat^{ℓ} satisfies a **guarantee property** AF φ for traces starting from a set of initial states $\mathscr{I}$ if $\mathscr{I} \subseteq \mathrm{dom}(\gamma_{A}(\mathscr{R}^{\varphi\#}_{G}[\![\mathsf{stat}^{\ell}]\!]))$

Abstract Guarantee Semantics

Example

Example

int : $x,\ y$

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

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

od

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

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

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

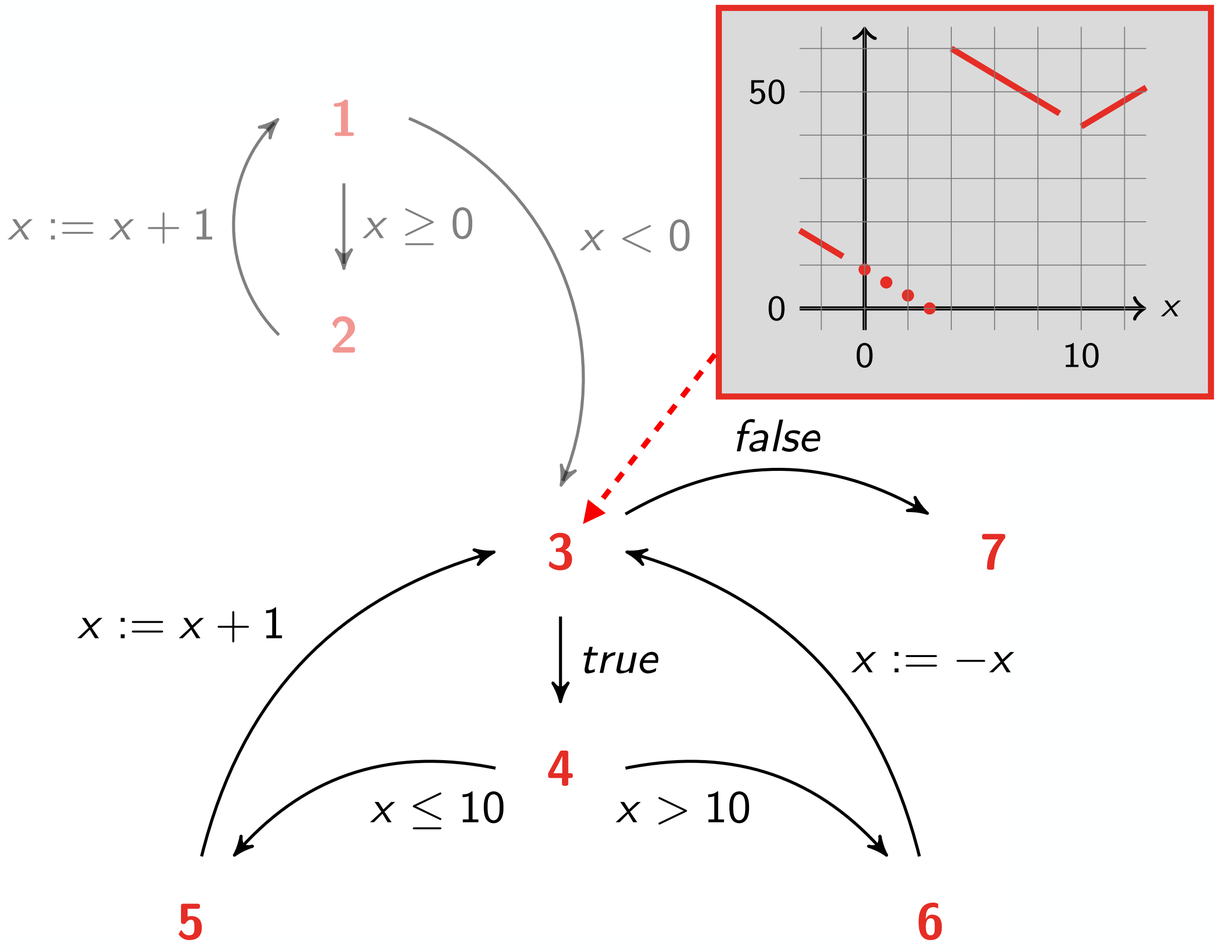
  else

    $^{6}x := -x$

od^{7}

Property

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



Abstract Guarantee Semantics

Example

int : x, y

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

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

od

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

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

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

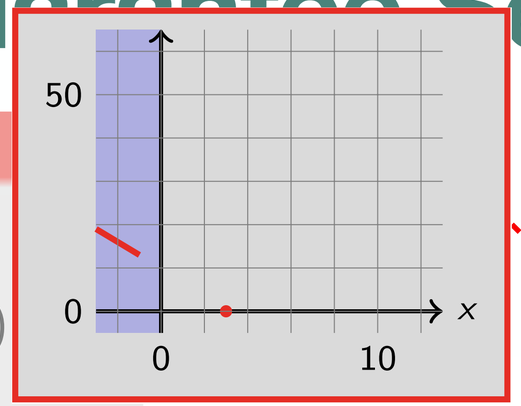
  else

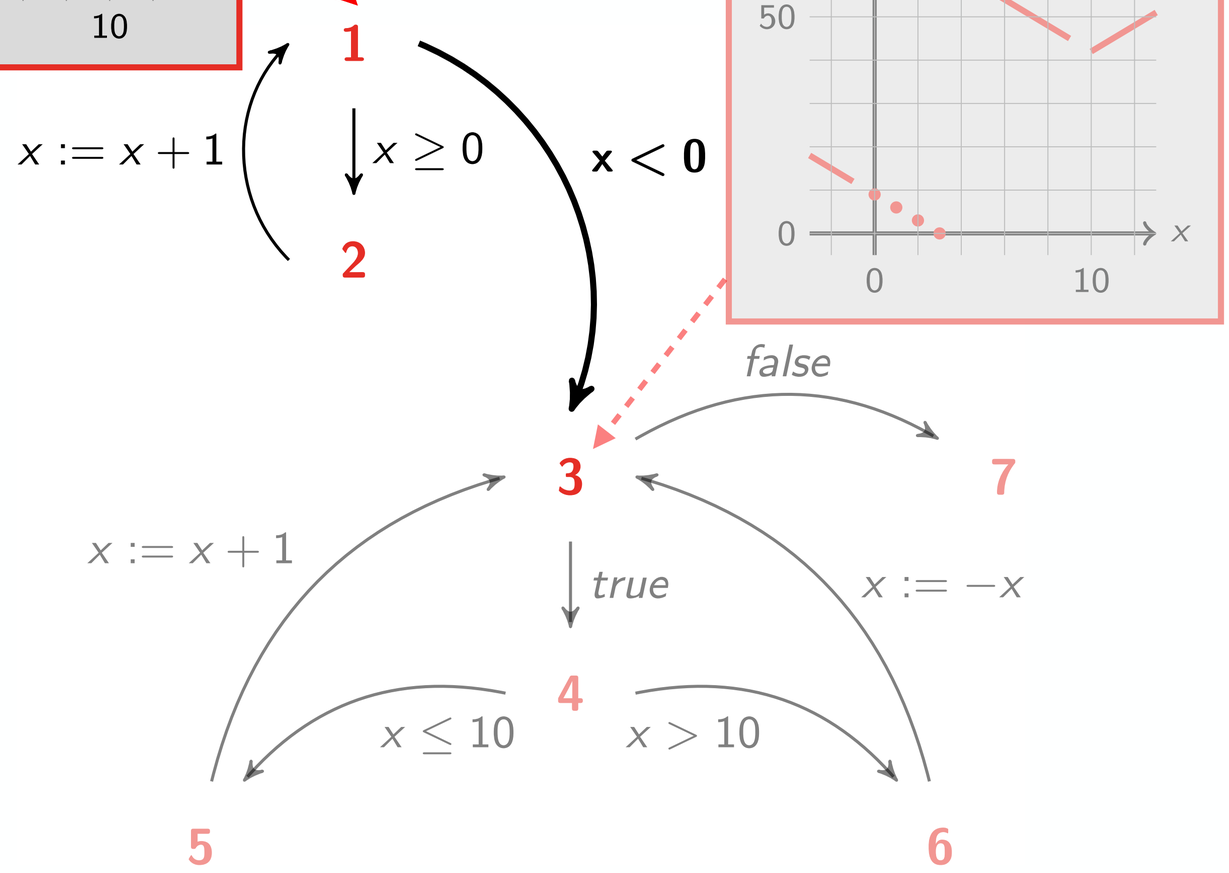
    $^{6}x := -x$

od^{7}

Property

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





Abstract Guarantee Semantics

Example

int : $x,\ y$

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

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

od

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

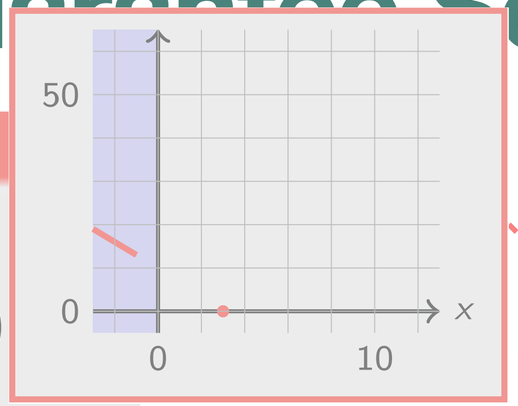
  if $^{4}(\ x \leq 10\ )$

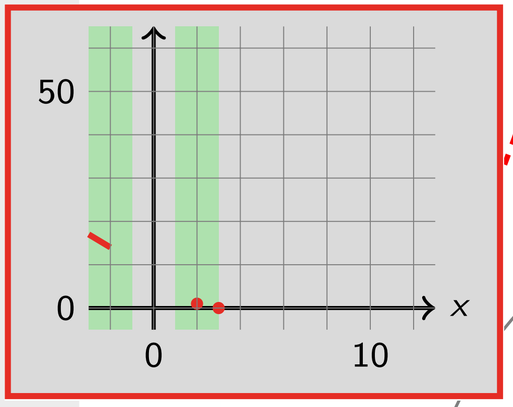
    $^{5}x := x + 1$

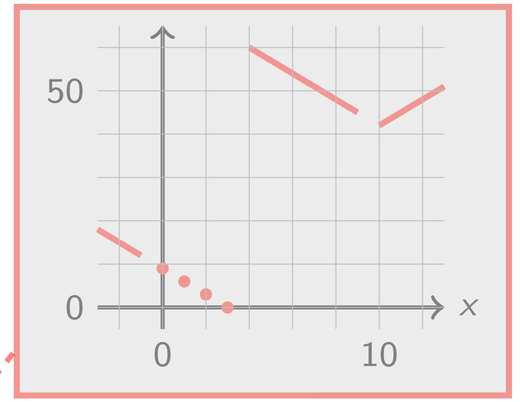
  else

    $^{6}x := -x$

od^{7}







Property

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

Abstract Guarantee Semantics

Example

int : $x,\ y$

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

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

od

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

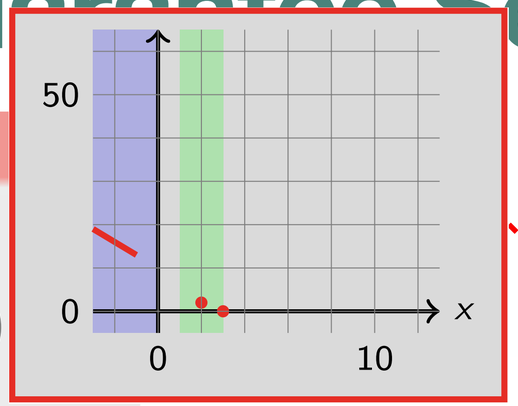
  if $^{4}(\ x \leq 10\ )$

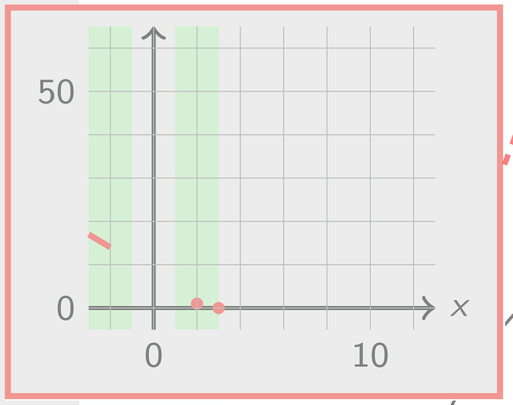
    $^{5}x := x + 1$

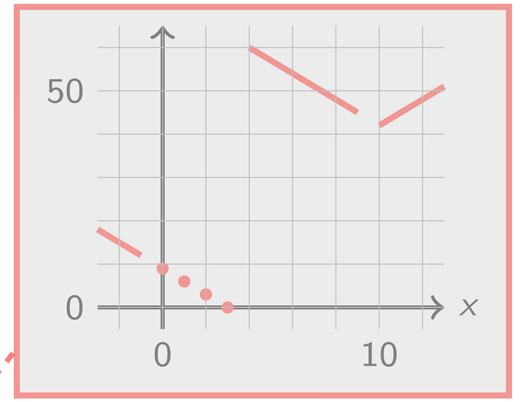
  else

    $^{6}x := -x$

od^{7}







Property

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

Abstract Guarantee Semantics

Example

int : $x,\ y$

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

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

od

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

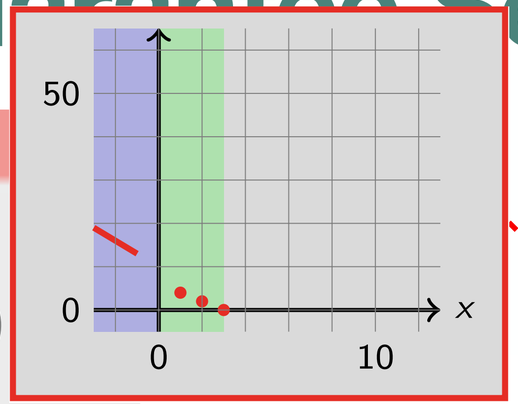
  if $^{4}(\ x \leq 10\ )$

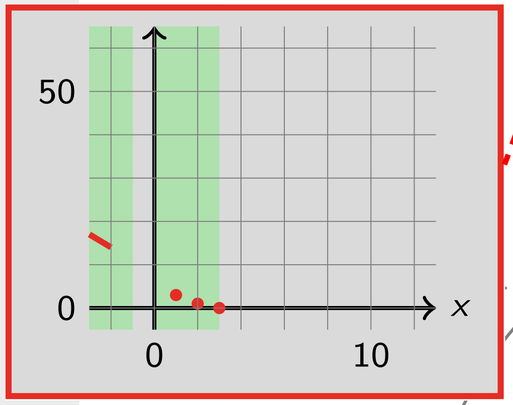
    $^{5}x := x + 1$

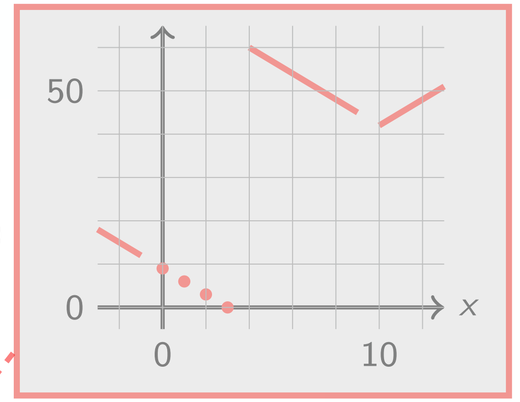
  else

    $^{6}x := -x$

od^{7}







Property

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

Abstract Guarantee Semantics

Example

int : x, y

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

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

od

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

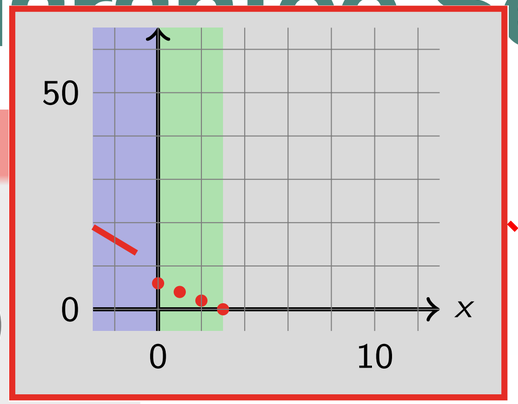
  if $^{4}(\ x \leq 10\ )$

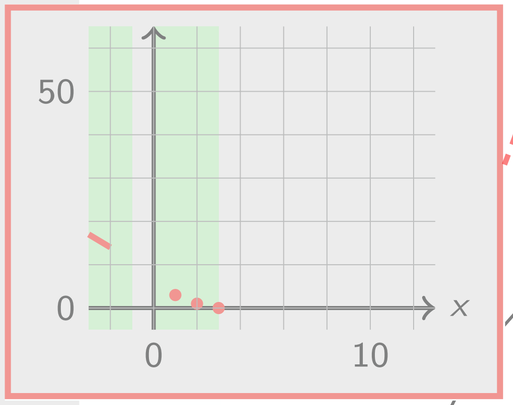
    $^{5}x := x + 1$

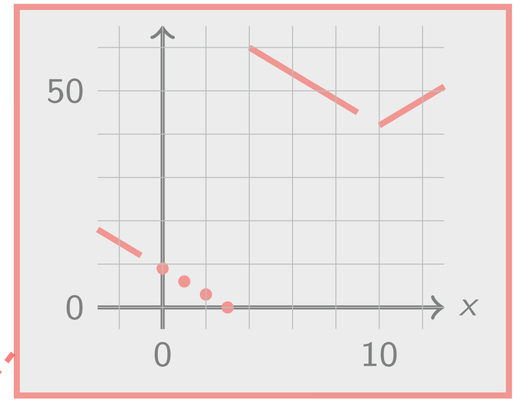
  else

    $^{6}x := -x$

od^{7}







Property

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

Abstract Guarantee Semantics

Example

int : x, y

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

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

od

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

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

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

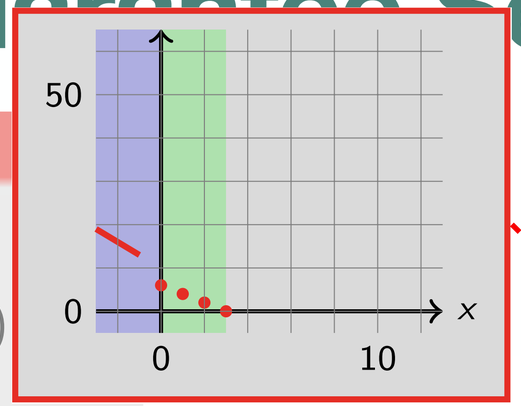
  else

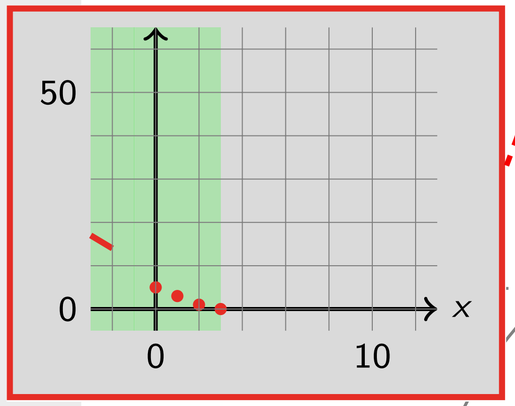
    $^{6}x := -x$

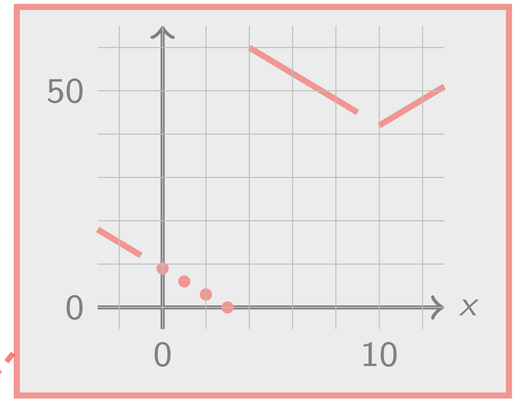
od^{7}

Property

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







Abstract Guarantee Semantics

Example

int : x, y

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

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

od

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

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

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

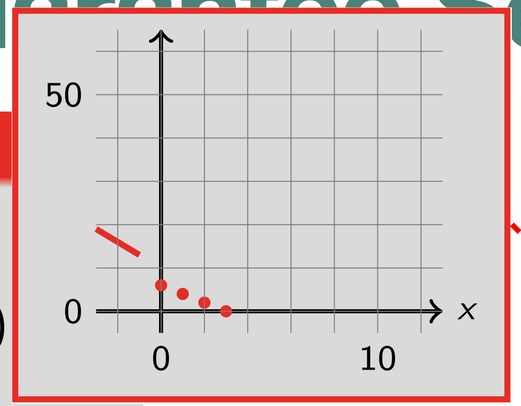
  else

    $^{6}x := -x$

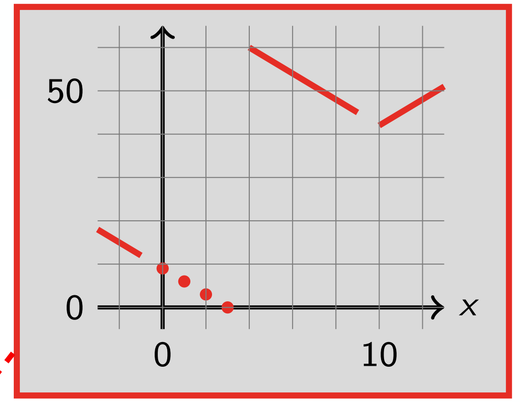
od^{7}

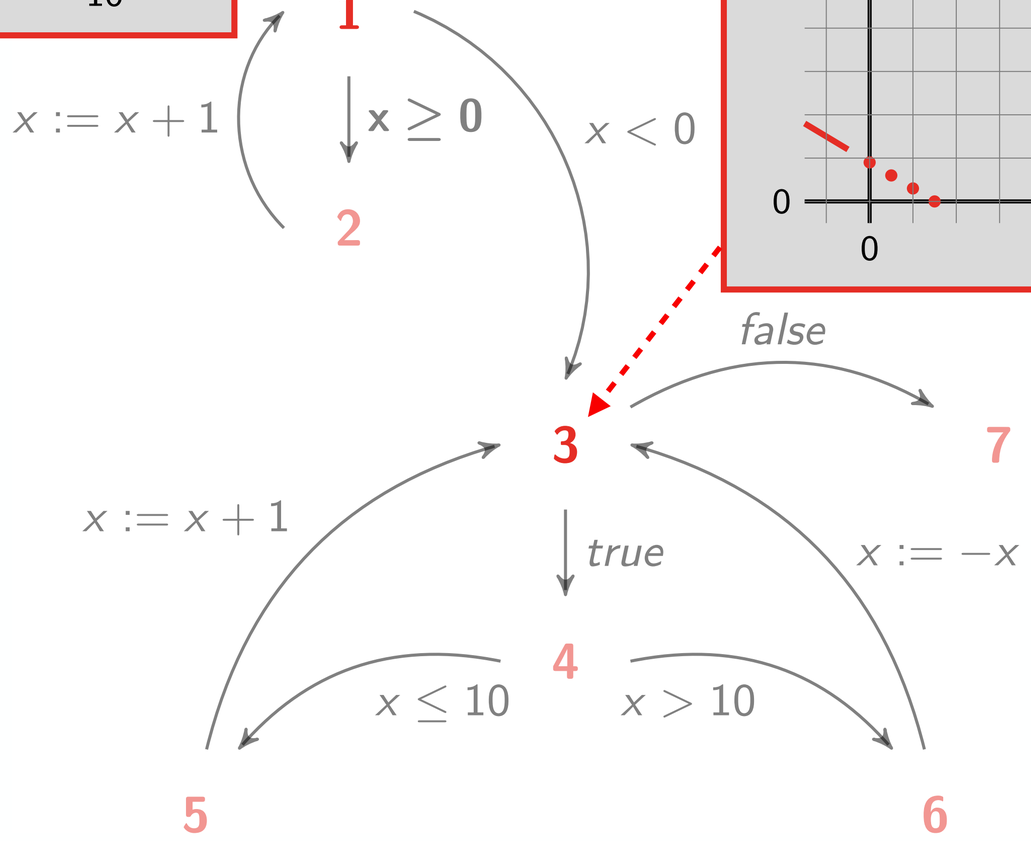
Property

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



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





Recurrence Properties

Recurrence Properties

"something good eventually happens infinitely often"

AG AF ϕ

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

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 $\mathscr{I} \stackrel{\text{def}}{=} \{(1,\rho) \in \Sigma \mid \rho(x) < 0\}$

Recurrence Semantics

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior

Recurrence Semantics

Guarantee Semantics

$$\mathscr{R}^{\varphi}_{G} \stackrel{\text{def}}{=} \text{lfp}^{\preceq}\, \overline{F}_{G}\left[\{\sigma \in \Sigma \mid \sigma \vDash \varphi\}\right]$$

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

$$\mathscr{R}^{\varphi}_{R} \stackrel{\text{def}}{=} \text{gfp}_{\mathscr{R}^{\varphi}_{G}}^{\preceq}\, \overline{F}_{R}$$

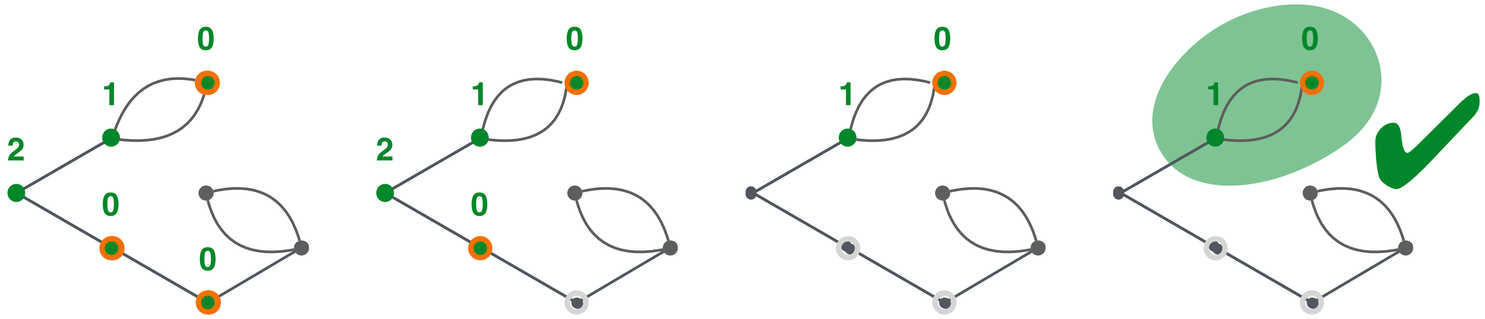
build upon the semantics of sub-formulas

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

Recurrence Semantics

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

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



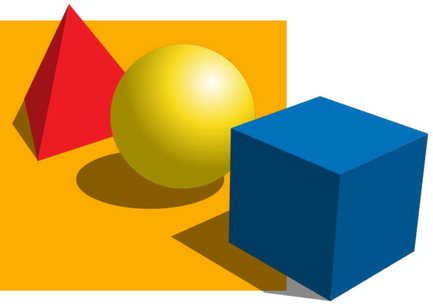
Theorem (Soundness and Completeness)

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

Abstract Recurrence Semantics

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Abstract Recurrence Semantics

Piecewise-Defined Ranking Functions Abstract Domain

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

- $\mathscr{R}^{\varphi\#}_{R}[\![^{\ell} X \leftarrow e]\!]t \stackrel{\text{def}}{=} \text{RESET}^{R}_{A}[\![\varphi]\!](\overleftarrow{\text{ASSIGN}}_{A}[\![X \leftarrow e]\!]t)$
- $\mathscr{R}^{\varphi\#}_{R}[\![\textbf{if } ^{\ell} e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=} \text{RESET}^{R}_{A}[\![\varphi]\!](\text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_{G}[\![s]\!]t) \curlyvee_{T} \text{FILTER}_{A}[\![e \not\bowtie 0]\!]t))$
- $\mathscr{R}^{\varphi\#}_{R}[\![\textbf{while } ^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]t \stackrel{\text{def}}{=} \text{gfp}^{\#}_{G(t)} \overline{F}^{\varphi\#}_{R}$
  where $G \stackrel{\text{def}}{=} \mathscr{R}^{\varphi\#}_{G}[\![\textbf{while } ^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]$
  and $\overline{F}^{\varphi\#}_{R}(x) \stackrel{\text{def}}{=} \text{RESET}^{R}_{A}[\![\varphi]\!](\text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_{R}[\![s]\!]x) \curlyvee_{T} \text{FILTER}_{A}[\![e \not\bowtie 0]\!](t)))$
- $\mathscr{R}^{\varphi\#}_{R}[\![s_1; s_2]\!]t \stackrel{\text{def}}{=} \mathscr{R}^{\varphi\#}_{R}[\![s_1]\!](\mathscr{R}^{\varphi\#}_{R}[\![s_2]\!]t)$

Dual Widening

Definition

Let $\langle \mathscr{D}, \sqsubseteq \rangle$ be a poset. A **dual widening** $\overline{\nabla} \colon \mathscr{D} \times \mathscr{D} \to \mathscr{D}$ is such that:

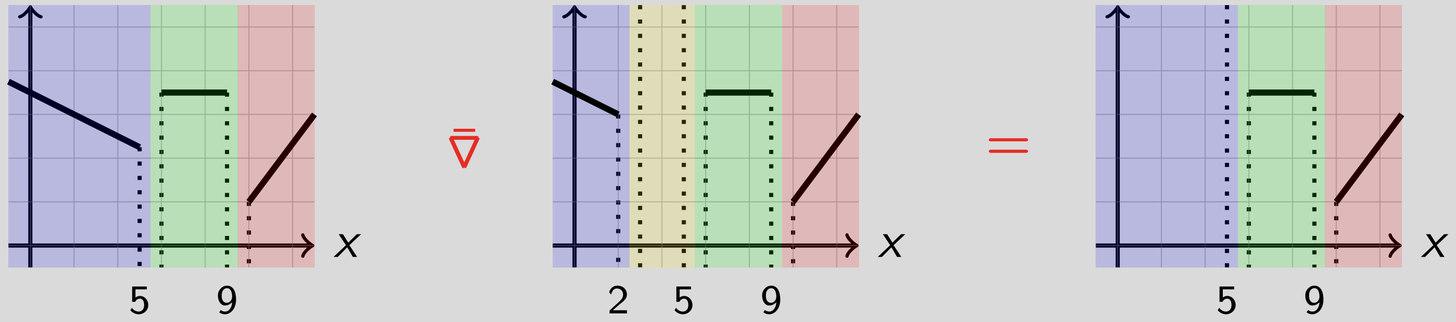
(i) for all elements $x, y \in \mathscr{D}$ we have $x \sqsupseteq x \mathbin{\overline{\nabla}} y$ and $y \sqsupseteq x \mathbin{\overline{\nabla}} y$

(ii) for all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq \ldots \sqsupseteq x_n \sqsupseteq \ldots$ the chain

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

is ultimately stationary

Example



Abstract Guarantee Semantics

Piecewise-Defined Ranking Functions Abstract Domain

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

Theorem (Soundness)

$$\mathscr{R}_G[\![\text{stat}^\ell]\!] \preccurlyeq \gamma_A(\mathscr{R}_G^{\#}[\![\text{stat}^\ell]\!])$$

Corollary (Soundness)

A program stat^ℓ satisfies a **recurrence property** AGAF 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 Recurrence Semantics

Example

Example

int : $x,\ y$

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

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

od

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

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

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

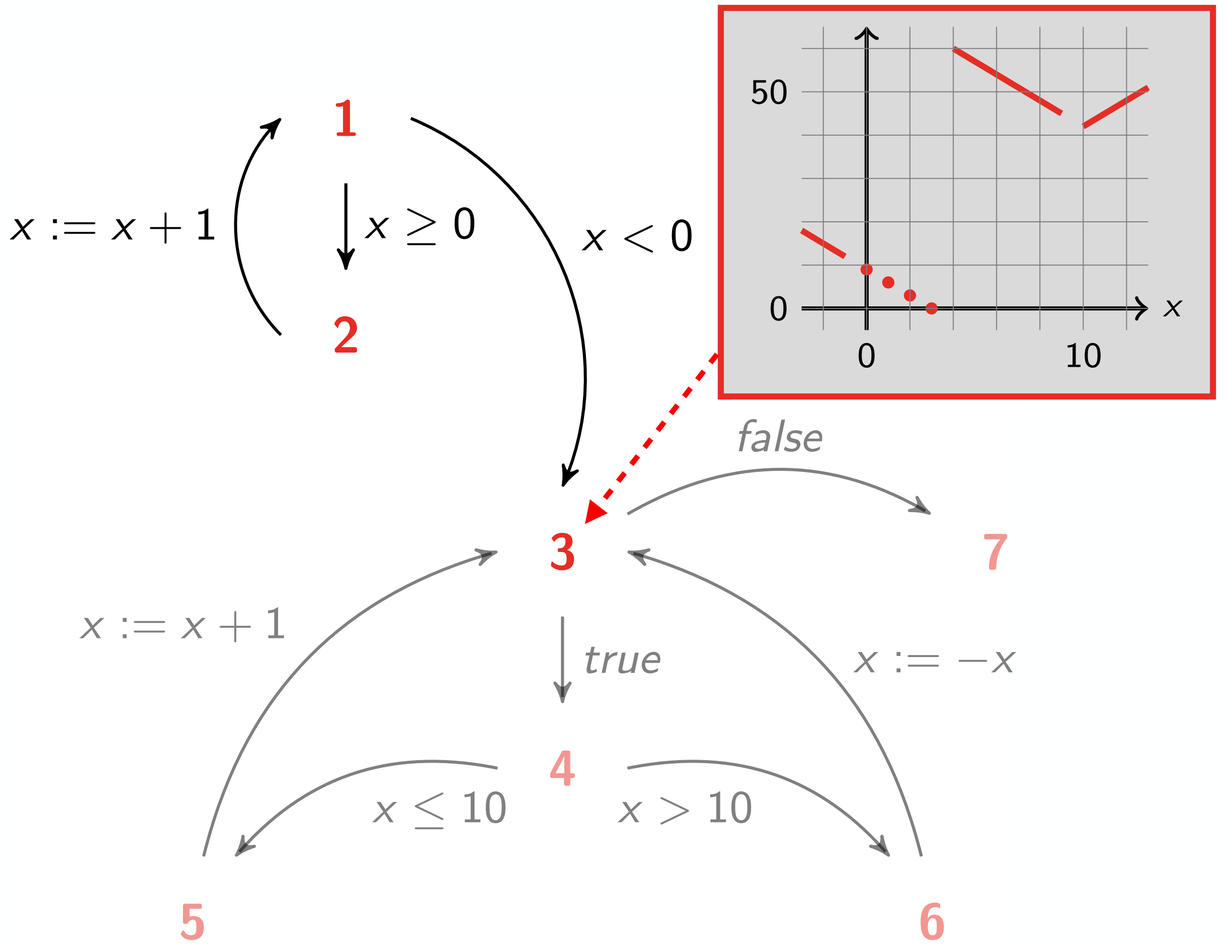
  else

    $^{6}x := -x$

od^{7}

Property

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



Abstract Recurrence Semantics

Example

int : x, y

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

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

od

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

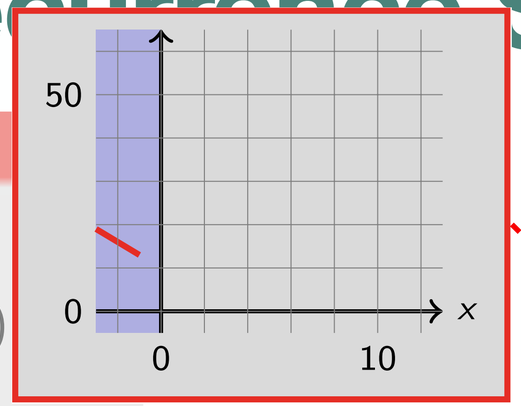
  if $^{4}(\ x \leq 10\ )$

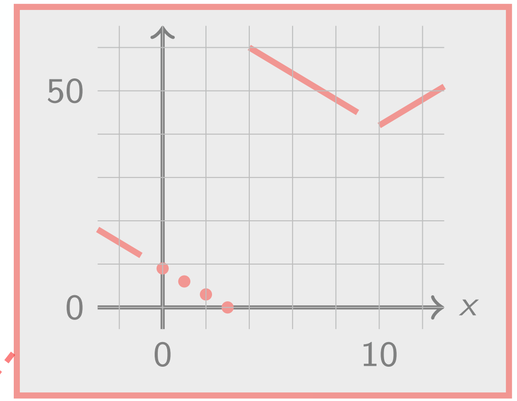
    $^{5}x := x + 1$

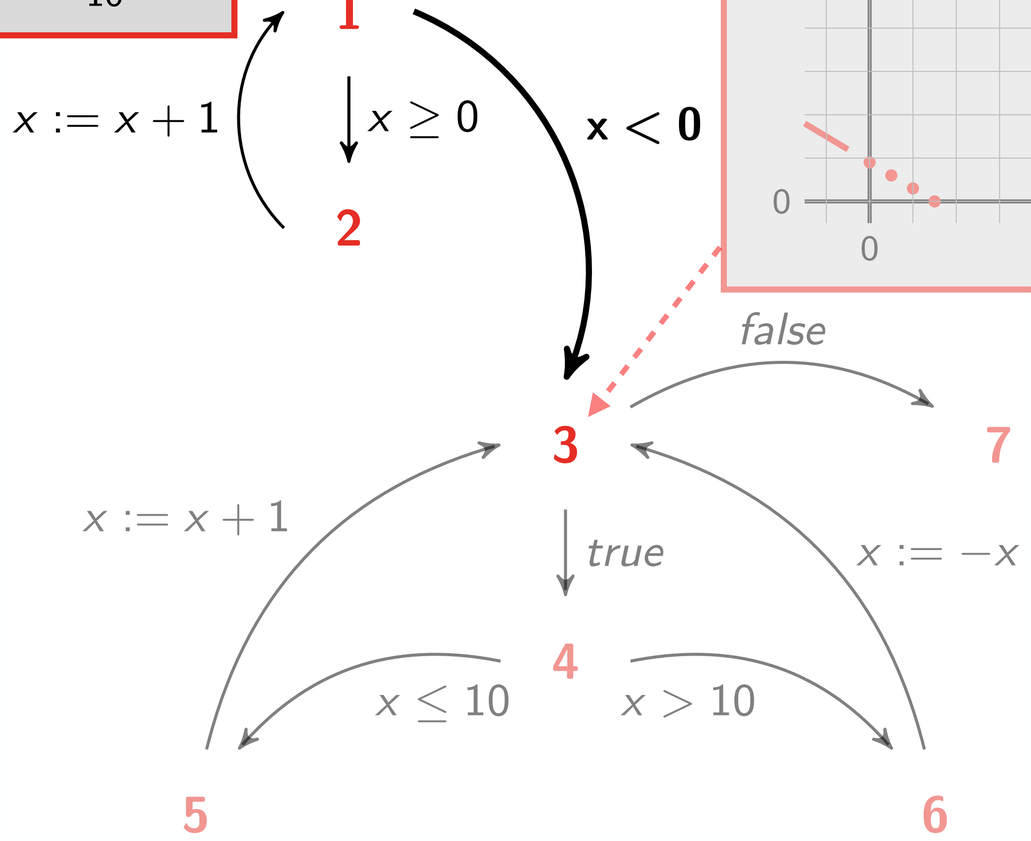
  else

    $^{6}x := -x$

od^{7}





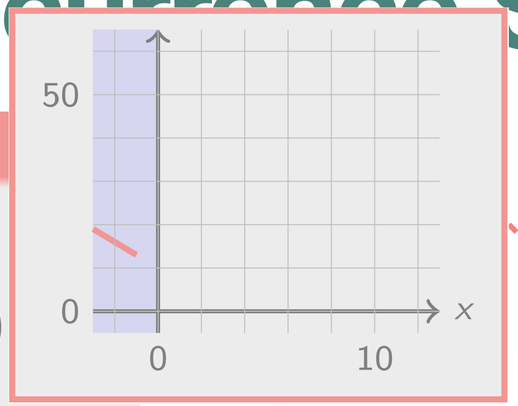


Property

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

Abstract Recurrence Semantics

Example



Example

int : x, y

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

  [2] $x := x + 1$

od

while [3] (true) do

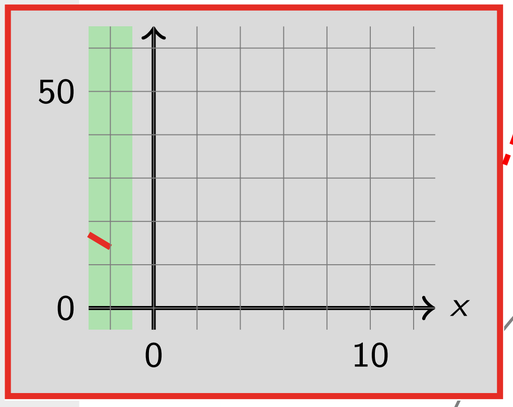
  if [4] ($x \leq 10$)

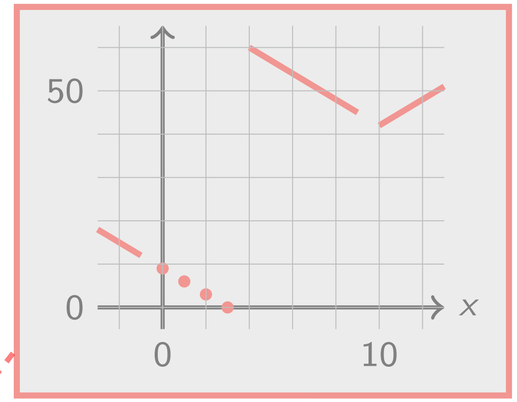
    [5] $x := x + 1$

  else

    [6] $x := -x$

od [7]





Property

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

Abstract Recurrence Semantics

Example

int : x, y

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

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

od

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

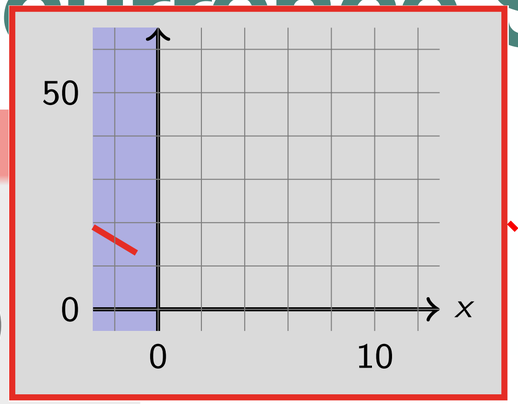
  if $^{4}(\ x \leq 10\ )$

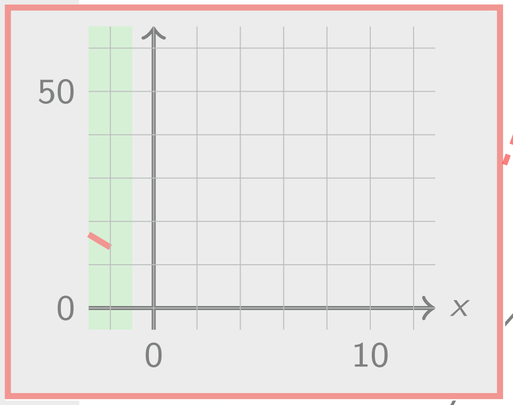
    $^{5}x := x + 1$

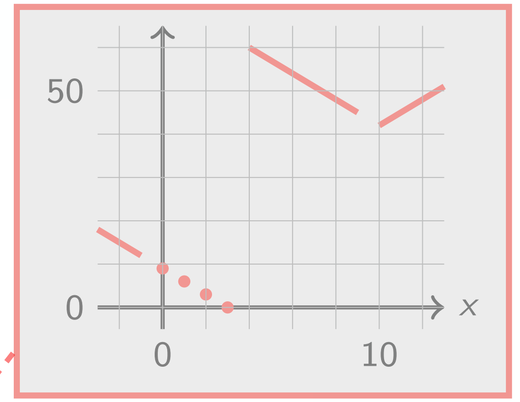
  else

    $^{6}x := -x$

od^{7}







Property

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

Abstract Recurrence Semantics

Example

int : x, y

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

  [2] $x := x + 1$

od

while [3] (true) do

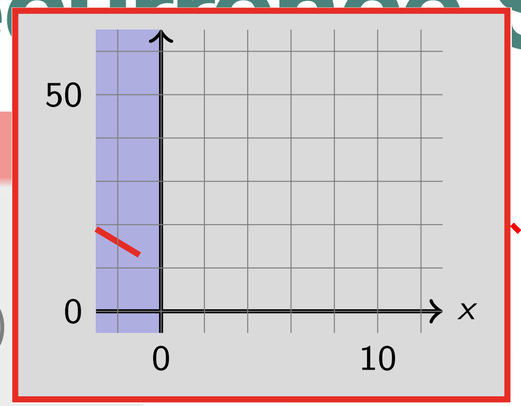
  if [4] ($x \leq 10$)

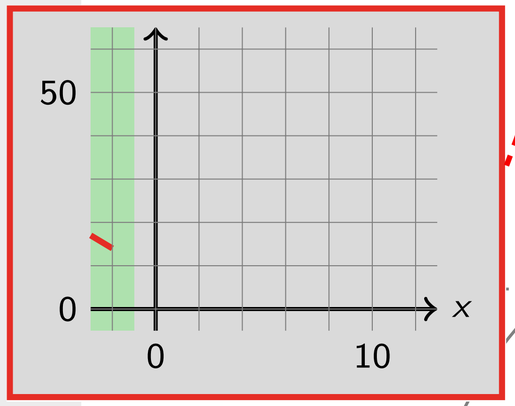
    [5] $x := x + 1$

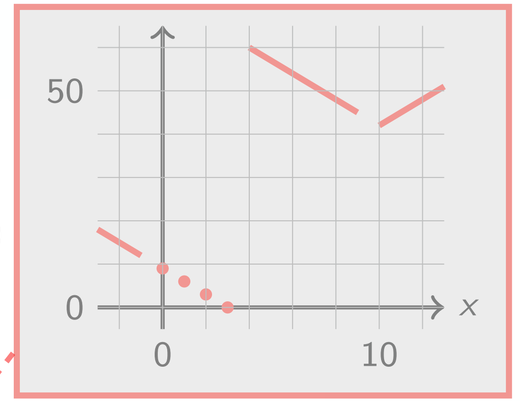
  else

    [6] $x := -x$

od [7]







Property

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

Abstract Recurrence Semantics

Example

int : x, y

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

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

od

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

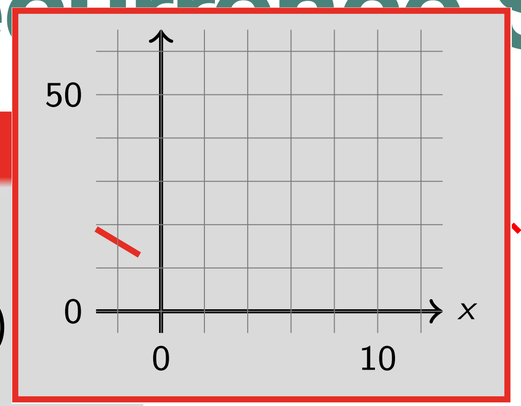
  if $^{4}(\ x \leq 10\ )$

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

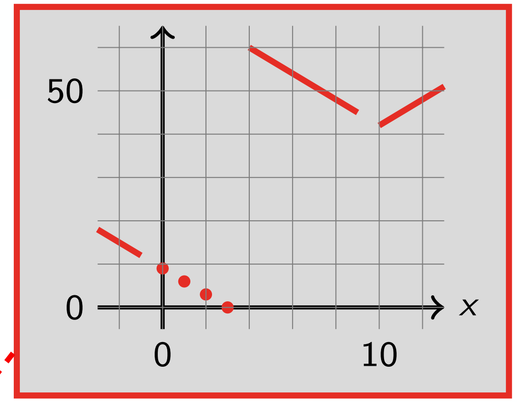
  else

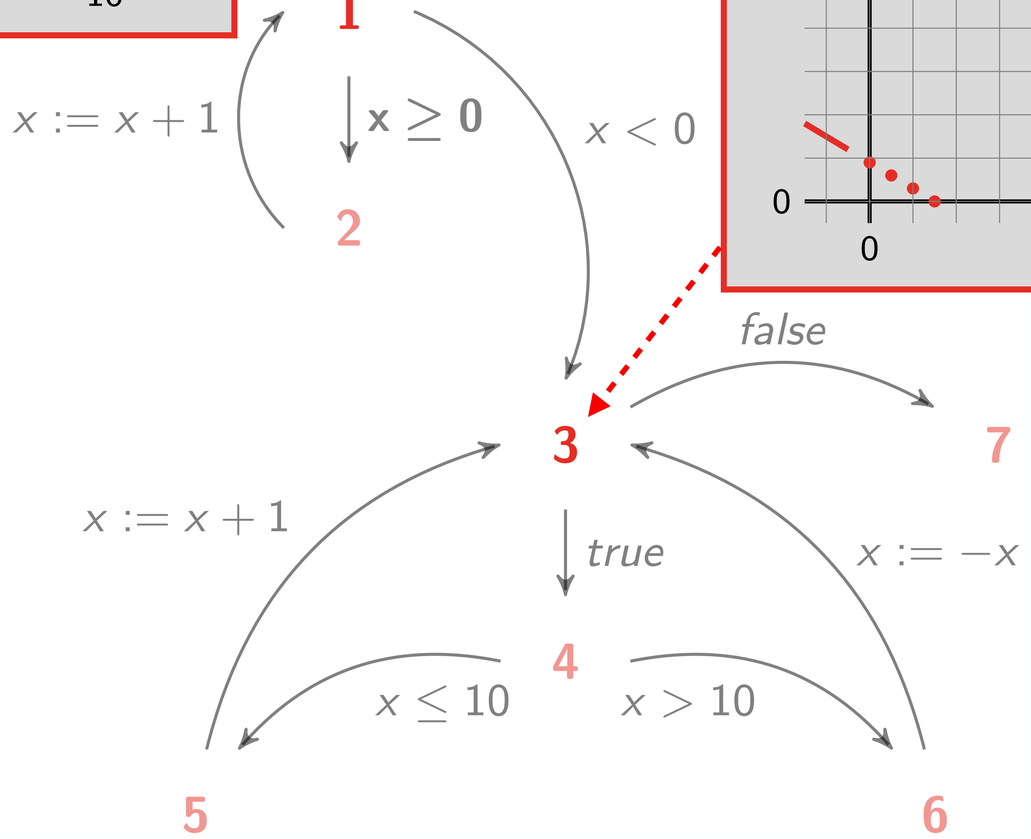
    $^{6}x := -x$

od^{7}



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





Property

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

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**"

AG AF ϕ

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

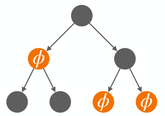
Example:

```
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 $\mathscr{T} \stackrel{\text{def}}{=} \{(1,\rho) \in \Sigma \mid \rho(x) < 0\}$

Guarantee Properties

"something good eventually happens **at least once**"



AF ϕ

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

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 $\mathscr{T} \stackrel{\text{def}}{=} \{(1,\rho) \in \Sigma \mid \rho(x) \leq 3\}$

CTL Program Semantics

practical tools
targeting specific p

algorithmic appro
to decide program

Recurrence Semantics

Guarantee Semantics

$$\mathscr{R}^{\varphi}_{G} \stackrel{\text{def}}{=} \text{lfp}^{\leq} \overline{F}_{G}\left[\{\sigma \in \Sigma \mid \sigma \vDash \varphi\}\right]$$

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

$$\mathscr{R}^{\varphi}_{R} \stackrel{\text{def}}{=} \text{gfp}_{\mathscr{R}^{\varphi}_{G}}^{\leq} \overline{F}_{R}$$

build upon the semantics of sub-formulas

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

mathematical models
of the program behavior

CTL Abstraction

Atomic Propositions

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

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

CTL Abstraction

Negation Formulas

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

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

CTL Abstraction

Conjunction Formulas

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

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

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

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

CTL Abstraction

Disjunction Formulas

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

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

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

CTL Abstraction

Next Formulas

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

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

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

CTL Abstraction

Globally Formulas

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

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

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

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

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

CTL Abstraction

Until Formulas (1)

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

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

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

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

$$\mathsf{sq}(T) \stackrel{\text{def}}{=} \{\sigma \in \Sigma^+ \mid \exists \sigma' \in \Sigma^*, \sigma'' \in \Sigma^{*\infty} : \sigma'\sigma\sigma'' \in T\}$$

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

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

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

CTL Abstraction

Until Formulas (2)

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

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

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

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

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

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

$$\mathsf{sf}(T) \stackrel{\text{def}}{=} \{\sigma \in \Sigma^{+\infty} \mid \exists \sigma' \in \Sigma^* : \sigma'\sigma \in T\}$$

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

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 (Soundness and Completeness)

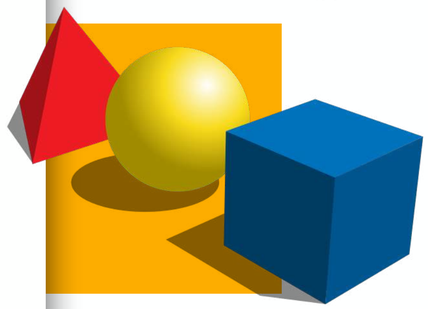
A program satisfies a **CTL property** ϕ for traces starting from a given set of initial states $\mathscr{I}$ if and only if $\mathscr{I} \subseteq \text{dom}(\mathscr{R}_\phi)$

Abstract CTL Program Semantics

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

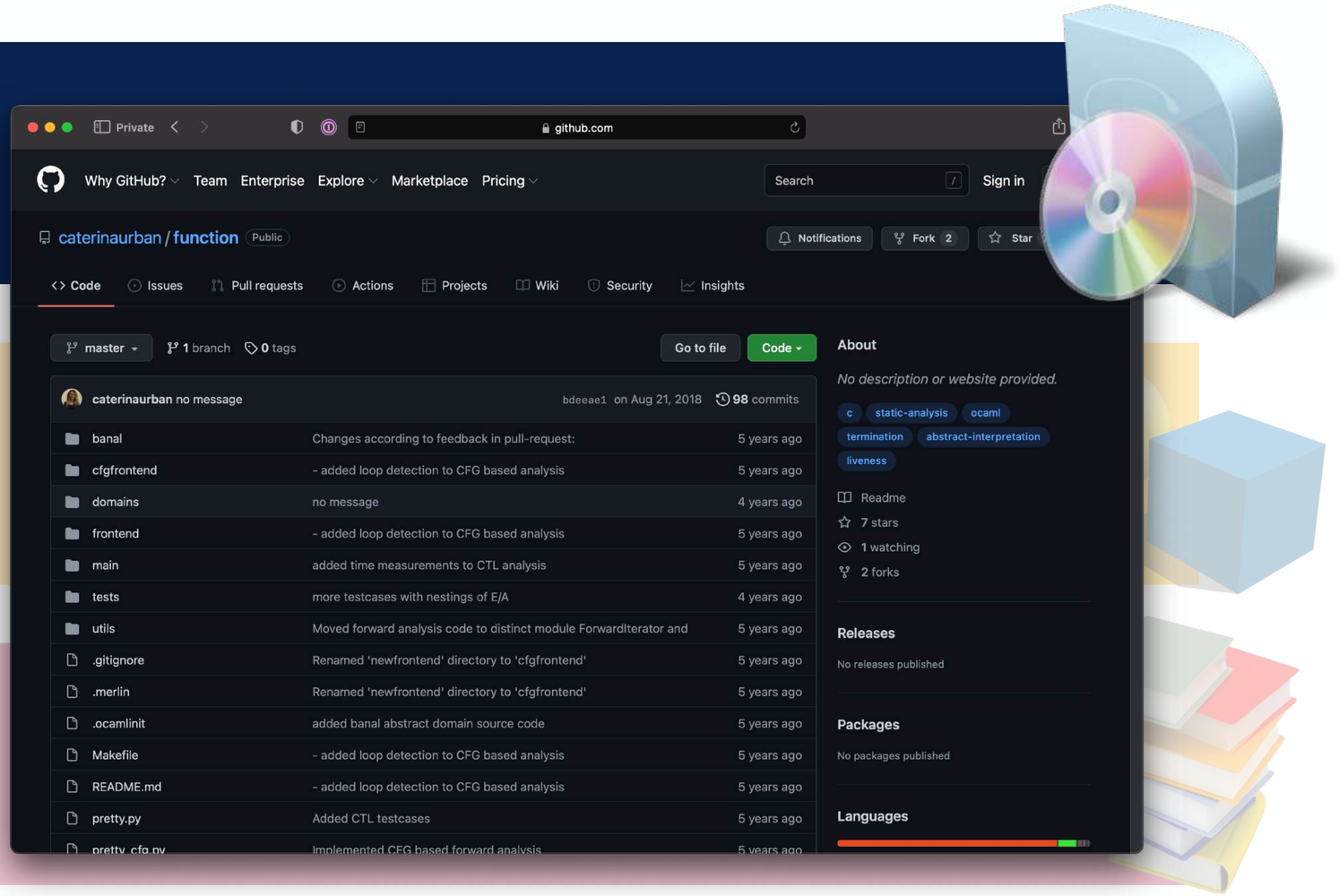
Computation tree logic (CTL) [6] is a temporal logic introduced by Clarke and Emerson to overcome certain limitations of linear temporal logic (LTL) [33] for program specification purposes. Most of the existing approaches for proving program properties expressed in CTL have limitations that restrict 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

Implementation

practical tools
targeting specific programs

algorithmic approaches
to decide program properties

mathematical models
of the program behavior



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.

[Urban18] **Caterina Urban, Samuel Ueltschi, Peter Müller.** Abstract Interpretation of CTL Properties. In SAS, 2018.