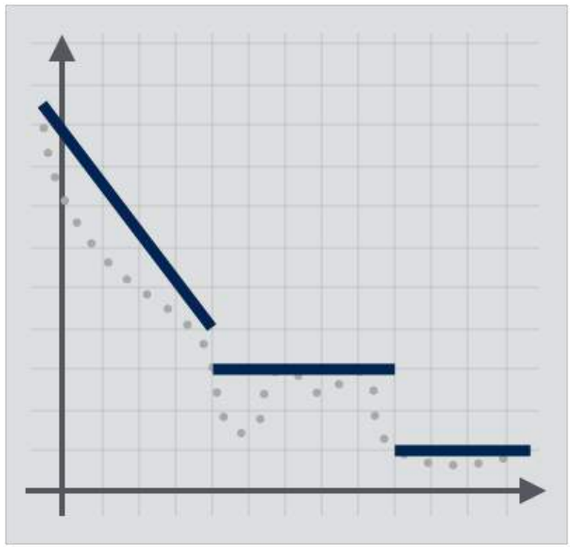
Abstract Interpretation for Liveness Properties

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



Caterina Urban

30 October 2020

Year 2020-2021

So far, we have focused on **using static analysis to avoid software failures**

The cost of software failure

- Patriot MIM-104 failure, 25 February 1991
  (death of 28 soldiers[1])
- Ariane 5 failure, 4 June 1996
  (cost estimated at more than 370 000 000 US$[2])
- Toyota electronic throttle control system failure, 2005
  (at least 89 death[3])
- Heartbleed bug in OpenSSL, April 2014
- the economic cost of software bugs is tremendous[4]
- ...

[1] R. Skeel. "Roundoff Error and the Patriot Missile". SIAM News, volume 25, nr 4.
[2] M. Dowson. "The Ariane 5 Software Failure". Software Engineering Notes 22 (2): 84, March 1997.
[3] CBSNews. Toyota "Unintended Acceleration" Has Killed 89. 20 March 2014.
[4] NIST. Software errors cost U.S. economy $59.5 billion annually. Tech. report, NIST Planning Report, 2002.

How can we avoid such failures?

- Choose a safe programming language.
  C (low level) / Ada, Java, OCaml (high level)
  yet, Ariane 5 software is written in Ada
- Carefully design the software.
  many software development methods exist
  yet, critical embedded software follow strict development processes
- Test the software extensively.
  yet, the erroneous code was well tested... on Ariane 4

$\Longrightarrow$ **not sufficient!**

We should use **formal methods.**

provide rigorous, mathematical insurance of correctness
may not prove everything, but give a precise notion of what is proved

This case triggered the first large scale static code analysis
(PolySpace Verifier, using abstract interpretation)

that is, for **proving** **Safety Properties**

Safety vs Liveness Properties

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

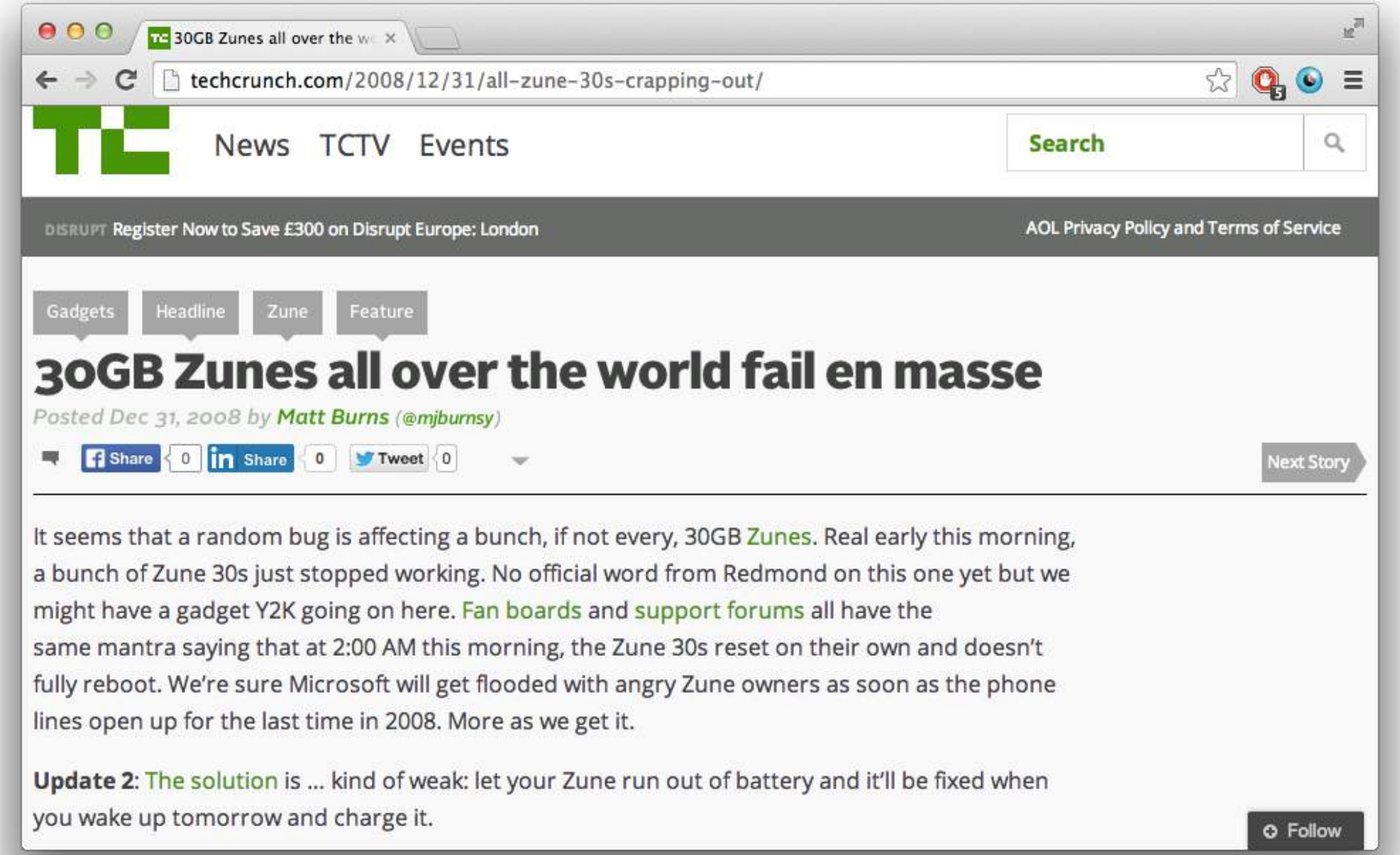

Amir Pnueli

Program Termination

The Zune Bug

31 December 2008

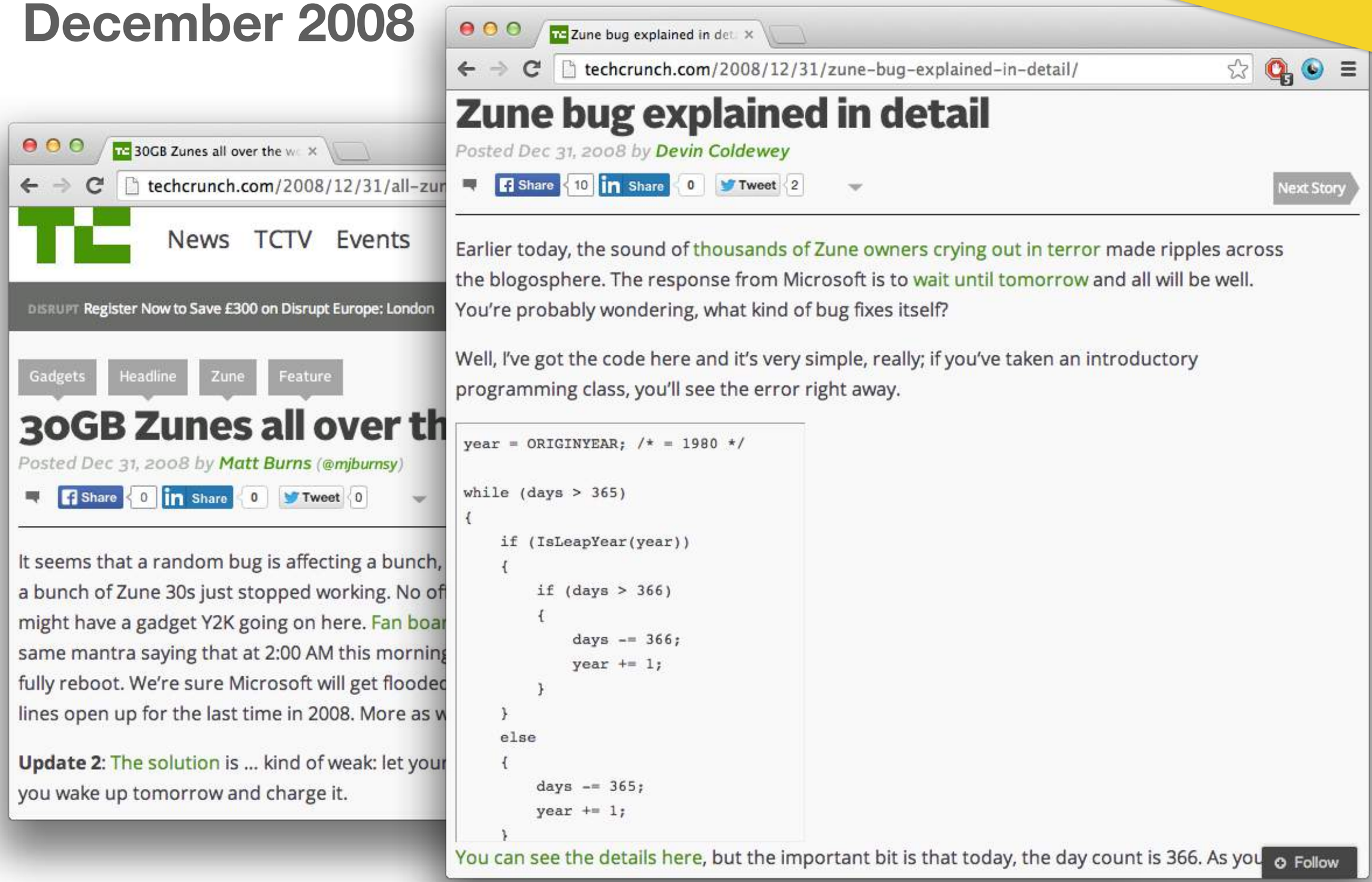
unresponsive systems



The Zune Bug

31 December 2008

unresponsive systems



Apache HTTP Server

Versions <2.3.3

denial-of-service attacks

cve.mitre.org

CVE List CNAs Board About News & Blog

NVD
Go to for:
CVSS Scores
CPE Info
Advanced Search

Common Vulnerabilities and Exposures

Search CVE List	Download CVE	Data Feeds	Request CVE IDs	Update a CVE Entry

TOTAL CVE Entries: 97475

HOME > CVE > CVE-2009-1890

Printer-Friendly View

CVE-ID	
CVE-2009-1890	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

The stream_reqbody_cl function in mod_proxy_http.c in the mod_proxy module in the Apache HTTP Server before 2.3.3, when a reverse proxy is configured, does not properly handle an amount of streamed data that exceeds the Content-Length value, which allows remote attackers to cause a denial of service (CPU consumption) via crafted requests.

References

Azure Storage Service

19 November 2014

Service interruptions

Update on Azure Storage Serv

Secure | https://azure.microsoft.com/en-us/blog/update-on-azure-st...

Microsoft Azure

Blog > Announcements

Update on Azure Storage Service Interruption

Posted on November 19, 2014

Jason Zander
Corporate Vice President, Microsoft Azure Team

Update: 11/22/2014, 12:41 PM PST Since Wednesday, we have been working to help a subset of customers take final steps to fully recover from Tuesday's storage service interruption. The incident has now been resolved and we are seeing normal activity in the system. You can find updates on the status dashboard: https://azure.microsoft.com/en-us/status. If you feel you are still having issues due to the incident, please contact azcommsm@microsoft.com, and we will be happy to assist, whether you have a support contract or not. Thank you all again for your feedback regarding communications around this incident. We are actively working to incorporate that feedback into our planning going forward. **Wednesday, November, 19, 2014** As part of a performance update to Azure Storage, an issue was discovered that resulted in reduced capacity across services utilizing Azure Storage, including Virtual Machines, Visual Studio Online, Websites, Search and other Microsoft services. Prior to applying the performance update, it had been tested over several weeks in a subset of our customer-facing storage service for Azure Tables. We typically call this "flighting," as we work to identify issues before we broadly deploy any updates. The flighting test demonstrated a notable performance improvement and we proceeded to deploy the update across the storage service. During the rollout we discovered an issue that resulted in storage blob front ends going into an infinite loop, which had gone undetected during flighting. The net result was an inability for the front ends to take on further traffic, which

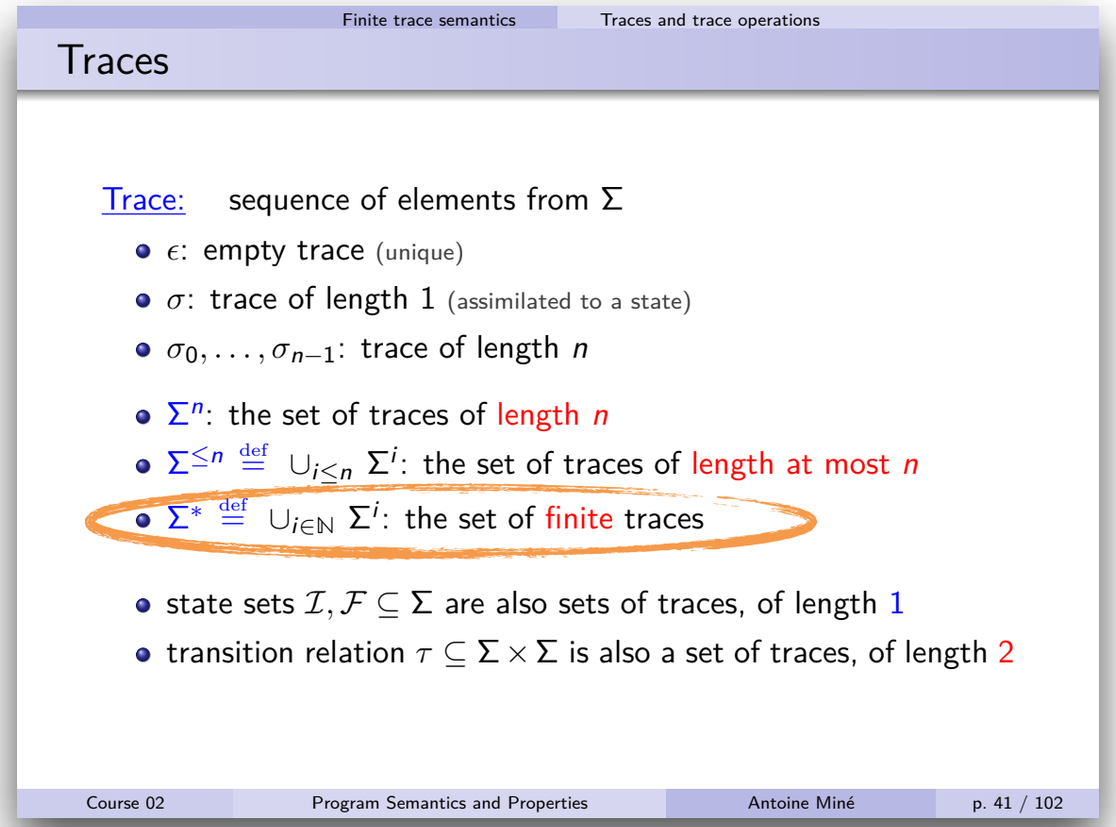
Potential and Definite Termination

Definition

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

Definition

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

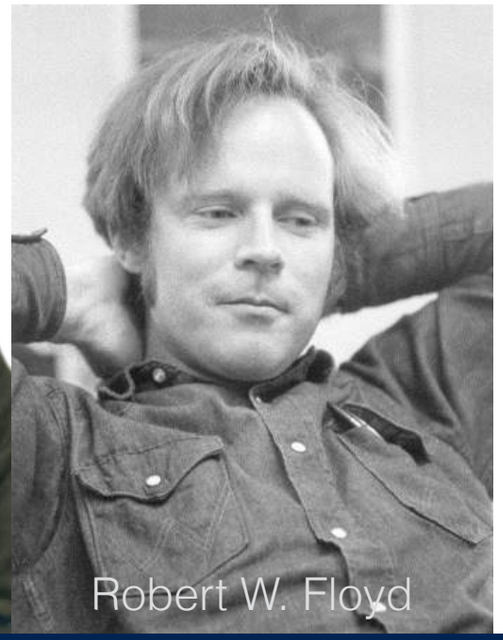


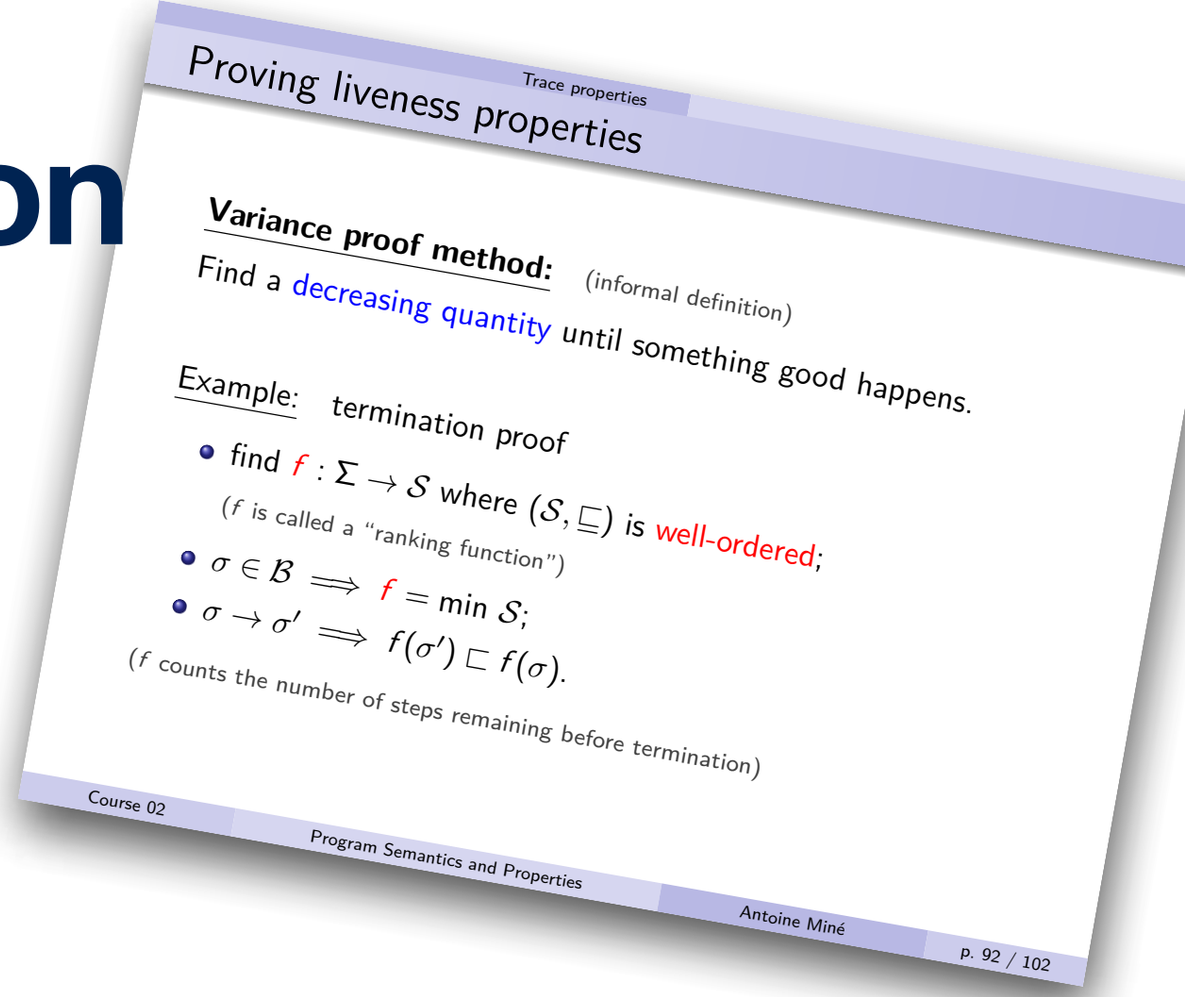
In *absence of non-determinism*, potential and definite termination coincide

Definite Termination

Ranking Functions


Alan Turing


Robert W. Floyd



Definition

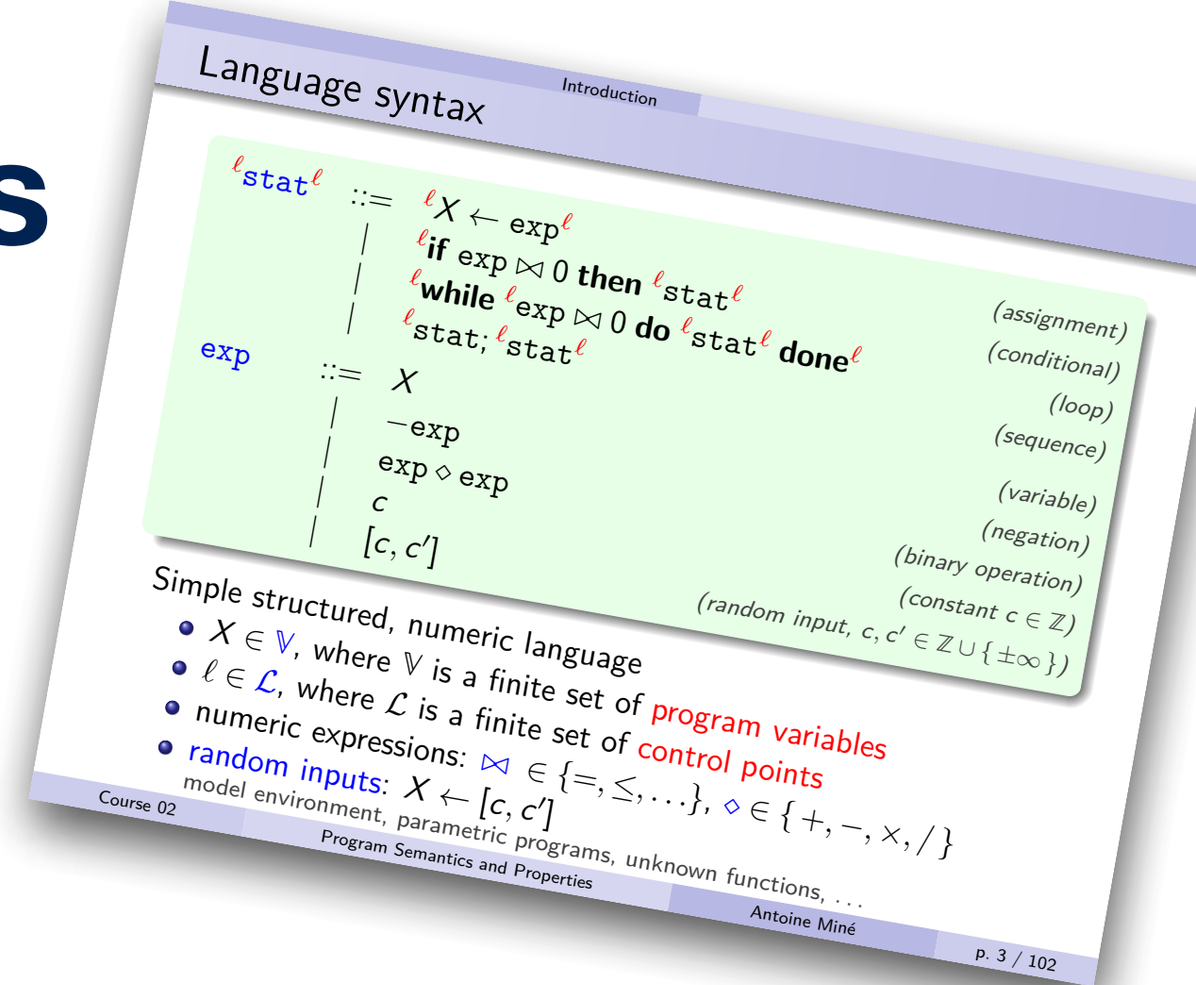
Given a transition system $\langle \Sigma, \tau \rangle$, a **ranking function** is a partial function $f : \Sigma \rightharpoonup \mathscr{W}$ from the set of program states Σ into a well-ordered set $\langle \mathscr{W}, \leq \rangle$ whose value *strictly decreases* through transitions between states, that is, $\forall \sigma, \sigma' \in \mathrm{dom}(f) : (\sigma, \sigma') \in \tau \Rightarrow f(\sigma') < f(\sigma)$

The best known *well-ordered sets* are **naturals** $\langle \mathbb{N}, \leq \rangle$ and **ordinals** $\langle \mathbb{O}, \leq \rangle$

Ranking Functions

Example

$^{1}x \leftarrow [-\infty, +\infty]$
while $^{2}(1 - x < 0)$ **do**
$^{3}x \leftarrow x - 1$
od^{4}



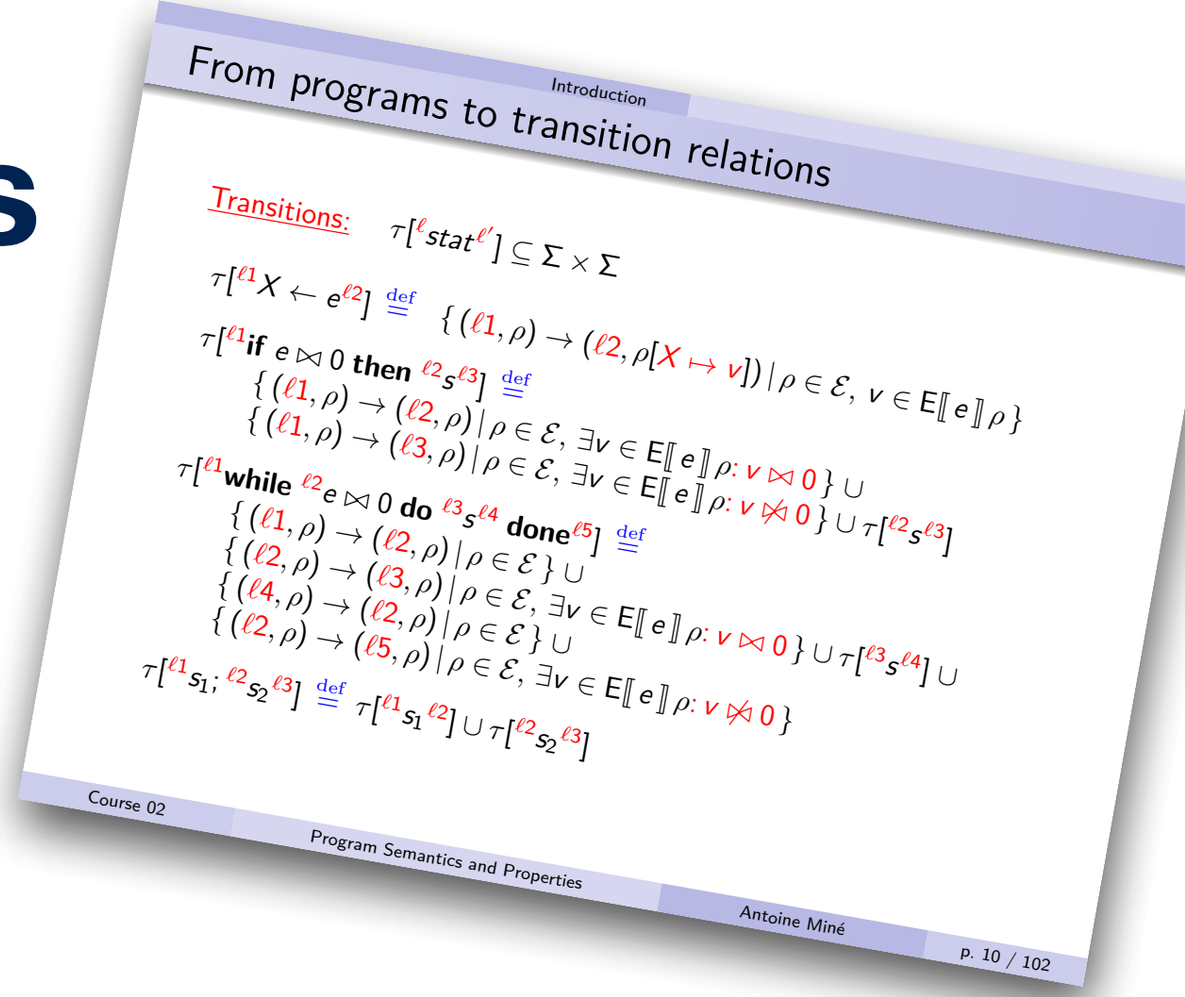
Ranking Functions

Example (continue)

[1]x ← [-∞, +∞]
while [2](1 - x < 0) **do**
[3]x ← x - 1
od[4]

$$\Sigma \stackrel{\text{def}}{=} \{\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4}\} \times \mathscr{E}$$

$$\begin{aligned}
\tau \stackrel{\text{def}}{=}\ & \{(\mathbf{1},\rho) \to (\mathbf{2},\rho[X \mapsto v]) \mid \rho \in \mathscr{E}, v \in \mathbb{Z}\} \\
\cup\ & \{(\mathbf{2},\rho) \to (\mathbf{3},\rho) \mid \ \mid \rho \in \mathscr{E}, \exists v \in E[\![1 - x]\!]\rho : v < 0\} \\
\cup\ & \{(\mathbf{3},\rho) \to (\mathbf{2},\rho[X \mapsto v]) \mid \rho \in \mathscr{E}, v \in E[\![x - 1]\!]\rho\} \\
\cup\ & \{(\mathbf{2},\rho) \to (\mathbf{4},\rho) \mid \ \mid \rho \in \mathscr{E}, \exists v \in E[\![1 - x]\!]\rho : v \not< 0\}
\end{aligned}$$

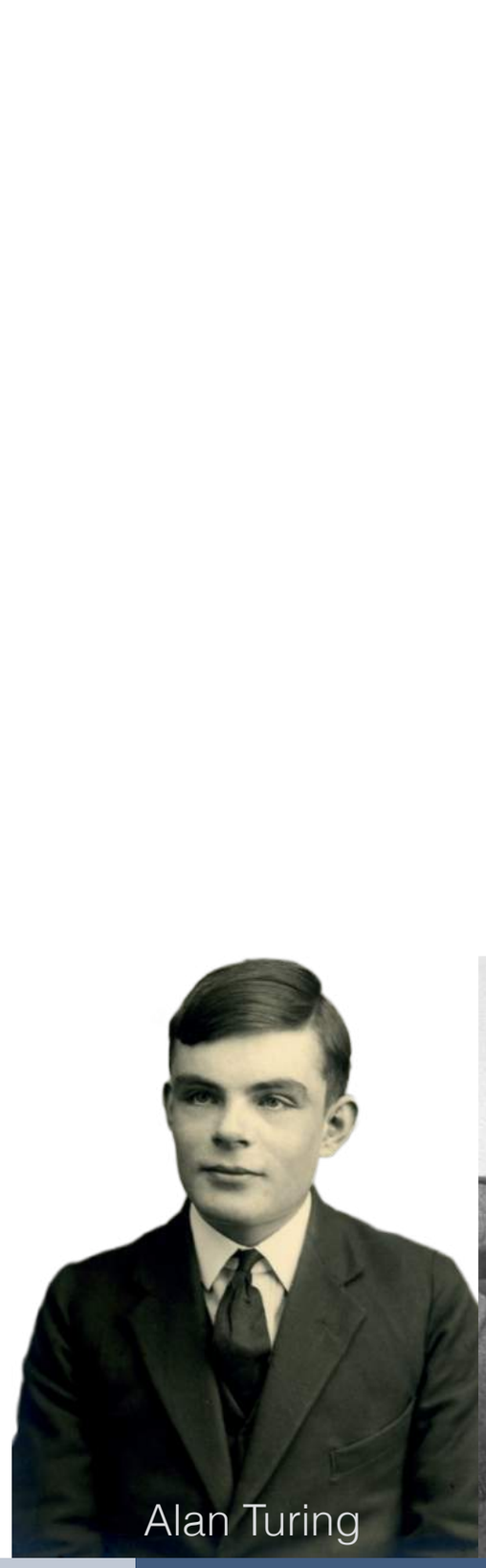


Ranking Functions

Example (continue)

[1]x ← [-∞, +∞]
 while [2](1 - x < 0) **do**
 [3]x ← x - 1
od[4]

Most obvious ranking function:
a mapping $f\colon \Sigma \rightharpoonup \mathbb{O}$
from each program state
to
(a well-chosen upper bound on)
the number of steps until termination


Alan Turing


Robert W. Floyd

Ranking Functions

Example (continue)

```
[1]x ← [-∞, +∞]
 while [2](1 - x < 0) do
       [3]x ← x - 1
od[4]
```

We define the ranking function $f\colon \Sigma \rightharpoonup \mathbb{O}$ by partitioning with respect to the program control points, i.e., $f\colon \mathscr{L} \to (\mathscr{E} \rightharpoonup \mathbb{O})$

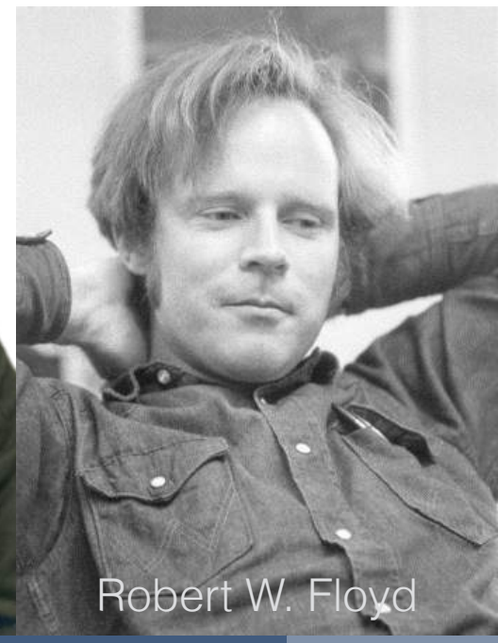
$$f(\mathbf{4}) \stackrel{\text{def}}{=} \lambda\rho.0$$

$$f(\mathbf{2}) \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 1 & 1-\rho(x) \not< 0 \\ 2\rho(x)-1 & 1-\rho(x) < 0 \end{cases}$$

$$f(\mathbf{3}) \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 2 & 2-\rho(x) \not< 0 \\ 2\rho(x)-2 & 2-\rho(x) < 0 \end{cases}$$

$$f(\mathbf{1}) \stackrel{\text{def}}{=} \lambda\rho\,.\,\omega$$


Alan Turing


Robert W. Floyd

Potential Termination

Potential Ranking Functions

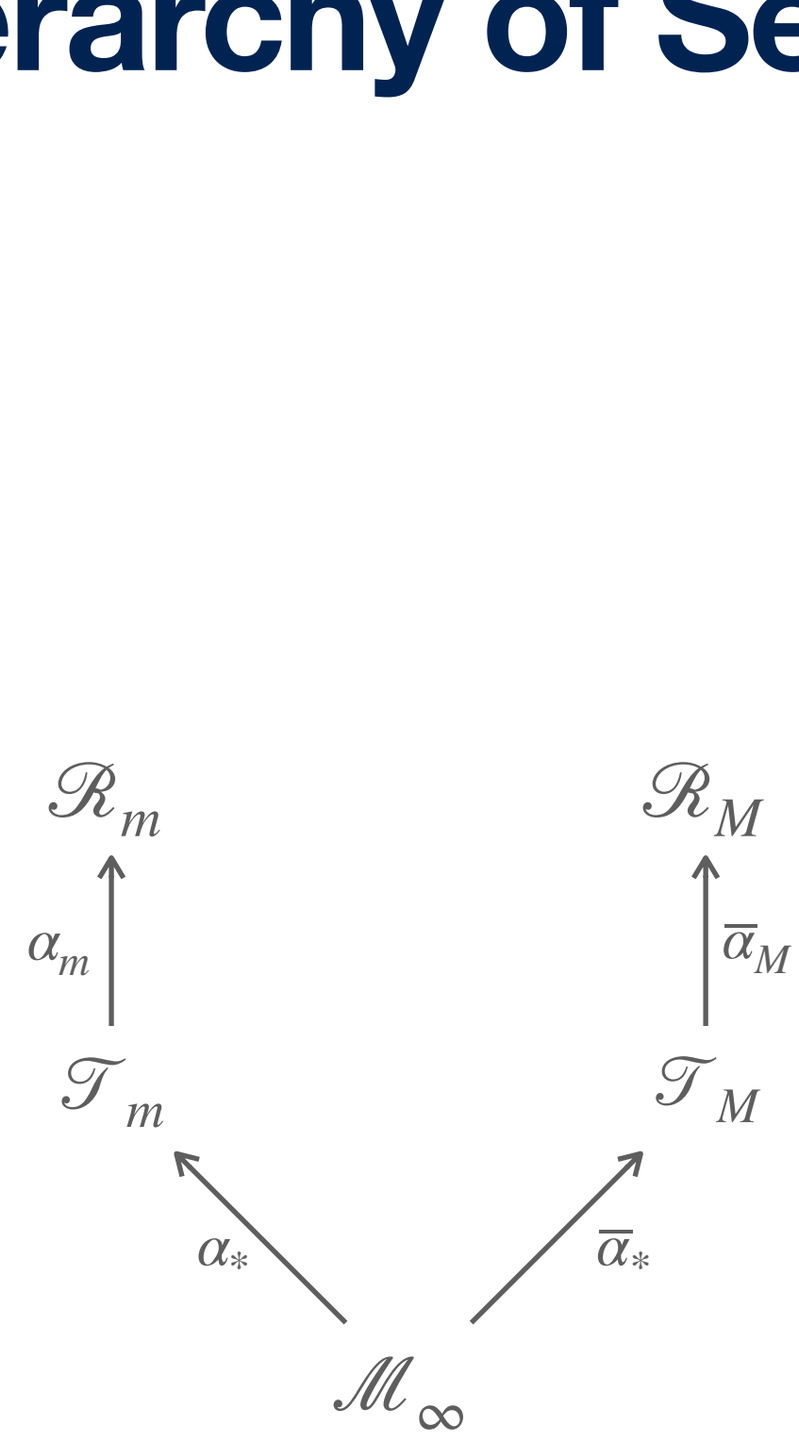
For proving potential termination, we use a *weaker* notion of ranking function, which *decreases along at least one transition* during program execution

Definition

Given a transition system $\langle \Sigma, \tau \rangle$, a **potential ranking function** is a partial function $f \colon \Sigma \rightharpoonup \mathscr{W}$ from the set of states Σ into a well-ordered set $\langle \mathscr{W}, \leq \rangle$ whose value *strictly decreases* through at least one transitions from each state, that is, $\forall \sigma \in \mathrm{dom}(f) \colon (\exists \bar{\sigma} \in \mathrm{dom}(f) \colon (\sigma, \bar{\sigma}) \in \tau) \Rightarrow \exists \sigma' \in \mathrm{dom}(f) \colon (\sigma, \sigma') \in \tau \wedge f(\sigma') < f(\sigma)$

Termination Semantics

Hierarchy of Semantics

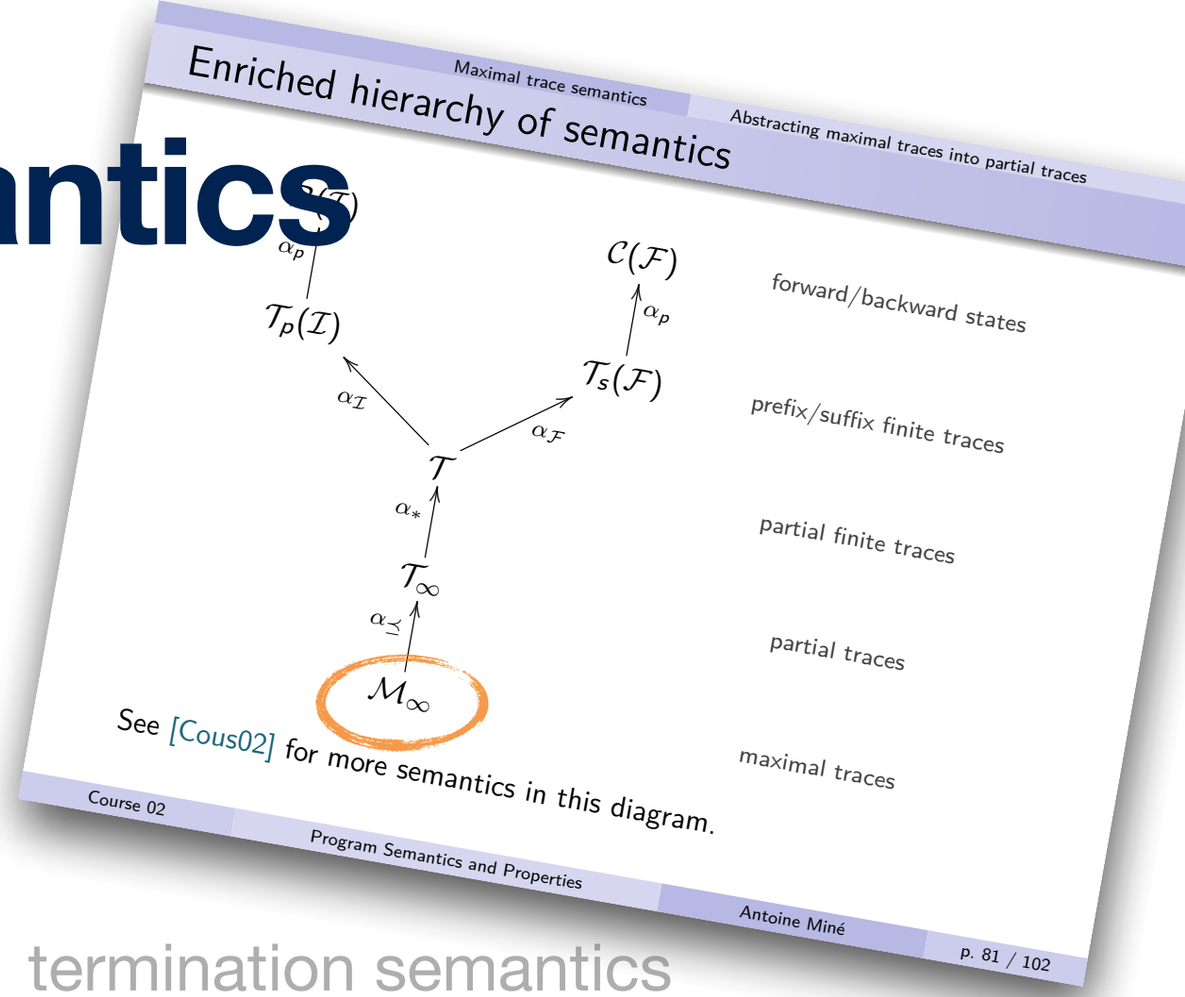


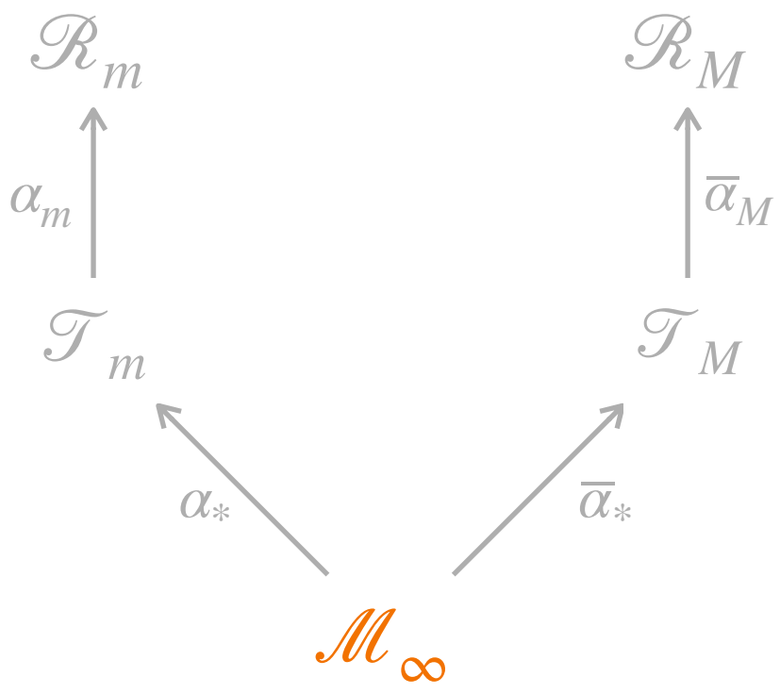
termination semantics

termination trace semantics

maximal trace semantics

Hierarchy of Semantics





termination semantics

termination trace semantics

maximal trace semantics

Maximal Trace Semantics

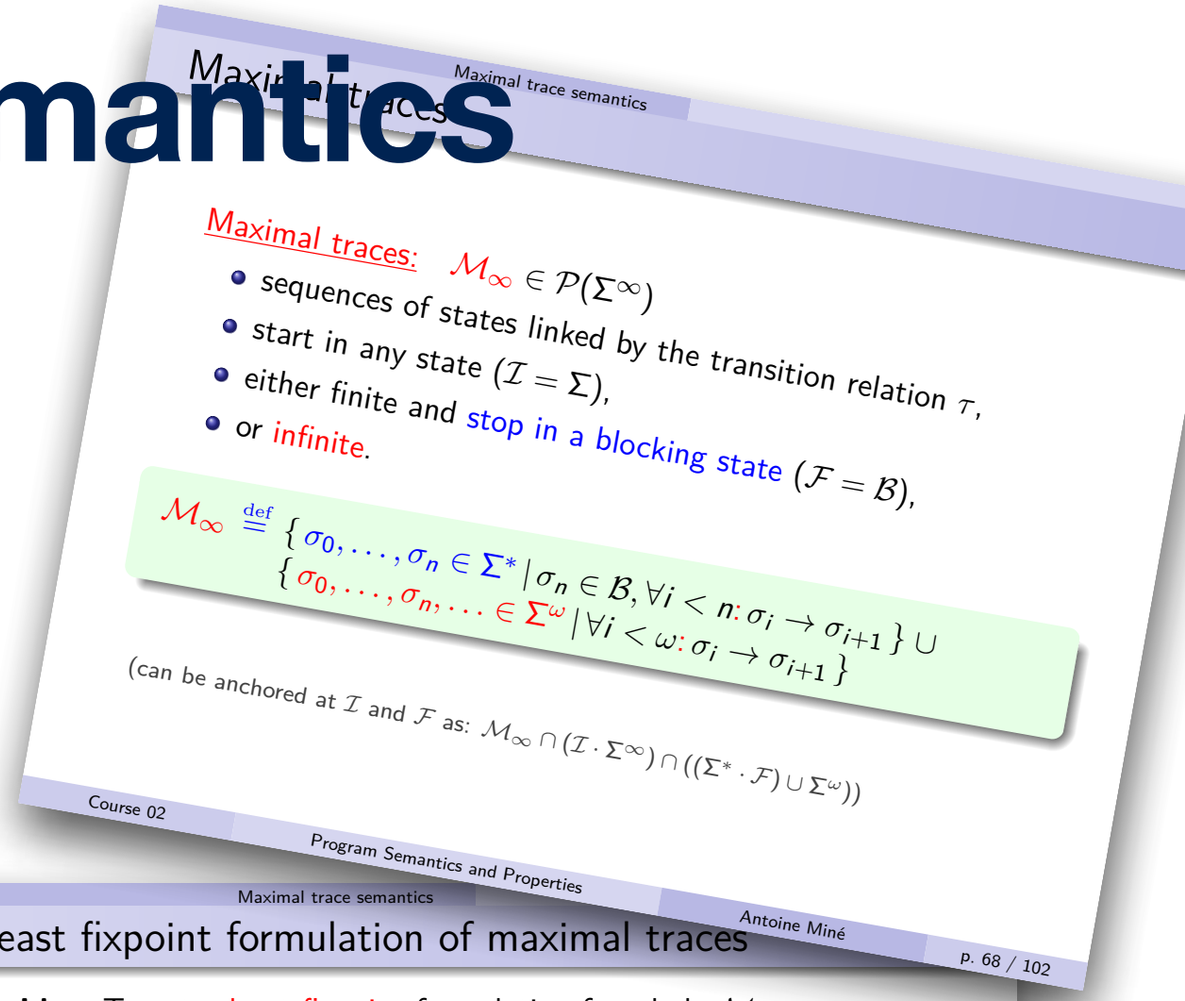
Example

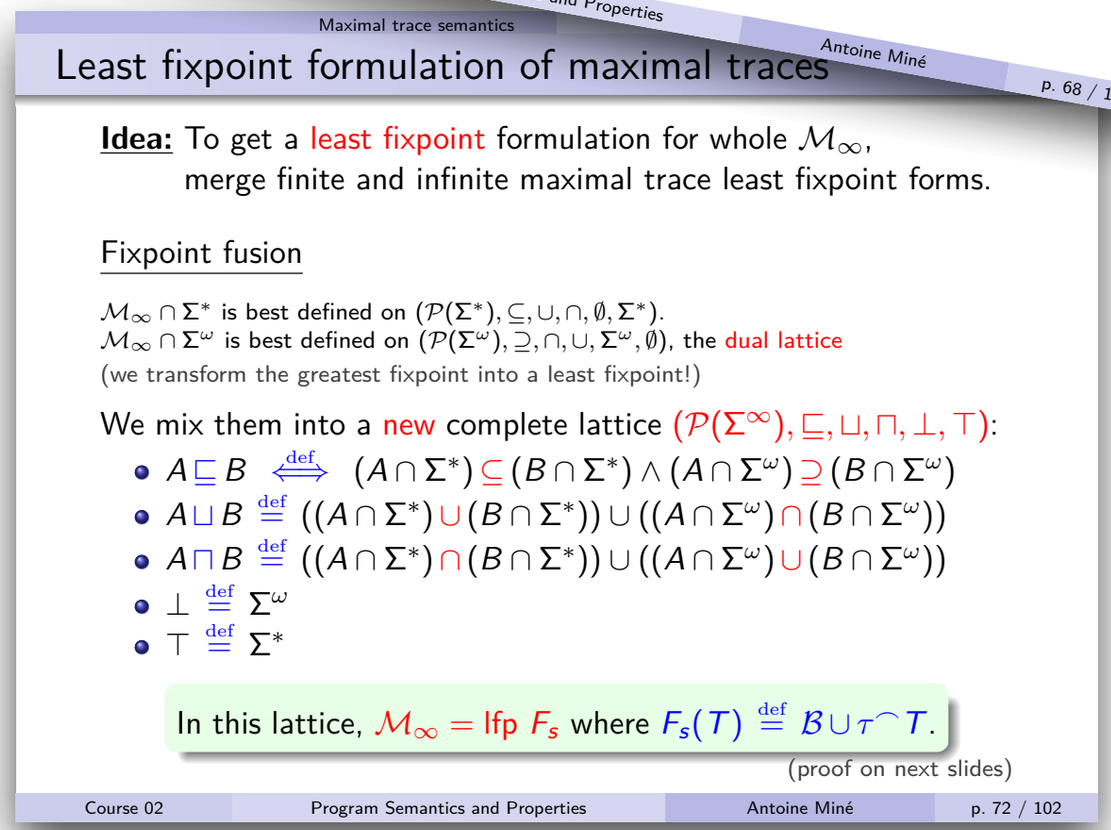
while [1]([-∞, +∞] ≠ 0) **do**
  [2]**skip**
od[3]

$$\Sigma \stackrel{\text{def}}{=} \{1,2,3\} \times \mathscr{E}$$

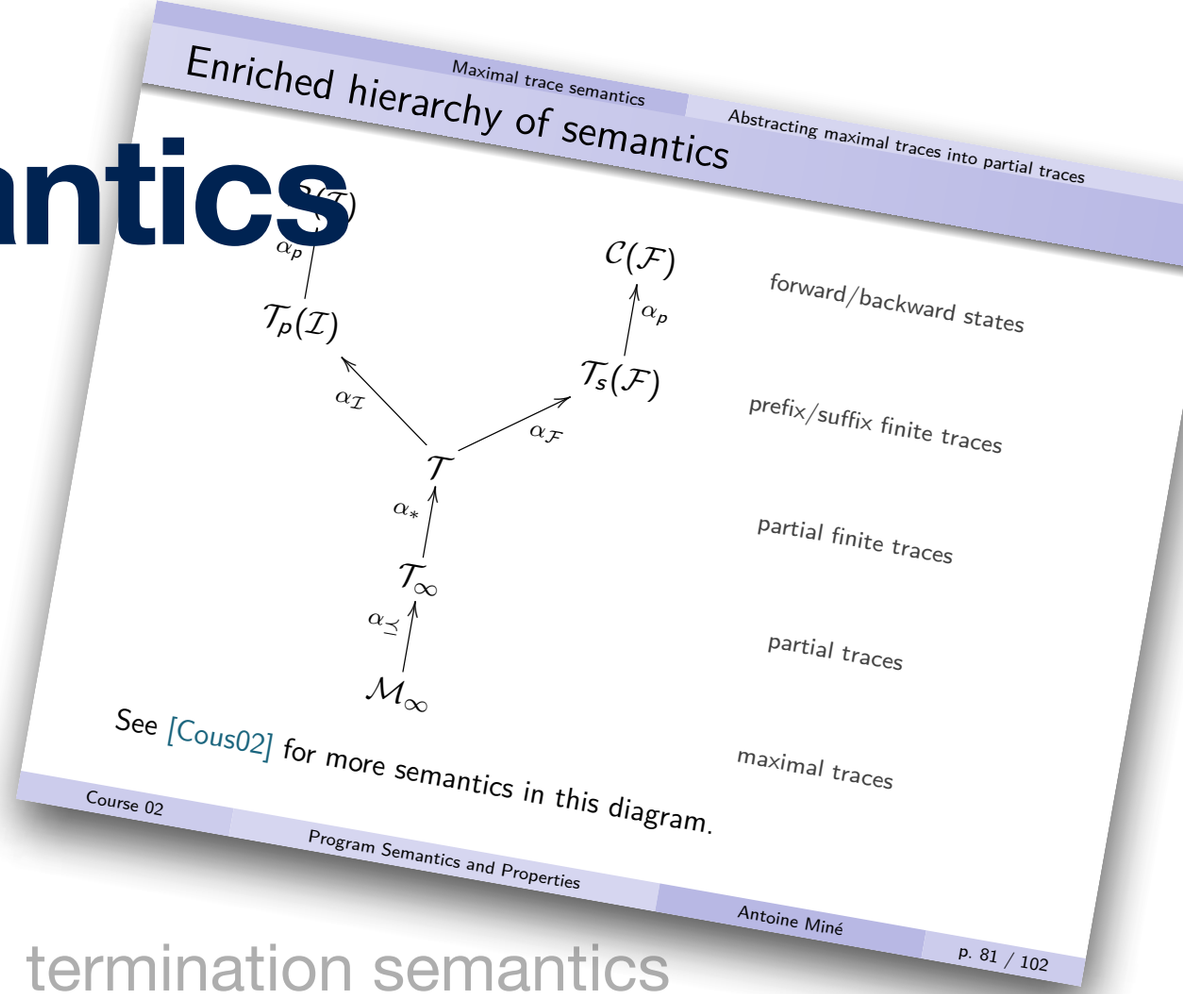
$$\begin{aligned}\tau \stackrel{\text{def}}{=} & \{(1,\rho) \to (2,\rho) \mid \rho \in \mathscr{E}\} \\ & \cup \{(2,\rho) \to (1,\rho) \mid \rho \in \mathscr{E}\} \\ & \cup \{(1,\rho) \to (3,\rho) \mid \rho \in \mathscr{E}\}\end{aligned}$$

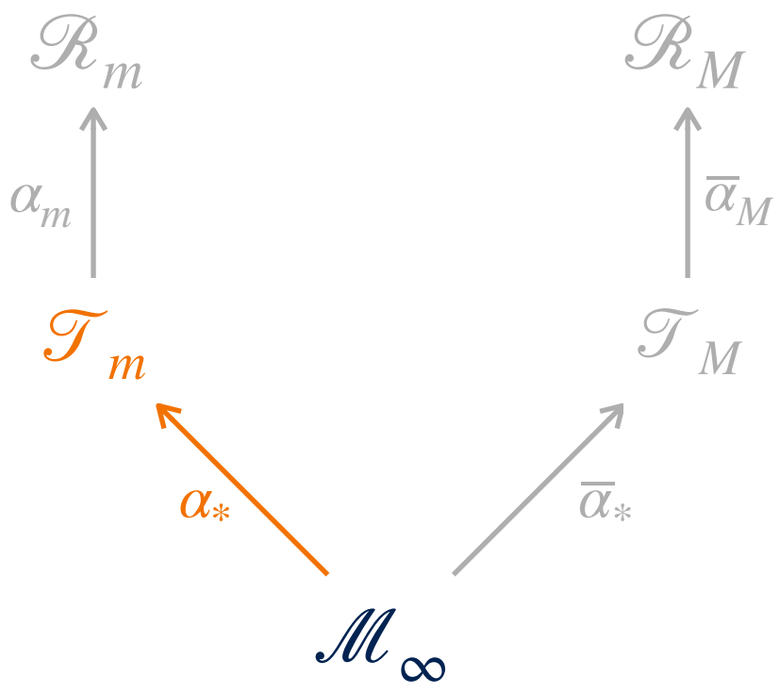
$$\begin{aligned}\mathscr{M}_\infty \stackrel{\text{def}}{=} & \{(1,\rho)(2,\rho)^*(3,\rho) \mid \rho \in \mathscr{E}\} \\ & \cup \{(1,\rho)(2,\rho)^\omega \mid \rho \in \mathscr{E}\}\end{aligned}$$





Hierarchy of Semantics





termination semantics

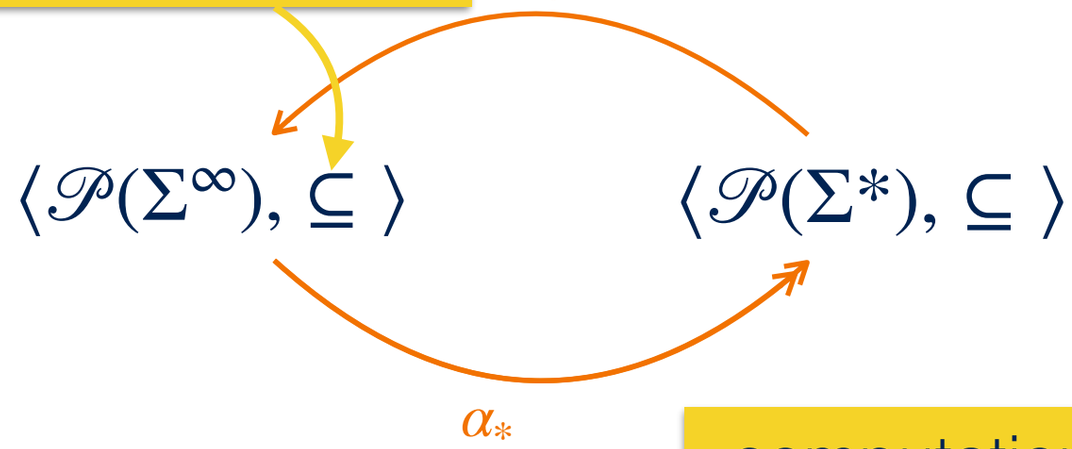
termination trace semantics

maximal trace semantics

Potential Termination Trace Semantics

Potential Termination Abstraction

approximation order

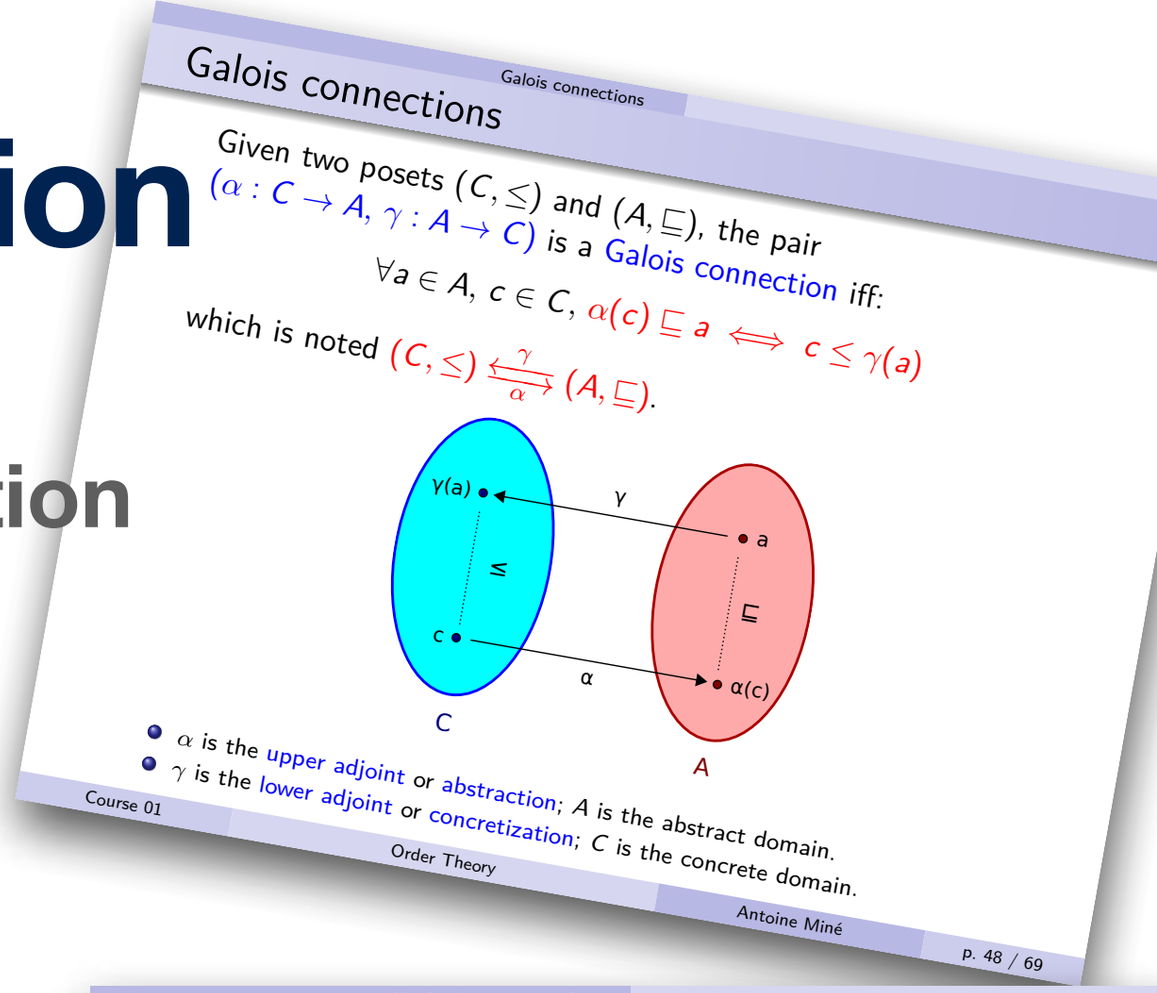


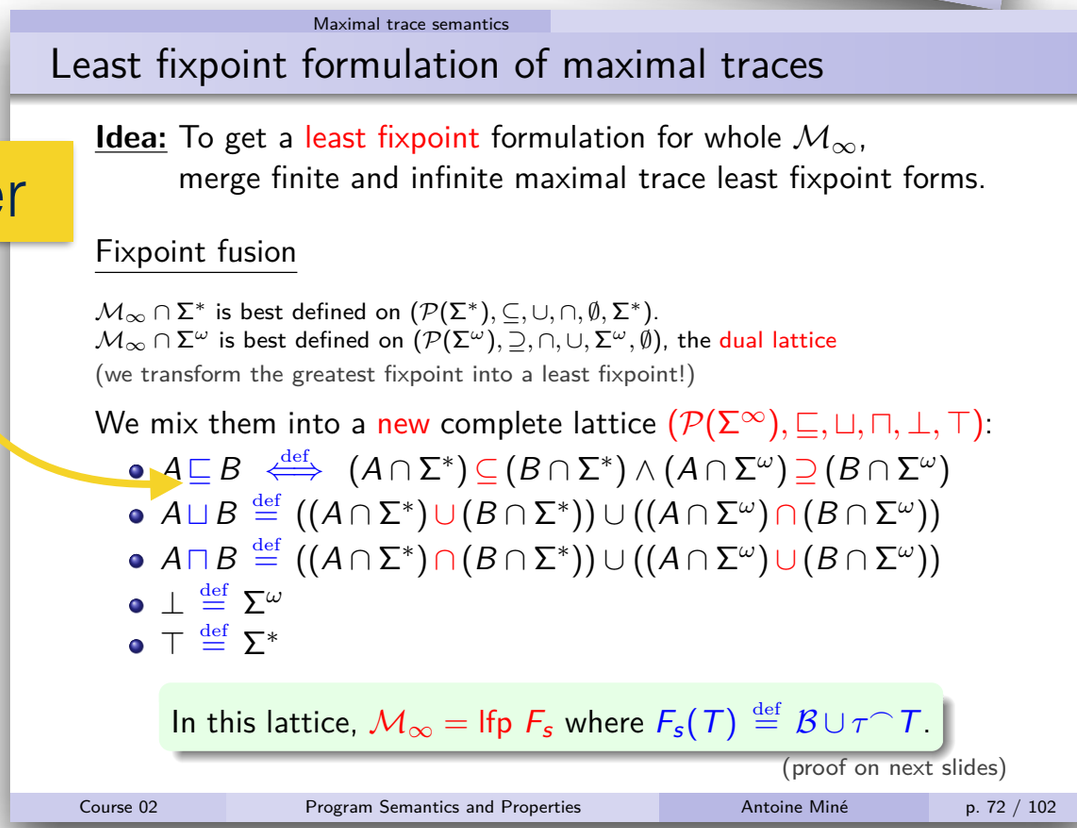
computational order

$$\alpha_*(T) \stackrel{\text{def}}{=} T \cap \Sigma^*$$

$$\gamma_*(T) \stackrel{\text{def}}{=} T \cup \Sigma^\omega$$

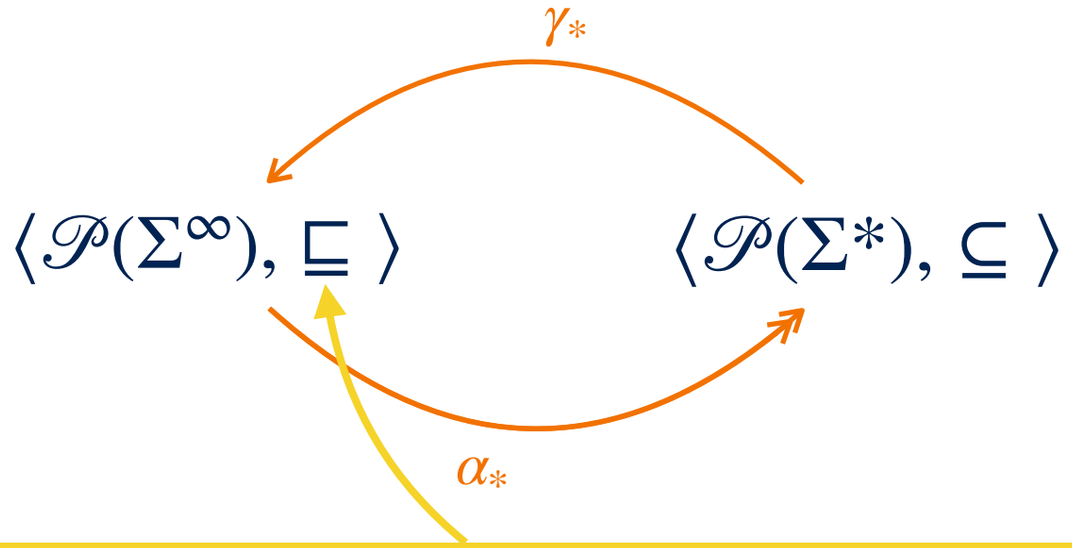
Example:
$\alpha_*(\{ab, aba, bb, ba^\omega\}) = \{ab, aba, bb\}$





Potential Termination Trace Semantics

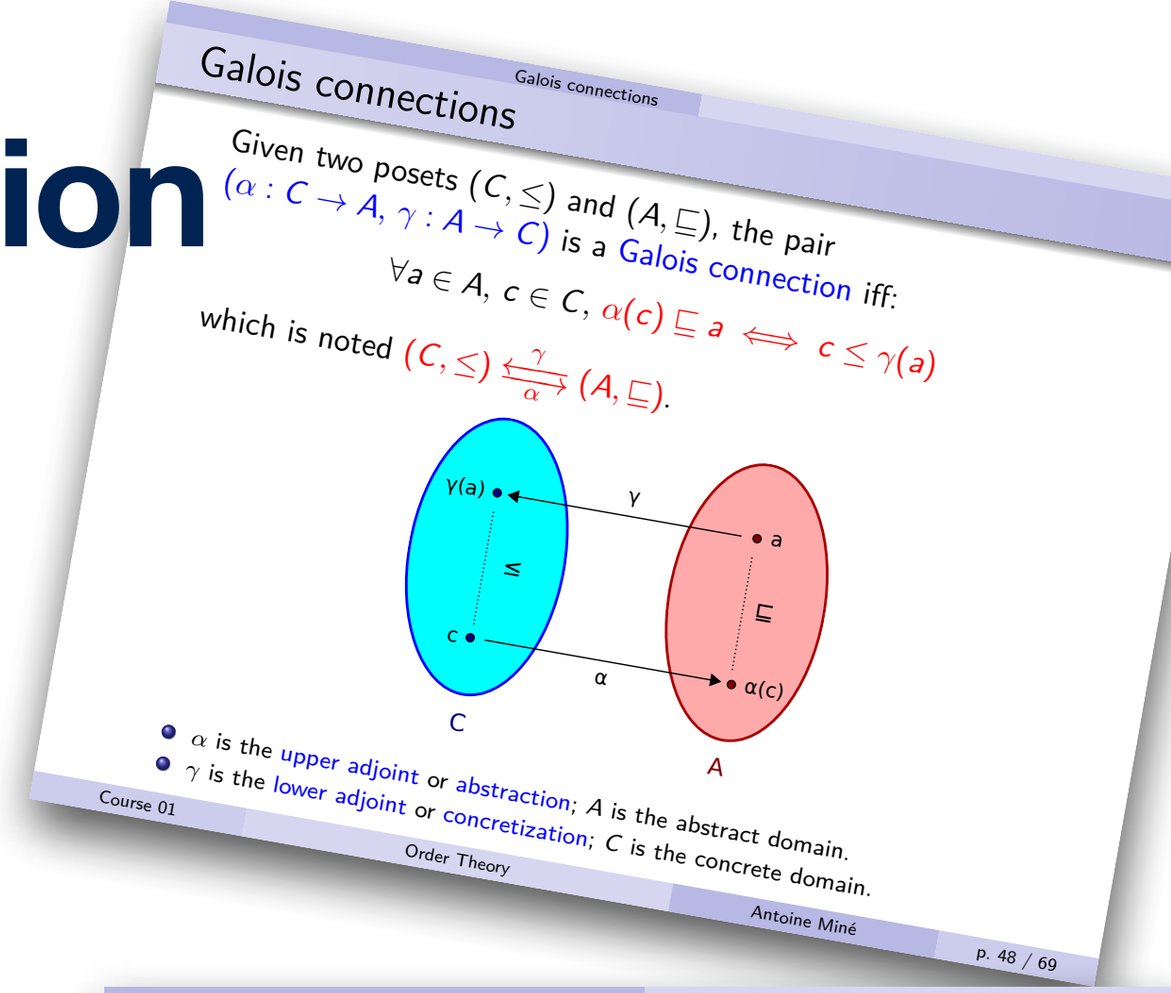
Finite Trace Abstraction

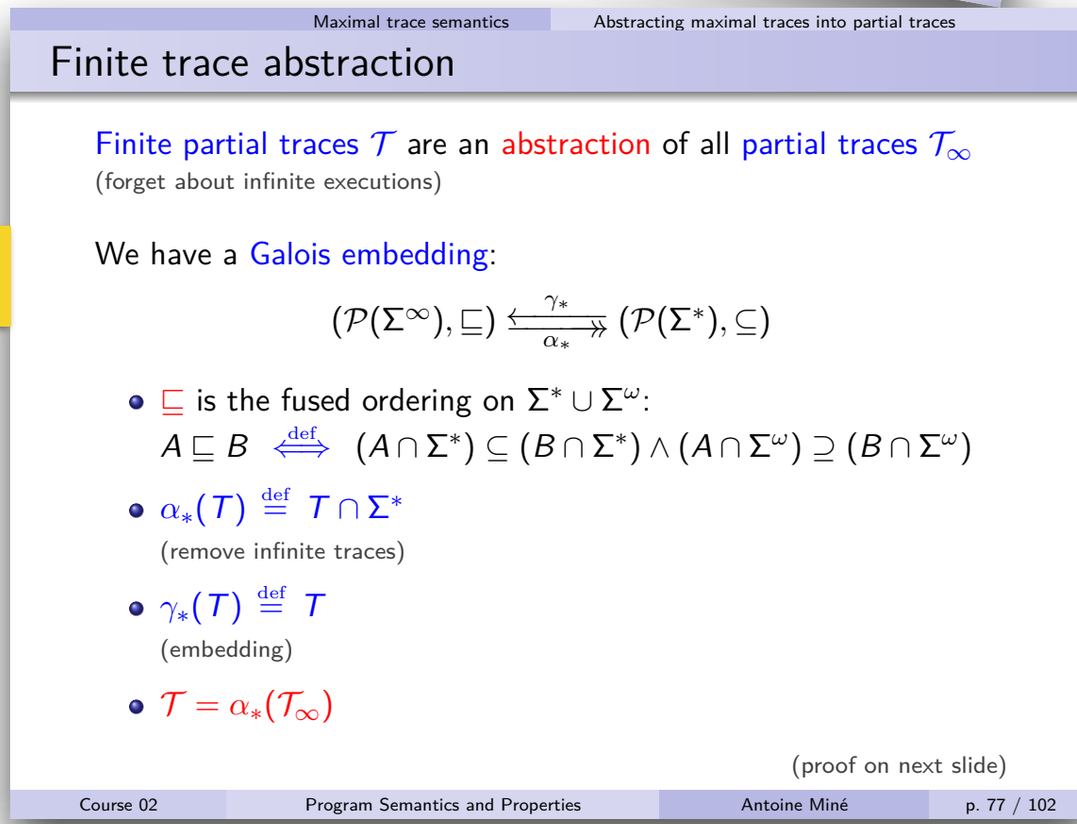


approximation and computational order coincide

$$\alpha_*(T) \stackrel{\text{def}}{=} T \cap \Sigma^*$$

$$\gamma_*(T) \stackrel{\text{def}}{=} T$$





Potential Termination Trace Semantics

Kleenian Fixpoint Transfer

- $\langle \mathscr{P}(\Sigma^{\infty}), \sqsubseteq \rangle$
- $\mathscr{M}_{\infty} \stackrel{\text{def}}{=} \text{lfp}^{\sqsubseteq} F_s$
  $F_s(T) \stackrel{\text{def}}{=} \mathscr{B} \cup \tau^{\frown} T$
- $\langle \mathscr{P}(\Sigma^{*}), \subseteq \rangle$
- $\alpha_{*} : \mathscr{P}(\Sigma^{\infty}) \to \mathscr{P}(\Sigma^{*})$
  $\alpha_{*}(T) \stackrel{\text{def}}{=} T \cap \Sigma^{*}$

$$\mathscr{T}_m \stackrel{\text{def}}{=} \alpha_{*}(\mathscr{M}_{\infty}) = \text{lfp}^{\subseteq} F_{*}$$

$$F_{*}(T) \stackrel{\text{def}}{=} \mathscr{B} \cup \tau^{\frown} T$$

Fixpoint transfer

If we have:

- a Galois connection $(C, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \sqsubseteq)$ between CPOs
- monotonic concrete and abstract functions $f : C \to C$, $f^{\sharp} : A \to A$
- a commutation condition $\alpha \circ f = f^{\sharp} \circ \alpha$
- an element a and its abstraction $a^{\sharp} = \alpha(a)$

then $\alpha(\text{lfp}_a\, f) = \text{lfp}_{a^{\sharp}}\, f^{\sharp}$.

Theorem

Let $\langle C, \leq \rangle$ and $\langle A, \sqsubseteq \rangle$ be complete partial orders, let $f : C \to C$ and $f^{\#} : A \to A$ be monotonic functions, and let $\alpha : C \to A$ be a *continous* abstraction function such that $\alpha(a) = a^{\#}$, for $a \in C$ and $a^{\#} \in A$, and that satisfies the commutation condition $\alpha \circ f = f^{\#} \circ \alpha$. Then, we have the fixpoint abstraction $\alpha(\text{lfp}^{\leq}_{a} f) = \text{lfp}^{\sqsubseteq}_{a^{\#}} f^{\#}$.

Potential Termination Trace Semantics

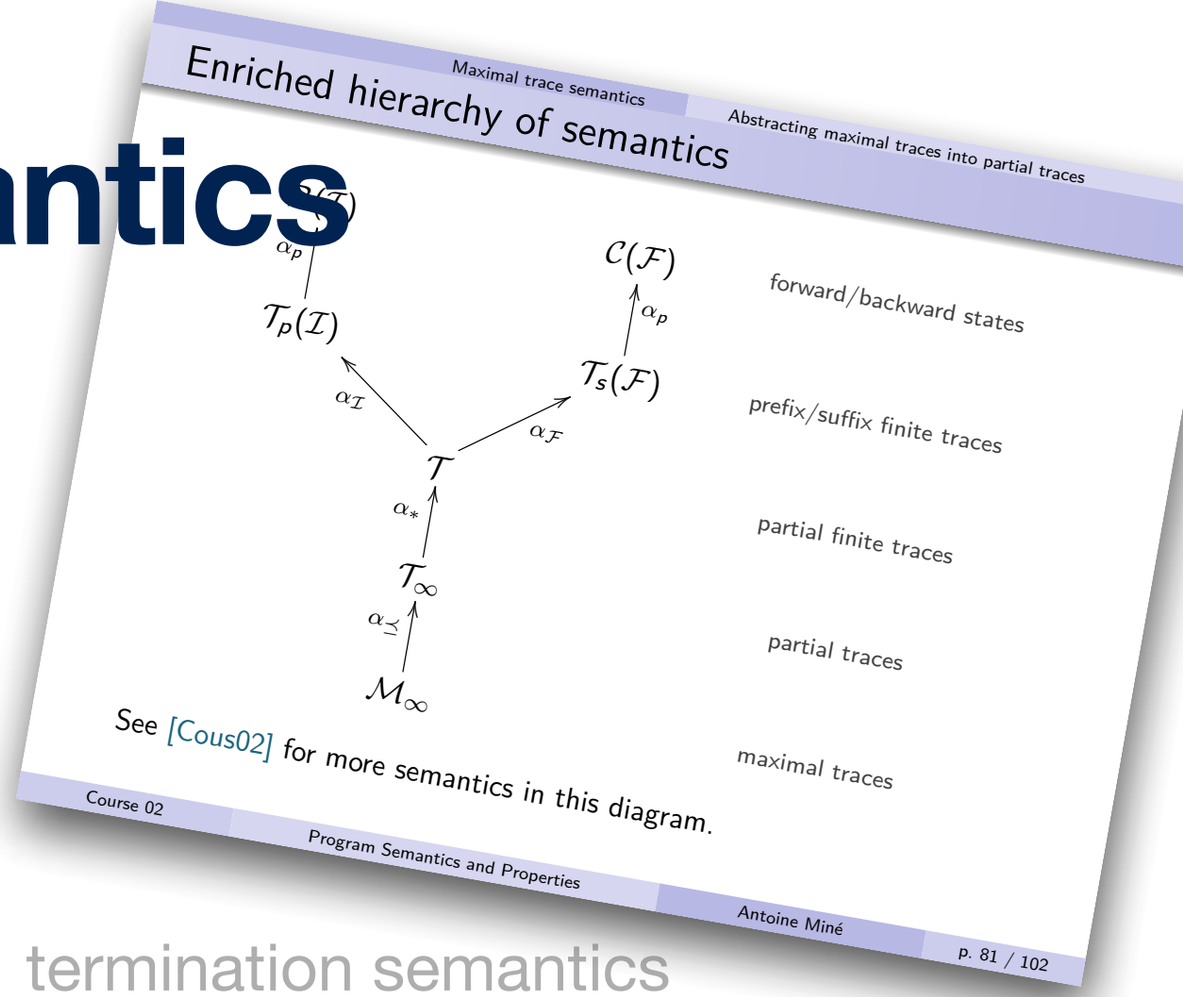
Example

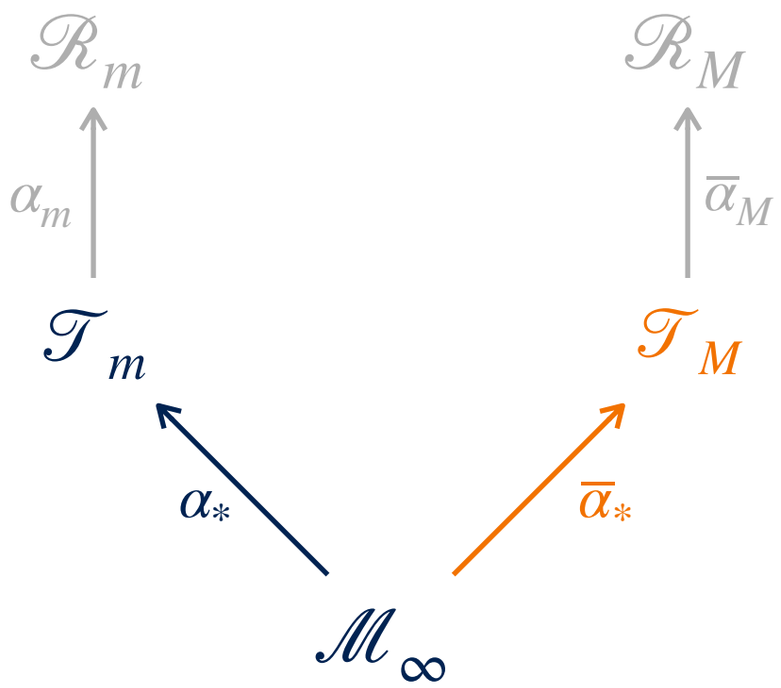
while [1]([-∞, +∞] ≠ 0) **do**
    [2]**skip**
od[3]

$$\mathscr{M}_\infty \stackrel{\text{def}}{=} \{(\mathbf{1},\rho)(\mathbf{2},\rho)^*(\mathbf{3},\rho) \mid \rho \in \mathscr{E}\} \cup \{(\mathbf{1},\rho)(\mathbf{2},\rho)^\omega \mid \rho \in \mathscr{E}\}$$

$$\mathscr{T}_m \stackrel{\text{def}}{=} \{(\mathbf{1},\rho)(\mathbf{2},\rho)^*(\mathbf{3},\rho) \mid \rho \in \mathscr{E}\}$$

Hierarchy of Semantics





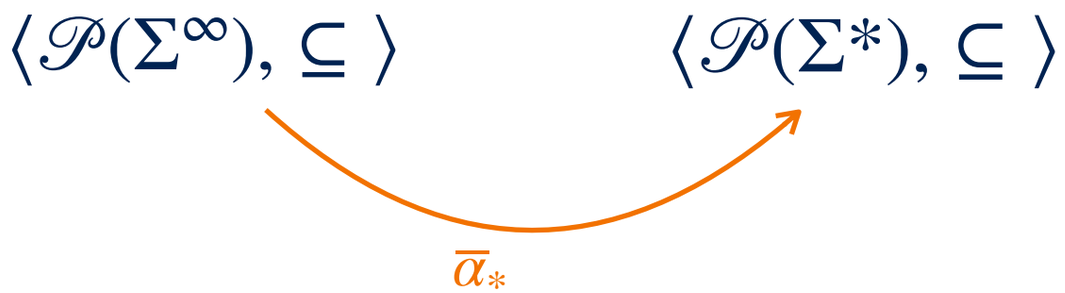
termination semantics

termination trace semantics

maximal trace semantics

Definite Termination Trace Semantics

Definite Termination Abstraction



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

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

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

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

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

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

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

Theorem

Let $\langle C, \leq , \vee , \wedge , \bot , \top \rangle$ and $\langle A, \sqsubseteq , \sqcup , \sqcap , \bot^{\#} , \top^{\#} \rangle$ be complete lattices, let $f : C \to C$ and $f^{\#} : A \to A$ be monotonic functions, and let $\alpha : C \to A$ be an abstraction function that is a complete $\wedge$-morphism $(\forall S \subseteq C : f(\wedge S) = \sqcap \{f(s) \mid s \in S\})$ 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 $\alpha(\mathsf{lfp}^{\leq} f) = \mathsf{lfp}^{\sqsubseteq} f^{\#}$.

(see proof in [Cousot02])

Definite Termination Trace Semantics

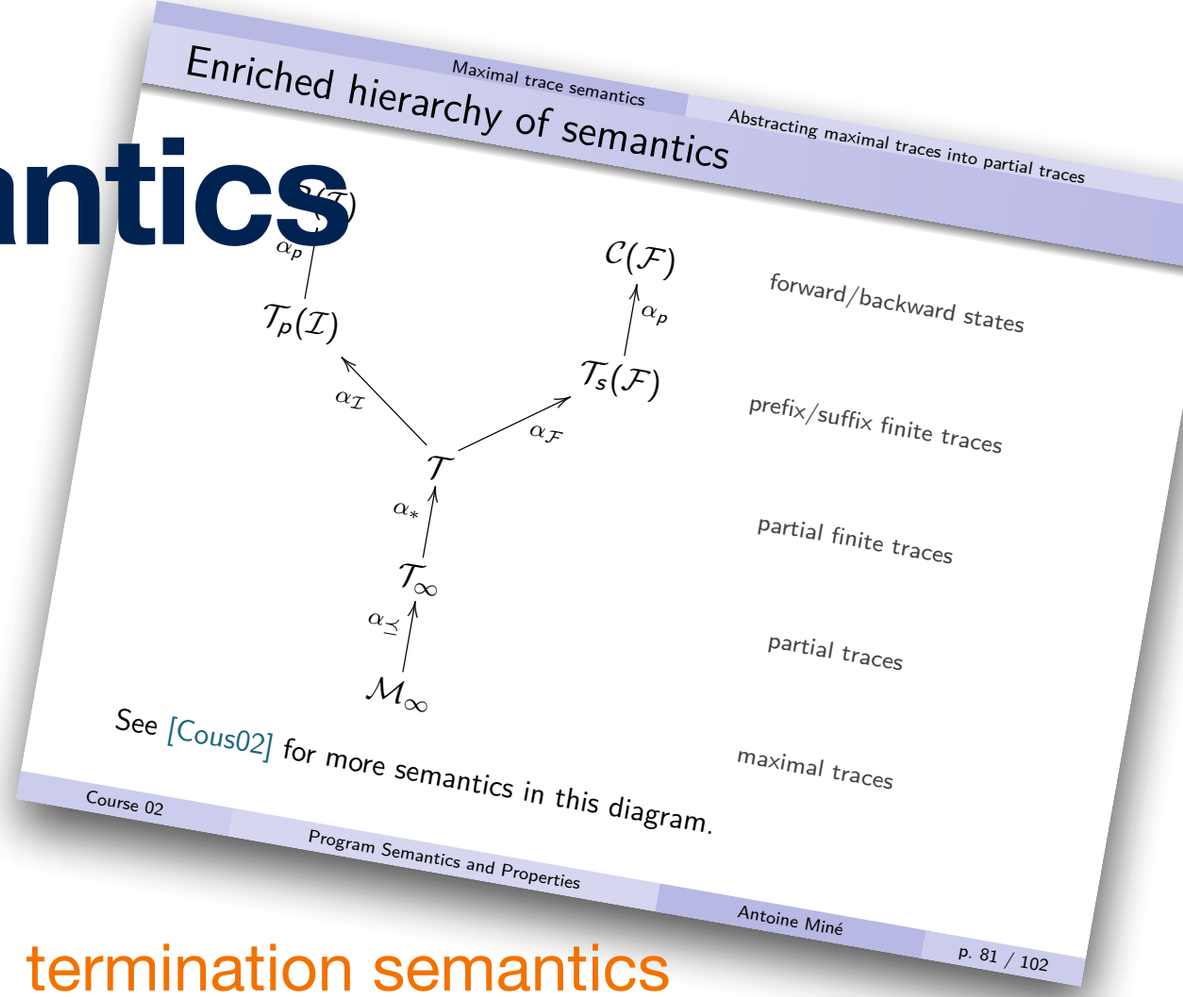
Example

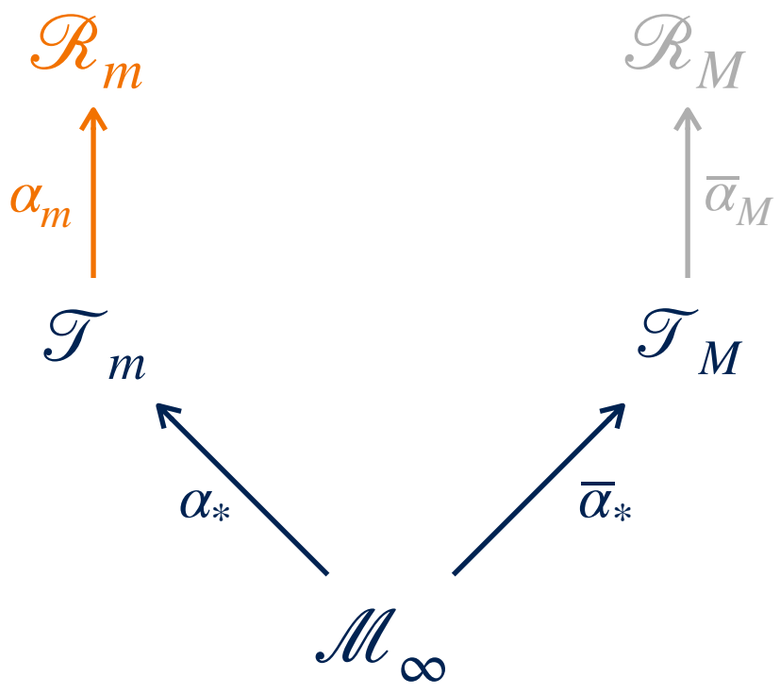
while [1]([-∞, +∞] ≠ 0) **do**
  [2]**skip**
od[3]

$$\mathscr{M}_\infty \stackrel{\text{def}}{=} \{(\mathbf{1}, \rho)(\mathbf{2}, \rho)^*(\mathbf{3}, \rho) \mid \rho \in \mathscr{E}\} \cup \{(\mathbf{1}, \rho)(\mathbf{2}, \rho)^\omega \mid \rho \in \mathscr{E}\}$$

$$\mathscr{T}_M \stackrel{\text{def}}{=} \varnothing$$

Hierarchy of Semantics





termination semantics

termination trace semantics

maximal trace semantics

Potential Termination Semantics

Potential Ranking Abstraction

count execution steps backwards

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

α_m

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

approximation order

$$\alpha_m(T) \stackrel{\text{def}}{=} \alpha_v(\vec{\alpha}(T))$$

$$\alpha_v(\varnothing) \stackrel{\text{def}}{=} \dot{\varnothing}$$

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

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

Potential Termination Semantics

approximation and computational order coincide

$$\mathscr{R}_m \overset{\text{def}}{=} \alpha_m(\mathscr{T}_m) = \text{lfp}^{\preceq} F_m$$

$$F_m(f)\sigma \overset{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathscr{B} \\ \inf\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

State semantics and properties | Backward semantics

Backward co-reachability

$\mathcal{C}(\mathcal{F})$: states co-reachable from $\mathcal{F}$ in the transition system:

$$\mathcal{C}(\mathcal{F}) \overset{\text{def}}{=} \{\, \sigma \mid \exists n \geq 0, \sigma_0, \ldots, \sigma_n : \sigma = \sigma_0, \sigma_n \in \mathcal{F}, \forall i : \sigma_i \to \sigma_{i+1} \,\} = \bigcup_{n \geq 0} \text{pre}_\tau^n(\mathcal{F})$$

where $\text{pre}_\tau(S) \overset{\text{def}}{=} \{\, \sigma \mid \exists \sigma' \in S : \sigma \to \sigma' \,\}$ $(\text{pre}_\tau = \text{post}_{\tau^{-1}})$

$\mathcal{C}(\mathcal{F})$ can also be expressed in fixpoint form:

$$\mathcal{C}(\mathcal{F}) = \text{lfp}\, F_{\mathcal{C}} \text{ where } F_{\mathcal{C}}(S) \overset{\text{def}}{=} \mathcal{F} \cup \text{pre}_\tau(S)$$

Justification: $\mathcal{C}(\mathcal{F})$ in τ is exactly $\mathcal{R}(\mathcal{F})$ in τ^{-1}.

Alternate characterization: $\mathcal{C}(\mathcal{F}) = \text{lfp}_{\mathcal{F}}\ G_{\mathcal{C}}$ where $G_{\mathcal{C}}(S) = S \cup \text{pre}_\tau(S)$

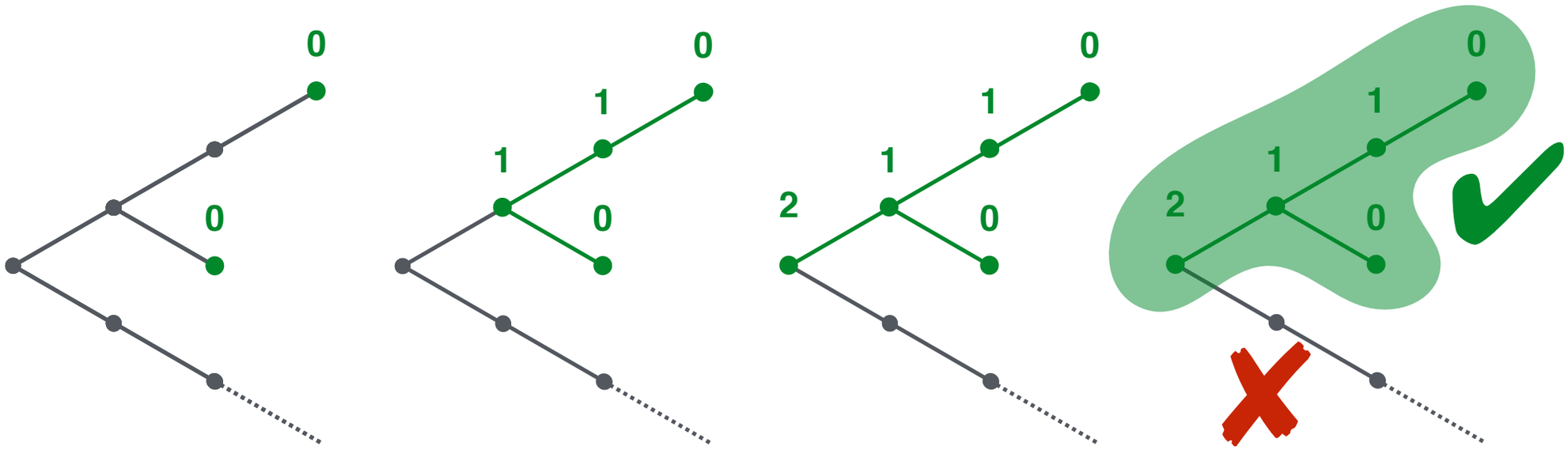
Course 02 | Program Semantics and Properties | Antoine Miné | p. 30 / 102

Potential Termination Semantics

approximation and computational order coincide

$$\mathscr{R}_m \overset{\text{def}}{=} \alpha_m(\mathscr{T}_m) = \text{lfp}^{\preceq} F_m$$

$$F_m(f)\sigma \overset{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathscr{B} \\ \inf\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \in \text{pre}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$



Theorem

A program **may terminate** for traces starting from a set of initial state $\mathscr{I}$ if and only if $\mathscr{I} \subseteq \text{dom}(\mathscr{R}_m)$

Potential Termination Semantics

Exercise

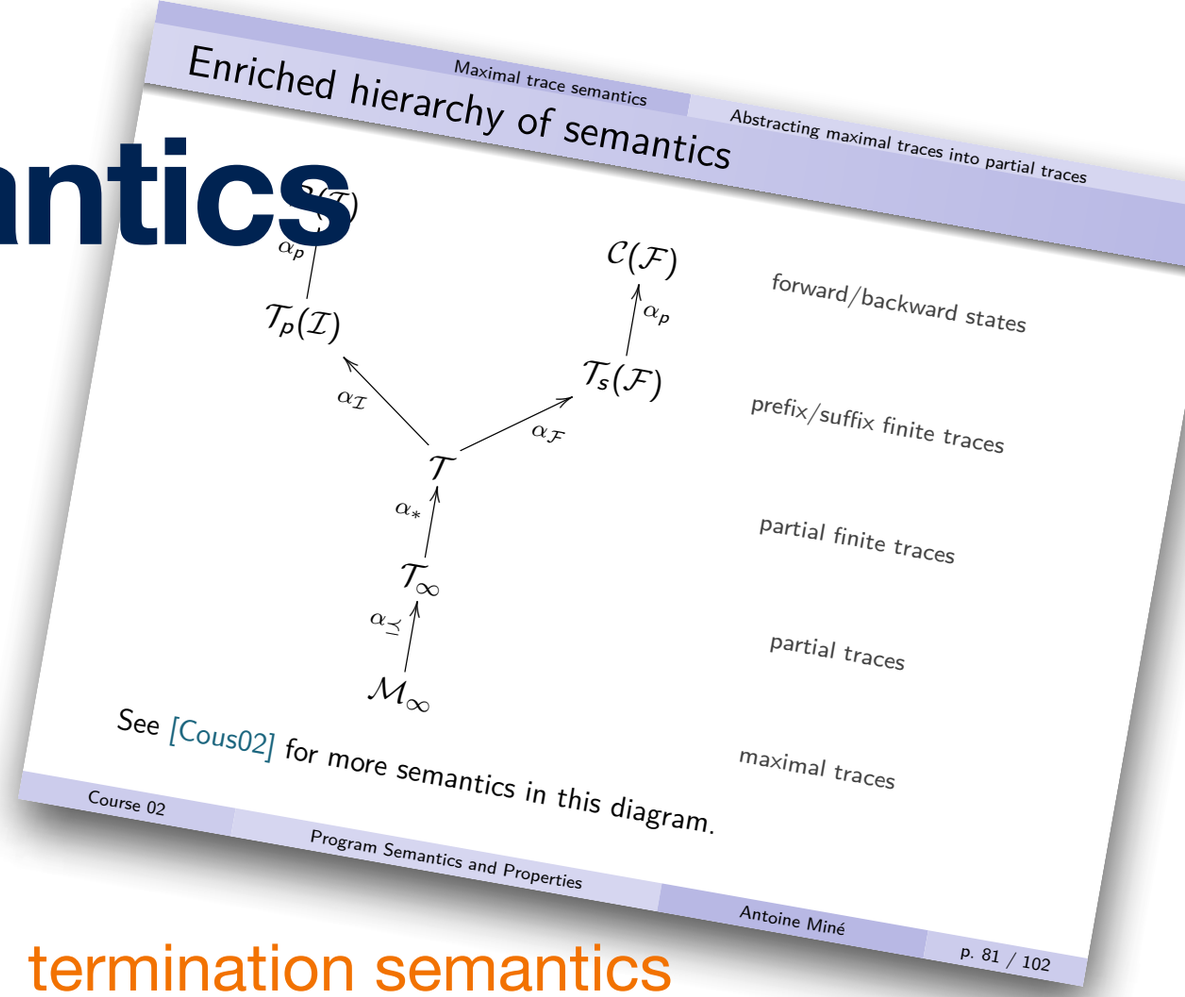
Show that the following fixpoint definition of the potential termination semantics **does not guarantee the existence of a least fixpoint**:

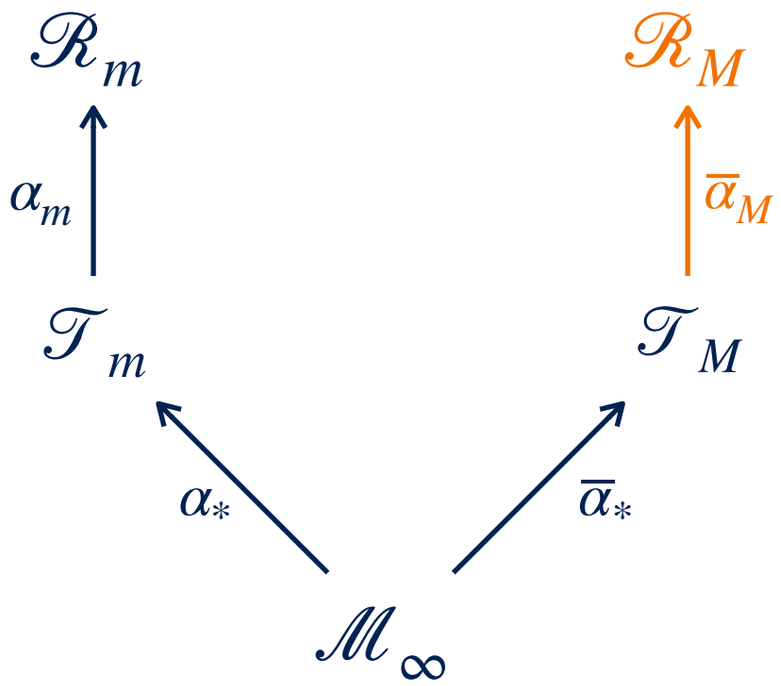
$$\mathscr{R}_m \stackrel{\text{def}}{=} \alpha_m(\mathscr{T}_m) = \text{lfp}^{\preceq} F_m$$

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

Hint: find a program for which the values of the iterates of the potential termination semantics are always increasing

Hierarchy of Semantics





termination semantics

termination trace semantics

maximal trace semantics

Definite Termination Semantics

Ranking Abstraction

count execution steps backwards

$\langle \mathscr{P}(\Sigma^*), \subseteq \rangle$ $\qquad$ $\langle \Sigma \rightharpoonup \mathbb{O}, \preccurlyeq \rangle$

α_m

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

approximation order

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

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

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

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

Definite Termination Semantics

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

computational order

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

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

State semantics and properties | Sufficient precondition semantics

Sufficient preconditions

$\mathcal{S}(\mathcal{Y})$: states with executions staying in $\mathcal{Y}$.

$$\mathcal{S}(\mathcal{Y}) \stackrel{\text{def}}{=} \{\, \sigma \mid \forall n \geq 0, \sigma_0, \ldots, \sigma_n : (\sigma = \sigma_0 \wedge \forall i : \sigma_i \to \sigma_{i+1}) \implies \sigma_n \in \mathcal{Y} \,\} = \bigcap_{n \geq 0} \widetilde{\mathrm{pre}}^n_\tau(\mathcal{Y})$$

where $\widetilde{\mathrm{pre}}_\tau(S) \stackrel{\text{def}}{=} \{\, \sigma \mid \forall \sigma' : \sigma \to \sigma' \implies \sigma' \in S \,\}$

(states such that all successors satisfy S, $\widetilde{\mathrm{pre}}$ is a complete $\cap$−morphism)

$\mathcal{S}(\mathcal{Y})$ can be expressed in fixpoint form:

$$\mathcal{S}(\mathcal{Y}) = \mathrm{gfp}\, F_{\mathcal{S}} \text{ where } F_{\mathcal{S}}(S) \stackrel{\text{def}}{=} \mathcal{Y} \cap \widetilde{\mathrm{pre}}_\tau(S)$$

proof sketch: similar to that of $\mathcal{R}(\mathcal{I})$, in the dual.

$F_{\mathcal{S}}$ is continuous in the dual CPO $(\mathcal{P}(\Sigma), \supseteq)$, because $\widetilde{\mathrm{pre}}_\tau$ is:
$F_{\mathcal{S}}(\cap_{i \in I} A_i) = \cap_{i \in I} F_{\mathcal{S}}(A_i)$.
By Kleene's theorem in the dual, $\mathrm{gfp}\, F_{\mathcal{S}} = \cap_{n \in \mathbb{N}} F^n_{\mathcal{S}}(\Sigma)$.
We would prove by recurrence that $F^n_{\mathcal{S}}(\Sigma) = \cap_{i<n} \widetilde{\mathrm{pre}}^i_\tau(\mathcal{Y})$.

Course 02 | Program Semantics and Properties | Antoine Miné | p. 35 / 102

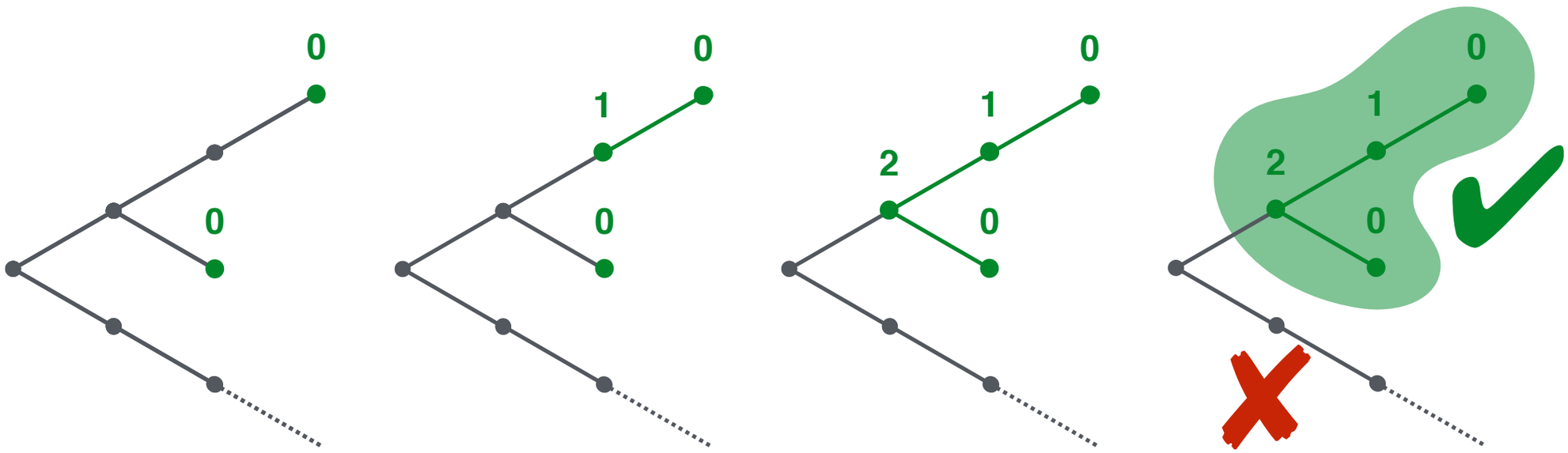
Definite Termination Semantics

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

computational order

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

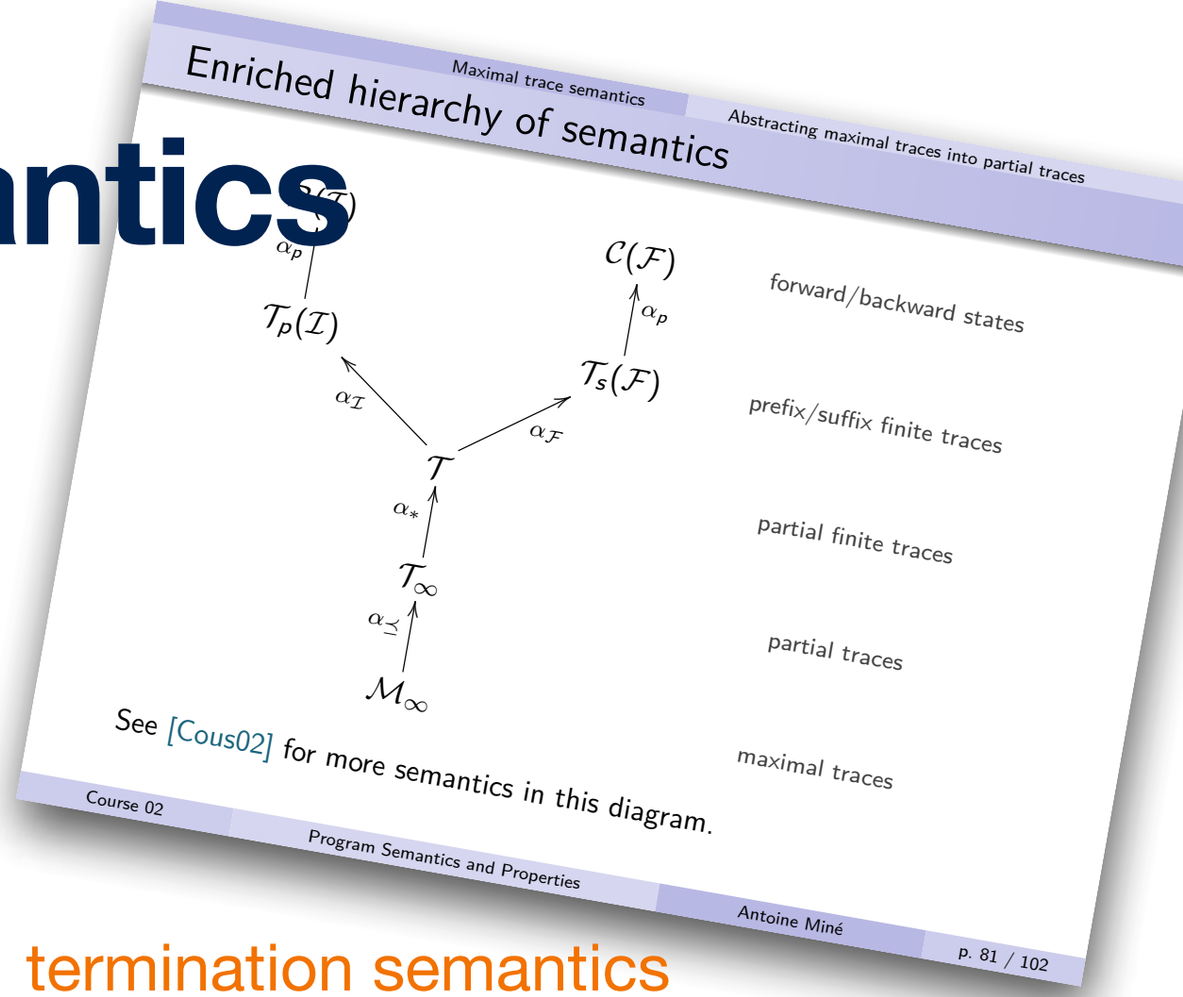
$$\overline{F}_M(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathscr{B} \\ \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\} & \sigma \in \widetilde{\text{pre}}_\tau(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

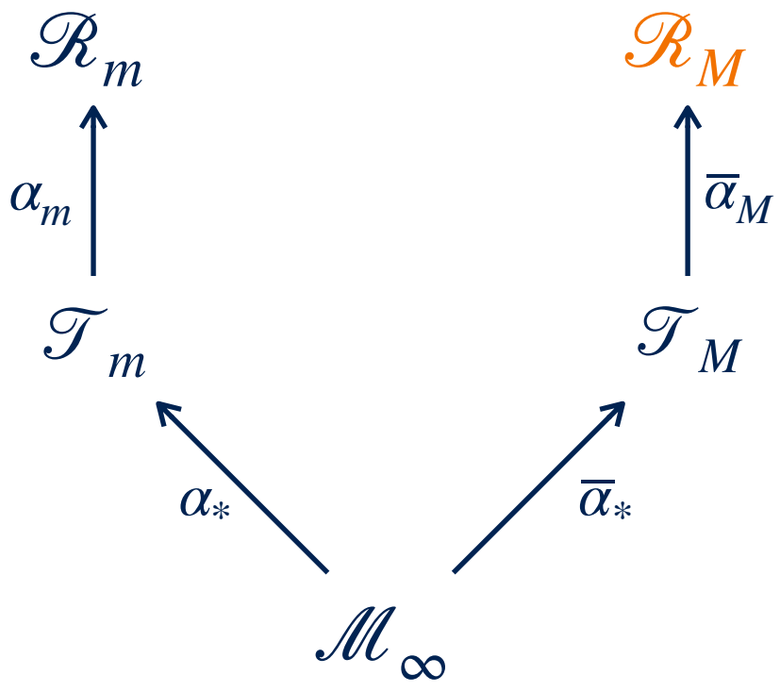


Theorem

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

Hierarchy of Semantics





termination semantics

termination trace semantics

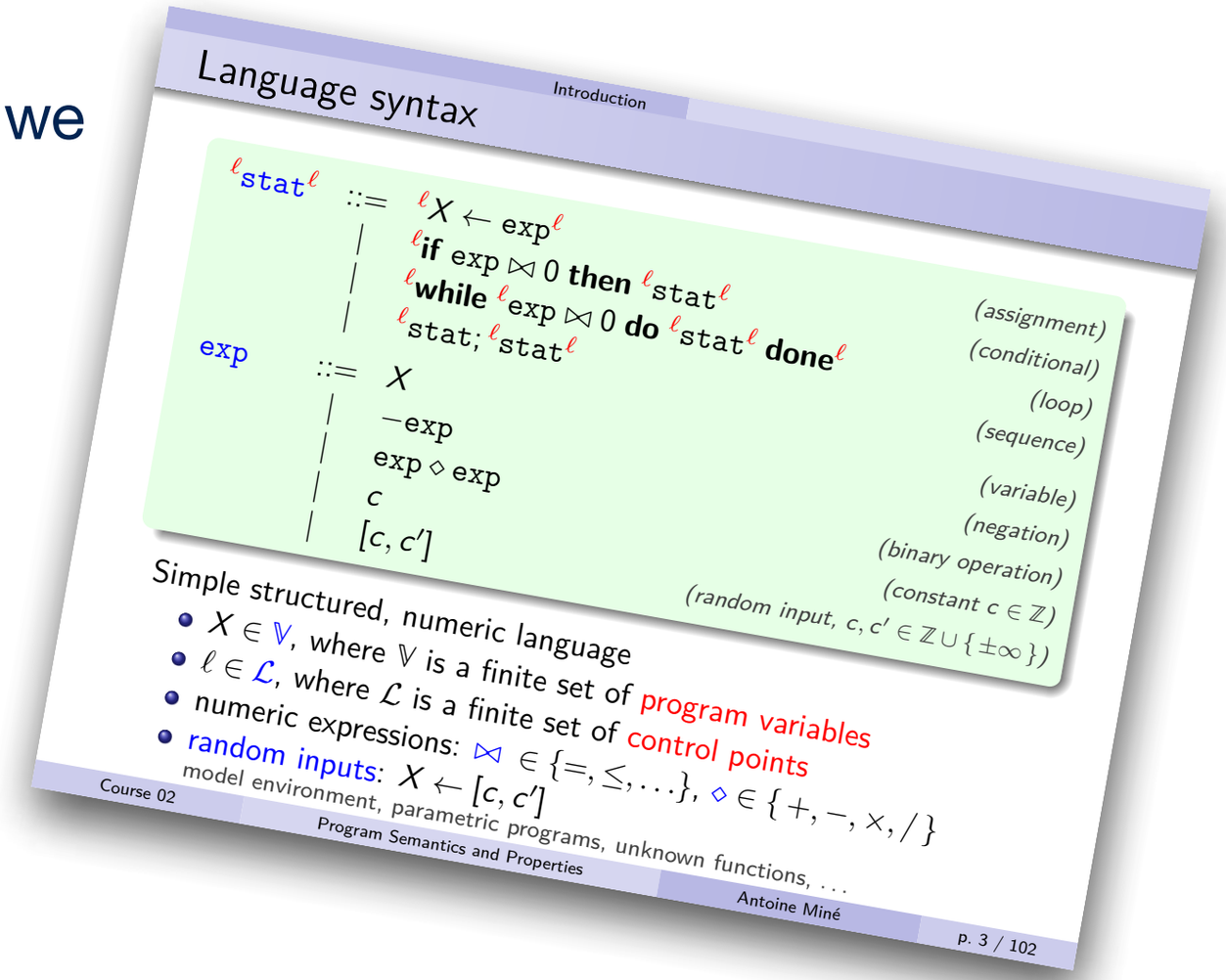
maximal trace semantics

Denotational Definite Termination Semantics

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

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

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



Denotational Definite Termination Semantics

$$\mathscr{R}_M[\![{}^{\ell} X \leftarrow e]\!]$$

$$\mathscr{R}_M[\![{}^{\ell} X \leftarrow e]\!]f \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} \sup\{f(\rho[X \mapsto v])+1 \mid v \in E[\![e]\!]\rho\} & \exists \bar{v} \in E[\![e]\!]\rho \wedge \\ & \forall v \in E[\![e]\!]\rho\colon \rho[X \mapsto v] \in \mathrm{dom}(f) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Example:

Let $\mathbb{V} = \{x\}$ and $f\colon \mathscr{E} \rightharpoonup \mathbb{O}$ defined as follows:

$$f(\rho) \stackrel{\text{def}}{=} \begin{cases} 2 & \rho(x) = 1 \\ 3 & \rho(x) = 2 \\ \text{undefined} & \text{otherwise} \end{cases}$$

We have

$$\mathscr{R}_M[\![x \leftarrow x + [1,2]]\!]f \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 4 & \rho(x) = 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Denotational Definite Termination Semantics

$$\mathscr{R}_M[\![\textbf{if}\ ^{\ell}\ e \bowtie 0\ \textbf{then}\ s]\!]$$

$$\mathscr{R}_M[\![\textbf{if}\ ^{\ell}\ e \bowtie 0\ \textbf{then}\ s]\!]f \stackrel{\text{def}}{=} \lambda\rho\,.\ \begin{cases} ① \\ ② \\ ③ \\ \text{undefined} \quad \text{otherwise} \end{cases}$$

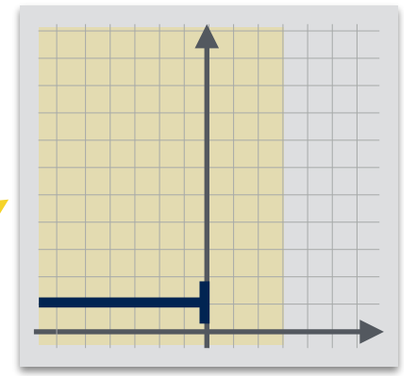
① $\sup\{\mathscr{R}_M[\![s]\!]f(\rho) + 1,\ f(\rho) + 1\}$ $\quad \rho \in \text{dom}(\mathscr{R}_M[\![s]\!]f) \cap \text{dom}(f) \wedge \exists v_1, v_2 \in E[\![e]\!]\rho :\ v_1 \bowtie 0 \wedge v_2 \not\bowtie 0$

② $\mathscr{R}_M[\![s]\!]f(\rho) + 1$ $\quad \rho \in \text{dom}(\mathscr{R}_M[\![s]\!]f) \wedge \forall v \in E[\![e]\!]\rho :\ v \bowtie 0$

③ $f(\rho) + 1$ $\quad \rho \in \text{dom}(f) \wedge \forall v \in E[\![e]\!]\rho :\ v \not\bowtie 0$

Denotational Definite Termination Semantics

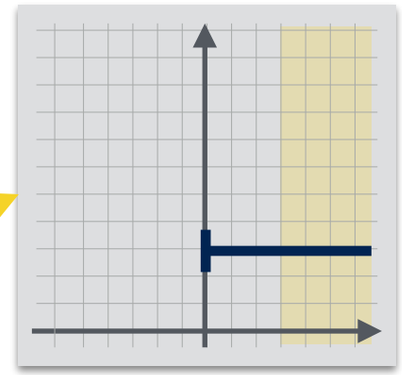
$\mathscr{R}_M[\![\textbf{if}\ ^{\ell}\ e \bowtie 0\ \textbf{then}\ s]\!]$ **(continue)**



Example:

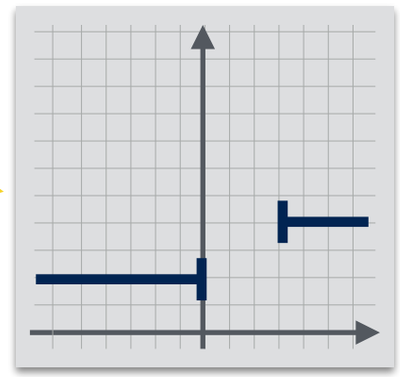
Let $\mathbb{V} = \{x\}$ and $f\colon \mathscr{E} \rightharpoonup \mathbb{O}$, and $\mathscr{R}_M[\![s]\!]f$ defined as follows:

$$f \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 1 & \rho(x) \leq 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$



$$\mathscr{R}_M[\![s]\!]f \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 3 & 0 \leq \rho(x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

We have



$$\mathscr{R}_M[\![\textbf{if}\ 3 - x < 0\ \textbf{then}\ s]\!]f \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 2 & \rho(x) \leq 0 \\ 4 & 3 < \rho(x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

and $\mathscr{R}_M[\![\textbf{if}\ [-\infty, +\infty] \neq 0\ \textbf{then}\ s]\!]f \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} 4 & \rho(x) = 0 \\ \text{undefined} & \text{otherwise} \end{cases}$

Denotational Definite Termination Semantics

$$\mathscr{R}_M[\![\textbf{while}\ ^{\ell}\ e \bowtie 0\ \textbf{do}\ s\ \textbf{done}]\!]$$

$$\mathscr{R}_M[\![\textbf{while}\ ^{\ell}\ e \bowtie 0\ \textbf{do}\ s\ \textbf{done}]\!]f \stackrel{\text{def}}{=} \mathsf{lfp}^{\preceq}_{\dot{\varnothing}} \overline{F}_M$$

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

computational order

$$F_M(x) \stackrel{\text{def}}{=} \lambda\rho\,.\begin{cases} ① \\ ② \\ ③ \\ \text{undefined} \quad \text{otherwise} \end{cases}$$

① $\sup\{\mathscr{R}_M[\![s]\!]x(\rho) + 1,\ f(\rho) + 1\}$ $\quad \rho \in \mathsf{dom}(\mathscr{R}_M[\![s]\!]x) \cap \mathsf{dom}(f) \wedge \exists v_1, v_2 \in E[\![e]\!]\rho : v_1 \bowtie 0 \wedge v_2 \not\bowtie 0$

② $\mathscr{R}_M[\![s]\!]x(\rho) + 1$ $\quad \rho \in \mathsf{dom}(\mathscr{R}_M[\![s]\!]x) \wedge \forall v \in E[\![e]\!]\rho : v \bowtie 0$

③ $f(\rho) + 1$ $\quad \rho \in \mathsf{dom}(f) \wedge \forall v \in E[\![e]\!]\rho : v \not\bowtie 0$

Denotational Definite Termination Semantics

$\mathscr{R}_M[\![s_1; s_2]\!]$

$\mathscr{R}_M[\![s_1; s_2]\!]f \stackrel{\text{def}}{=} \mathscr{R}_M[\![s_1]\!](\mathscr{R}_M[\![s_2]\!]f)$

Denotational Definite Termination Semantics

Definition

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

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

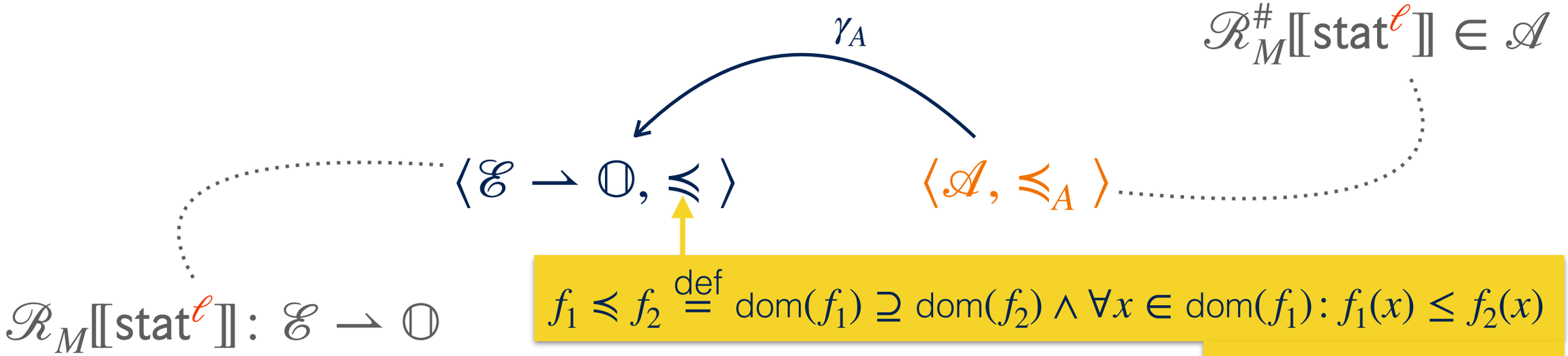
where $\mathscr{R}_M[\![\text{stat}]\!] : (\mathscr{E} \rightharpoonup \mathbb{O}) \to (\mathscr{E} \rightharpoonup \mathbb{O})$ is the definite termination semantics of each program instruction stat

Theorem

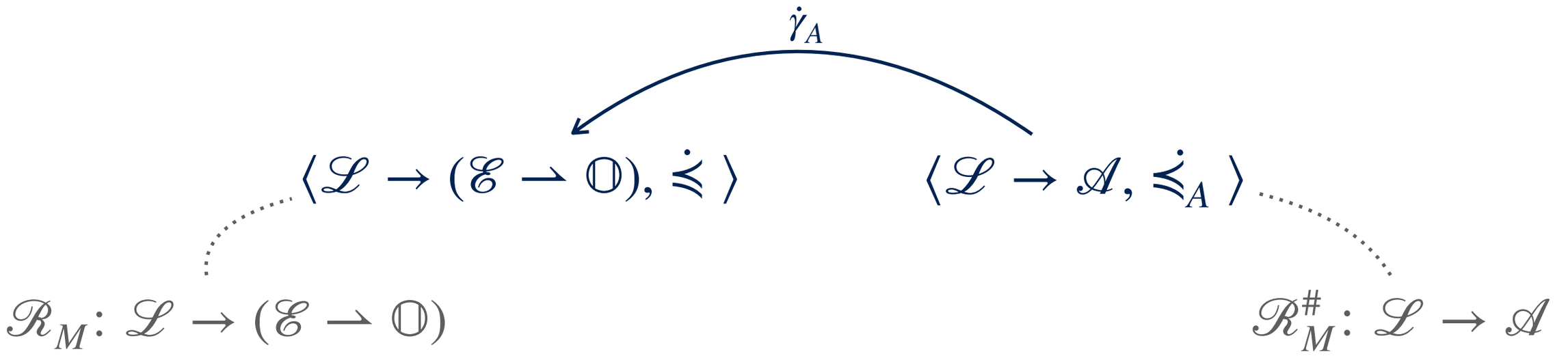
A program stat^ℓ **must terminate** for traces starting from a set of initial states $\mathscr{I}$ if $\mathscr{I} \subseteq \text{dom}(\mathscr{R}_m[\![\text{stat}^\ell]\!])$

Piecewise-Defined Ranking Functions Abstract Domain

Concretization-Based Piecewise Abstraction



By *pointwise lifiting* we obtain an abstraction $\mathscr{R}^{\#}_M$ of $\mathscr{R}_M$:

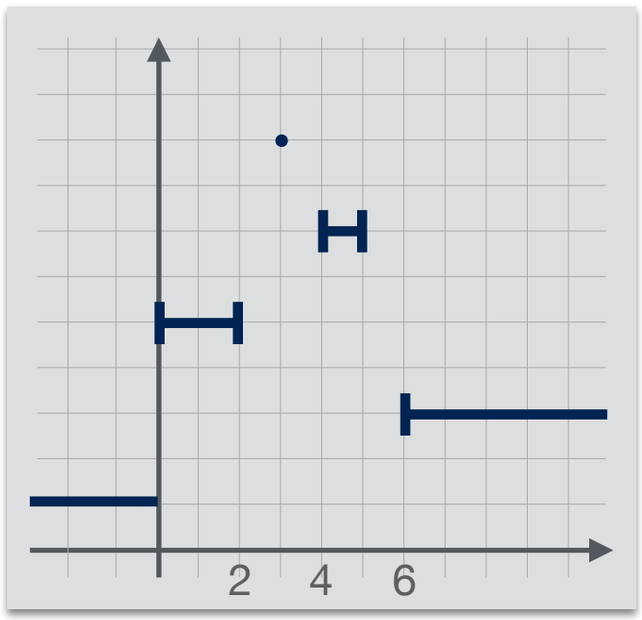


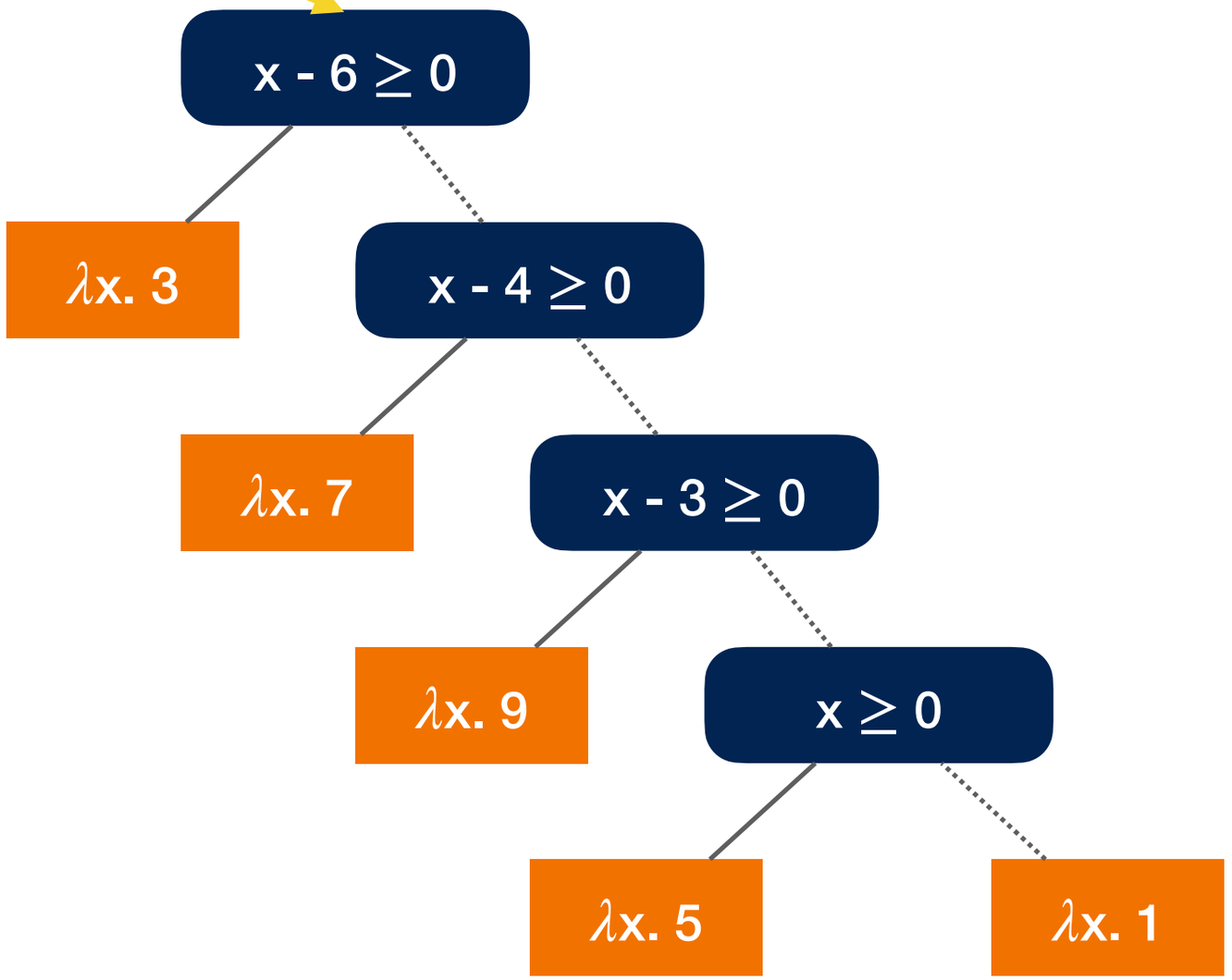
Piecewise-Defined Ranking Functions Abstract Domain

$\langle \mathscr{A}, \preccurlyeq_A \rangle$

Example

$^1x \leftarrow [-\infty, +\infty]$
while $^2(x \geq 0)$ **do**
$^3x \leftarrow -2 \cdot x + 10$
od^4

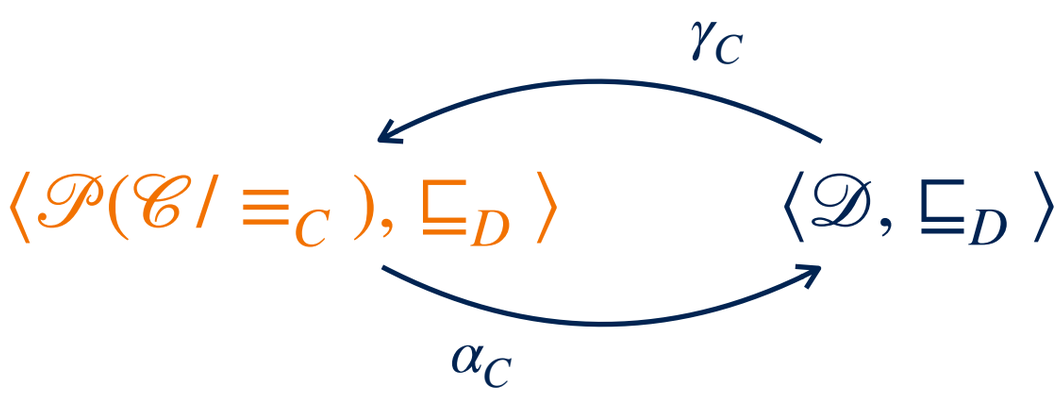


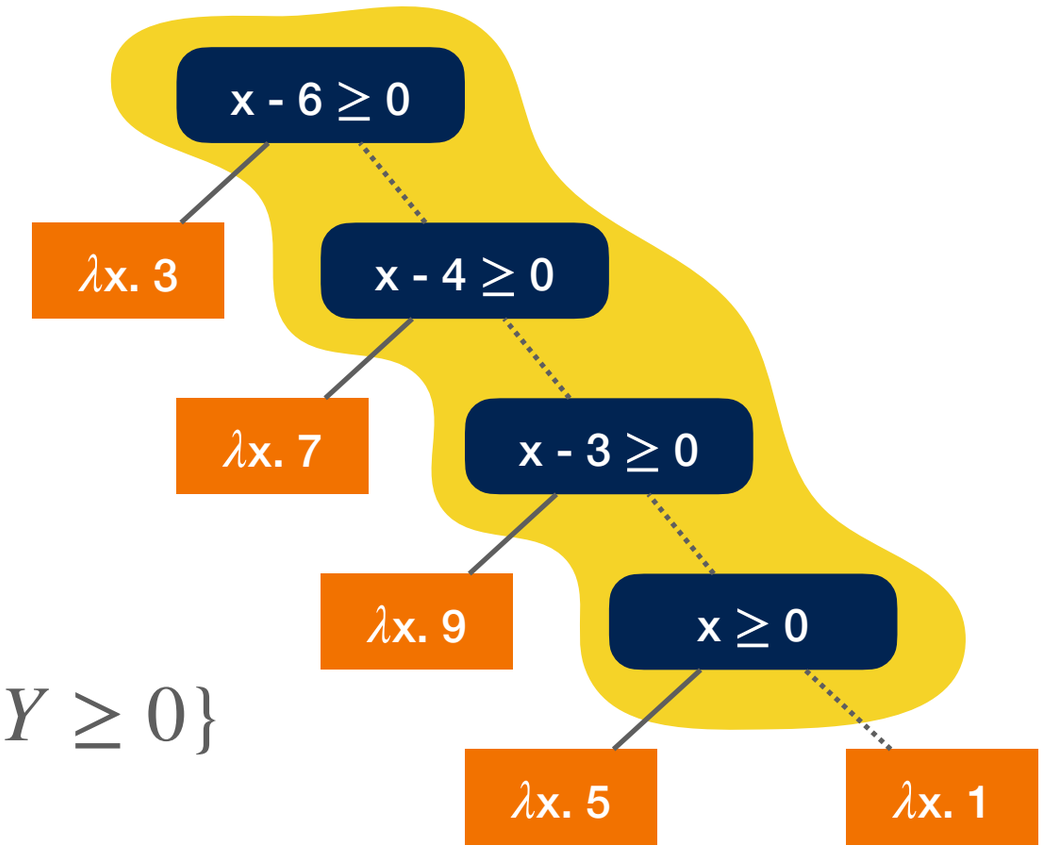


Piecewise-Defined Ranking Functions Abstract Domain

Linear Constraints Auxiliary Abstract Domain

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





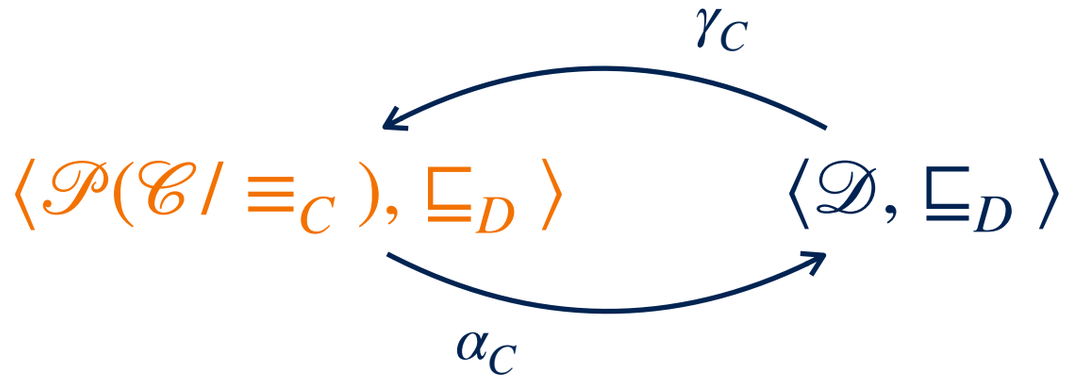
Example:

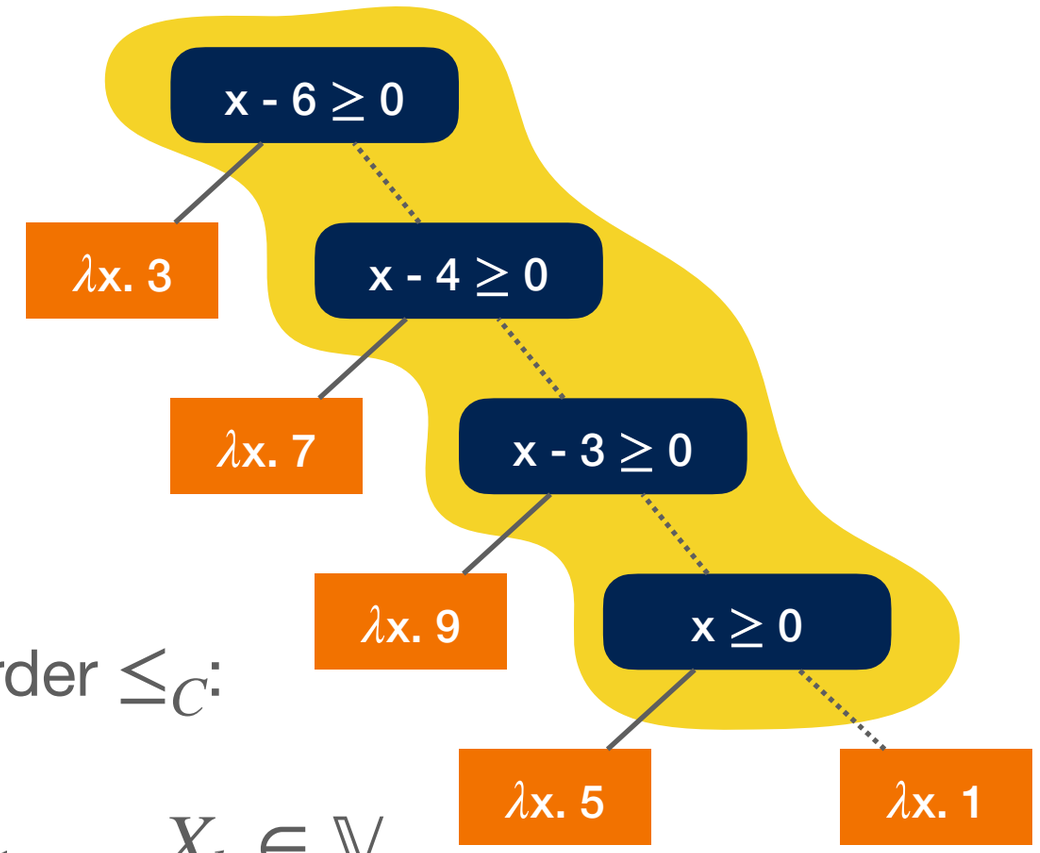
$$X \to [-\infty, 3] \qquad Y \to [0, +\infty] \qquad \xrightarrow{\gamma_C} \qquad \{3 - X \geq 0,\ Y \geq 0\}$$

Piecewise-Defined Ranking Functions Abstract Domain

Linear Constraints Auxiliary Abstract Domain

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





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

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

Piecewise-Defined Ranking Functions Abstract Domain

Functions Auxiliary Abstract Domain

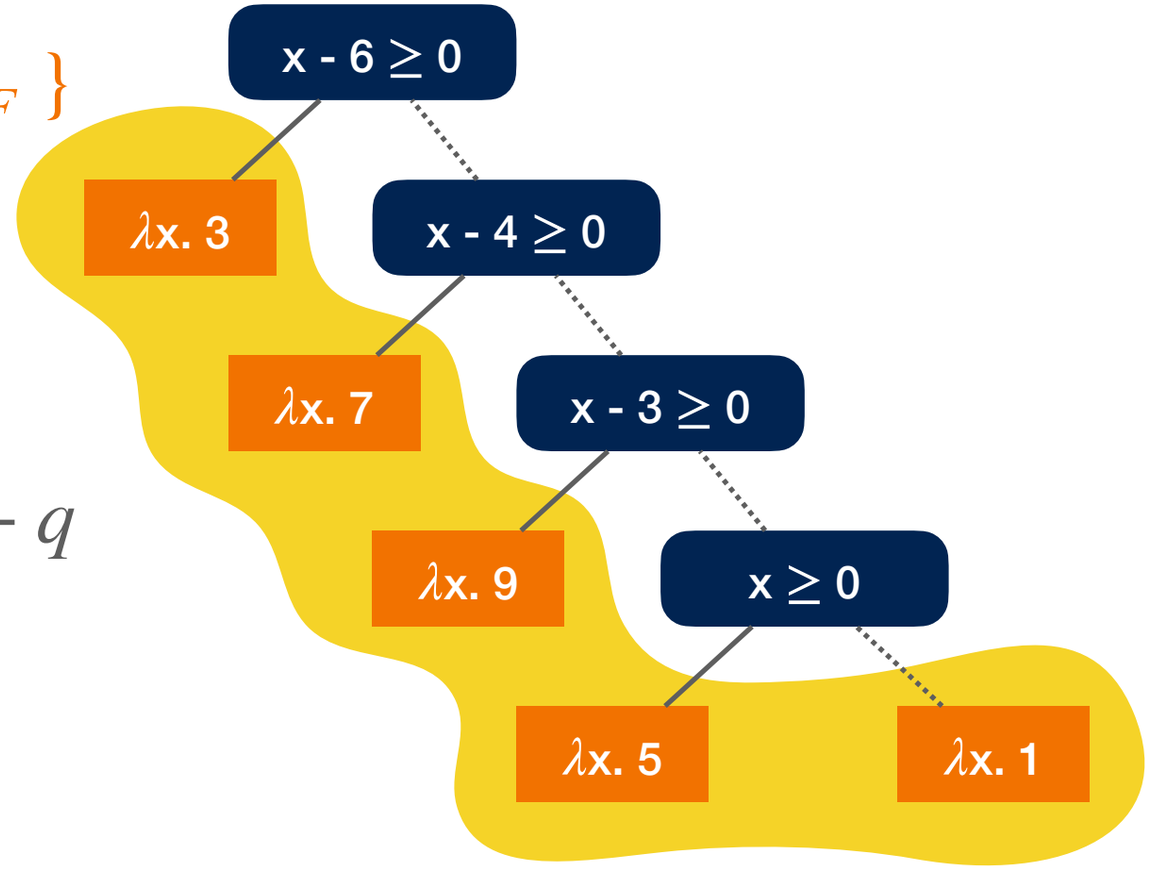
- Parameterized by an ***underlying numerical abstract domain*** $\langle \mathscr{D}, \sqsubseteq_D \rangle$
- $\mathscr{F} \overset{\text{def}}{=} \{ \bot_F \} \cup (\mathbb{Z}^{|\mathbb{V}|} \to \mathbb{N}) \cup \{ \top_F \}$

We consider **affine functions**:

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

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

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



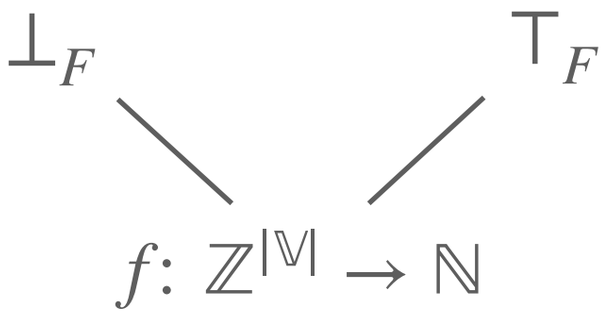
Piecewise-Defined Ranking Functions Abstract Domain

Functions Auxiliary Abstract Domain (continue)

- **approximation order** $\preccurlyeq_F [D]$, where $D \in \mathscr{D}$:
  - between defined leaf nodes:

$$f_1 \preccurlyeq_F [D]\ f_2 \overset{\text{def}}{=} \forall \rho \in \gamma_D(D) : f_1(\ldots, \rho(X_i), \ldots) \leq f_2(\ldots, \rho(X_i), \ldots)$$

  - otherwise (i.e., when one or both leaf nodes are undefined):



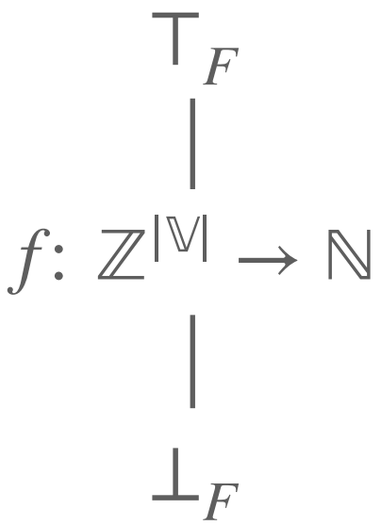
Piecewise-Defined Ranking Functions Abstract Domain

Functions Auxiliary Abstract Domain (continue)

- **computational order** $\sqsubseteq_F[D]$, where $D \in \mathscr{D}$:
  - between defined leaf nodes:

$$f_1 \sqsubseteq_F[D] f_2 \overset{\text{def}}{=} \forall \rho \in \gamma_D(D) : f_1(\ldots, \rho(X_i), \ldots) \leq f_2(\ldots, \rho(X_i), \ldots)$$

  - otherwise (i.e., when one or both leaf nodes are undefined):



Piecewise-Defined Ranking Functions Abstract Domain

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

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

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

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

$$\overline{\gamma}_A[C](\text{LEAF}: f) \overset{\text{def}}{=} \gamma_F[\alpha_C(C)](f)$$

$$\overline{\gamma}_A[C](\text{NODE}\{c\}: t_1; t_2) \overset{\text{def}}{=} \overline{\gamma}_A[C \cup \{c\}](t_1) \dot{\cup} \overline{\gamma}_A[C \cup \{\neg c\}](t_2)$$

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

$$\gamma_F[D](\bot_F) \overset{\text{def}}{=} \dot{\varnothing}$$

$$\gamma_F[D](f) \overset{\text{def}}{=} \forall \rho \in \gamma_D(D): f(\ldots, \rho(X_i), \ldots)$$

$$\gamma_F[D](\top_F) \overset{\text{def}}{=} \dot{\varnothing}$$

Piecewise-Defined Ranking Functions Abstract Domain

Abstract Domain Operators

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

- 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 $\triangledown_A$

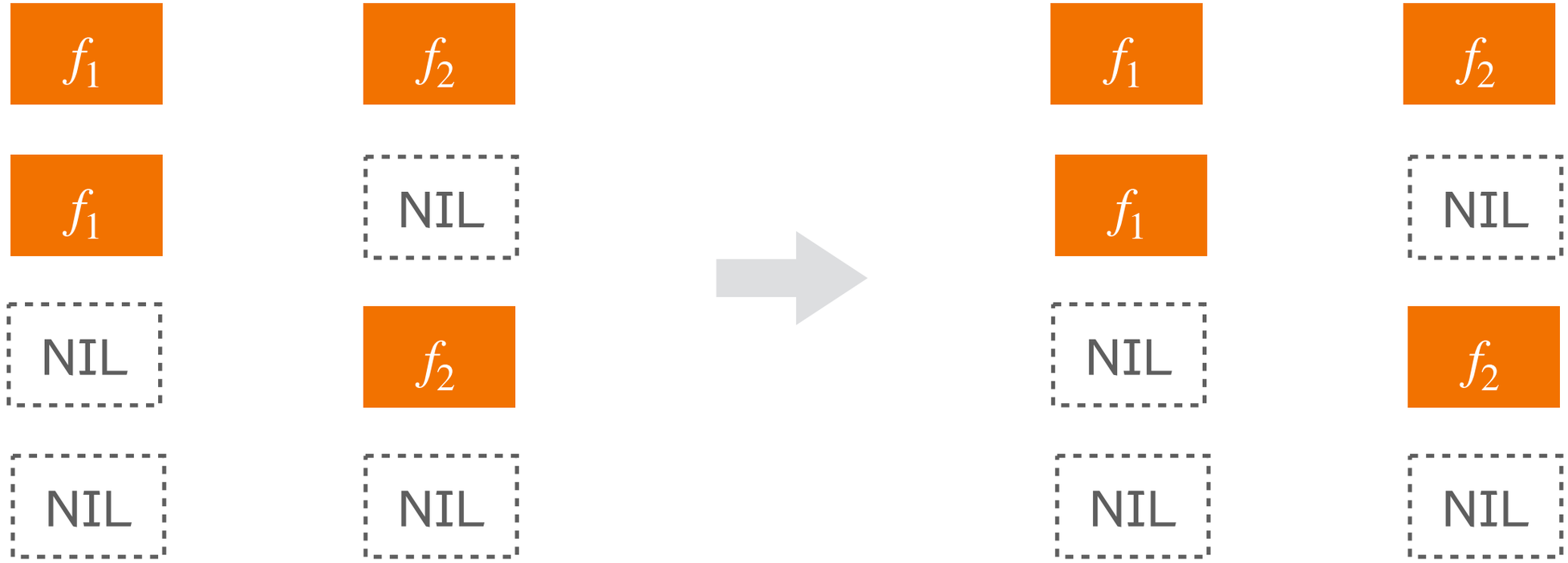
- The **unary operators** rely on a tree pruning algorithm
  - assignment $\overleftarrow{\mathsf{ASSIGN}}_A[\![X \leftarrow e]\!]$
  - test $\mathsf{FILTER}_A[\![e]\!]$

Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification

Goal: find a **common refinement** for the given decision trees

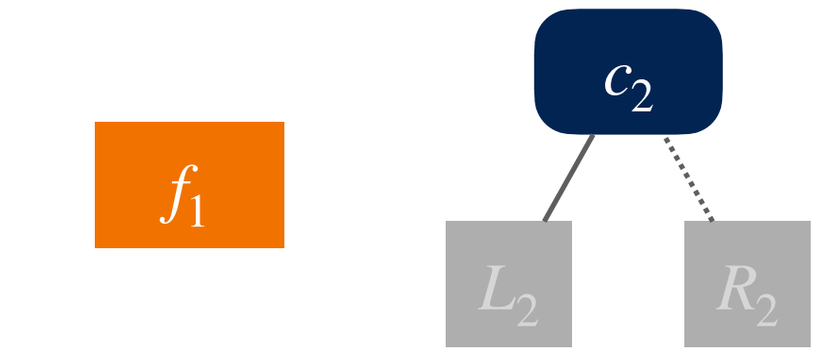
- Base cases:

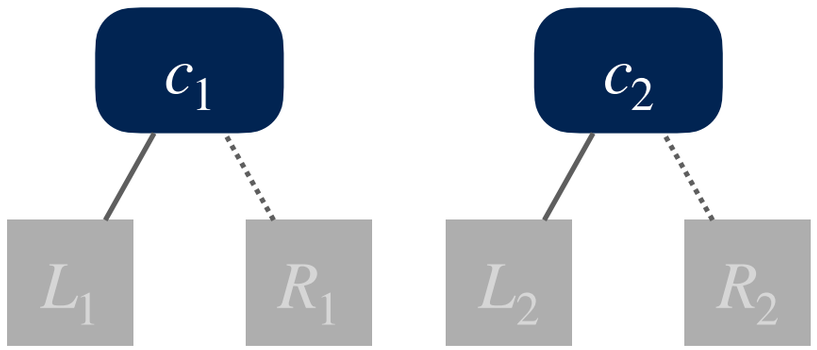


Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

- Case ①



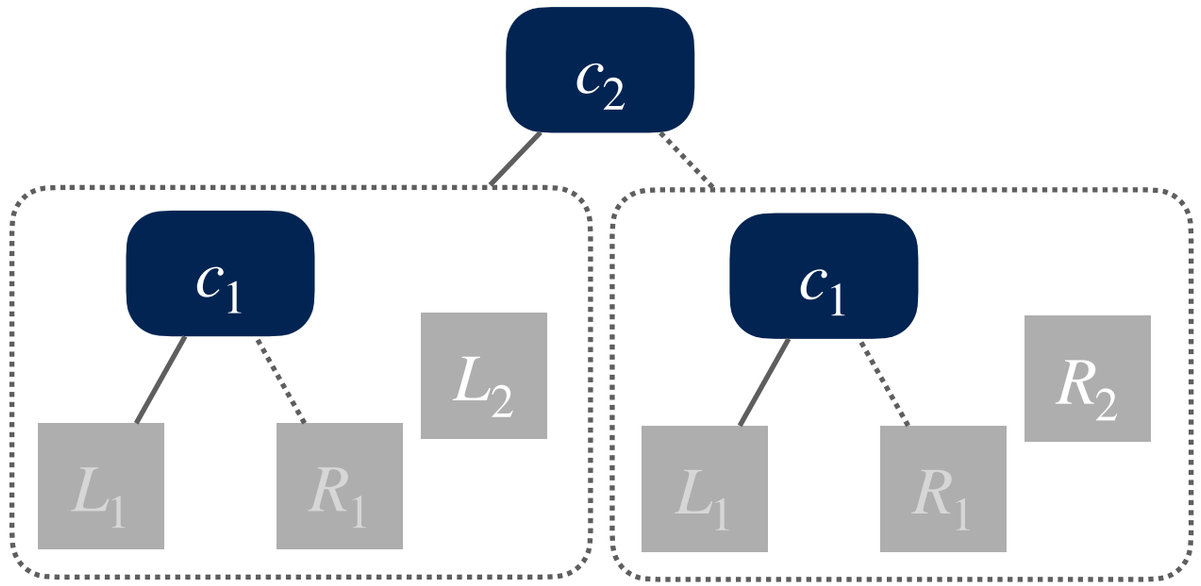


$c_2 \leq_C c_1$

① c_2 is redundant

① $\neg c_2$ is redundant

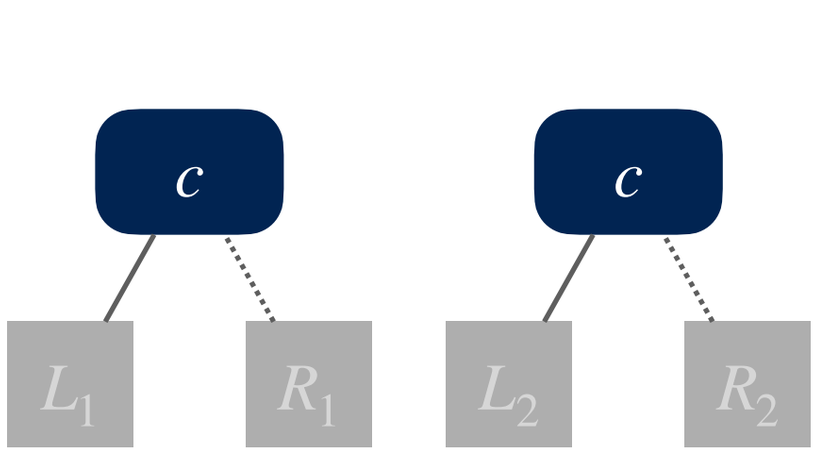
① c_2 is added to t_1



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

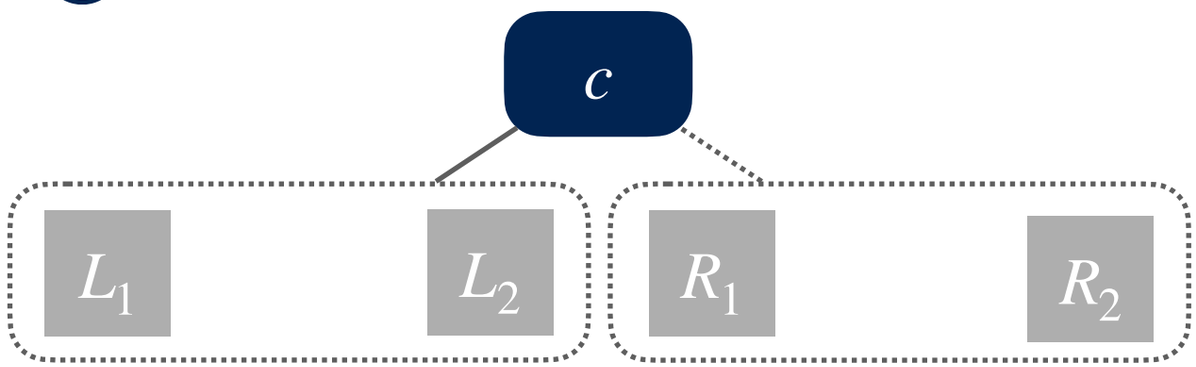
- Case ② (simmetric to ①)
- Case ③



1a c is redundant

1b $\neg c$ is redundant

1c c is kept in t_1 and t_2



Piecewise-Defined Ranking Functions Abstract Domain

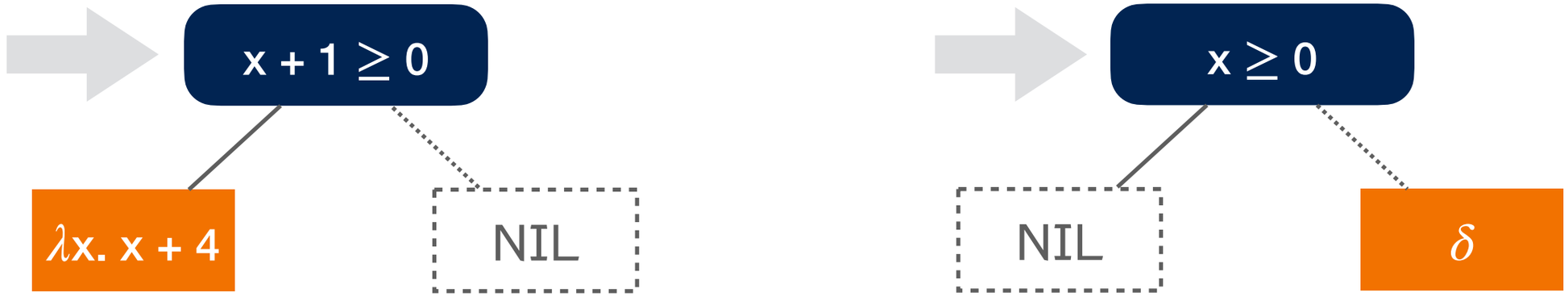
Tree Unification (continue)

Example

Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

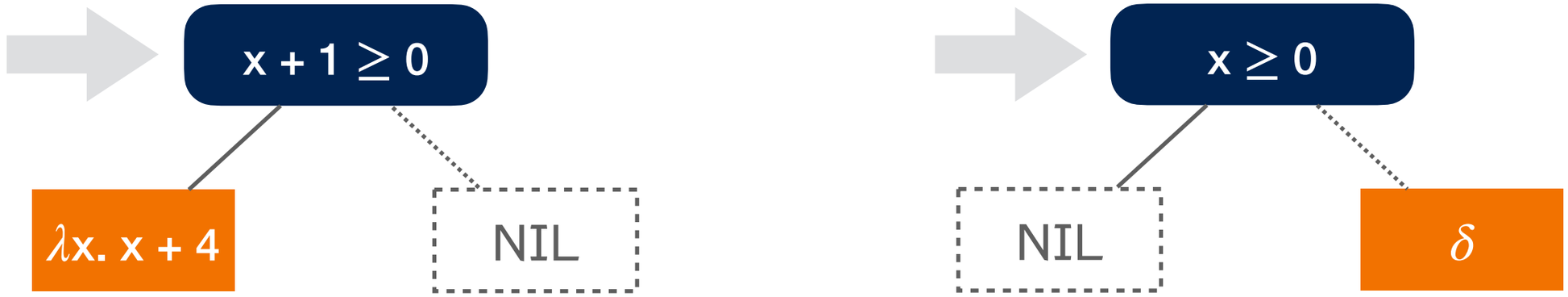
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

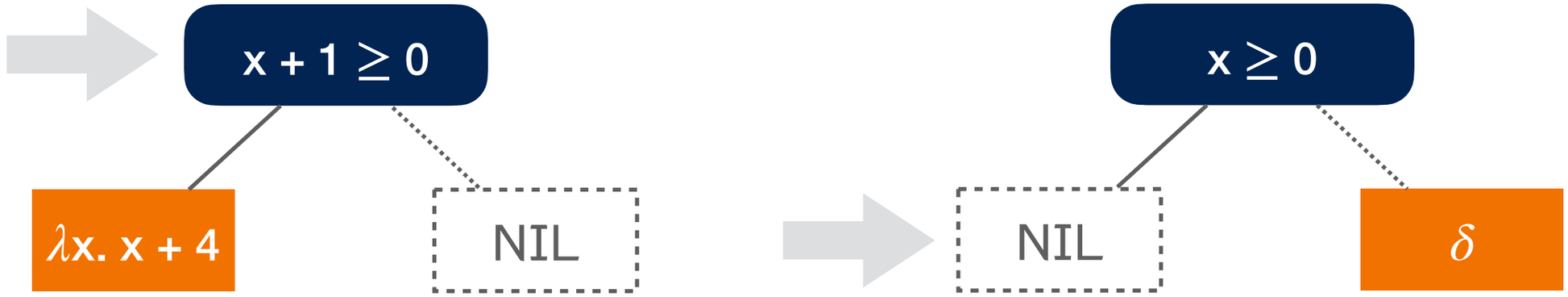
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

Example



Piecewise-Defined Ranking Functions Abstract Domain

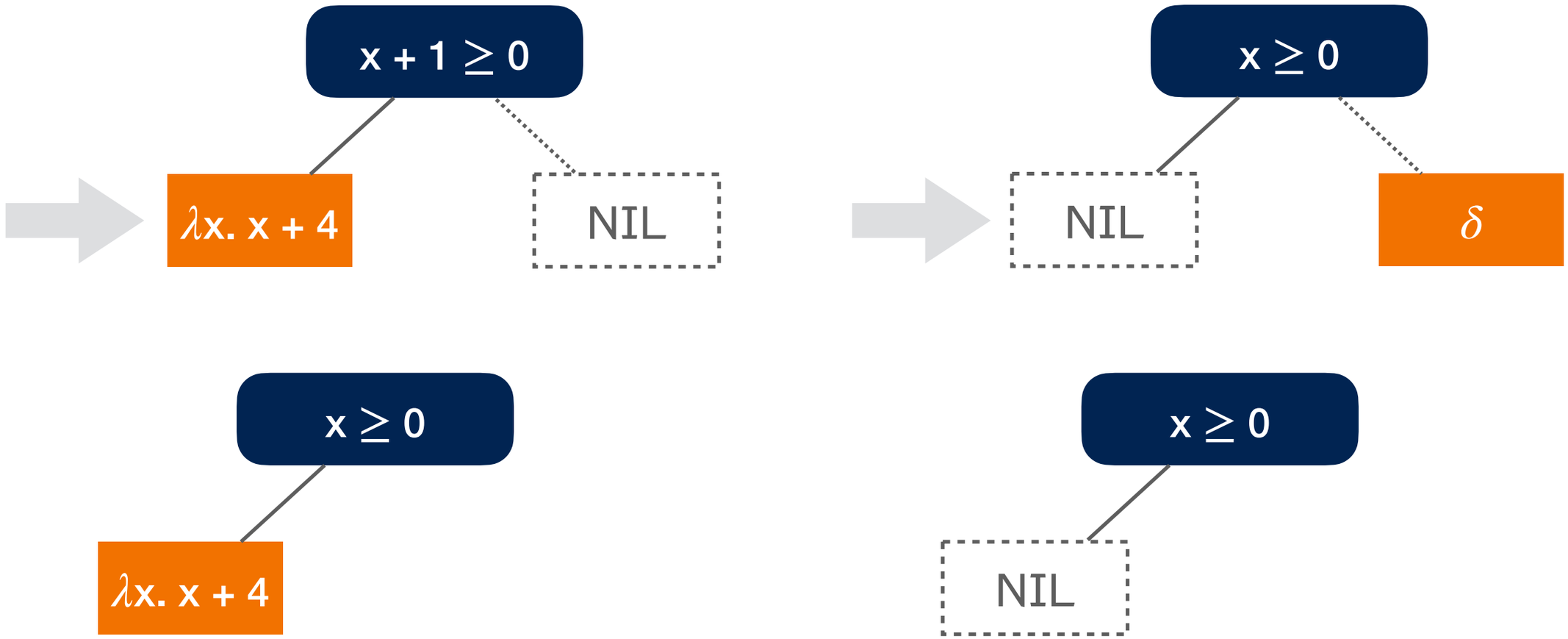
Tree Unification (continue)

Example

Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

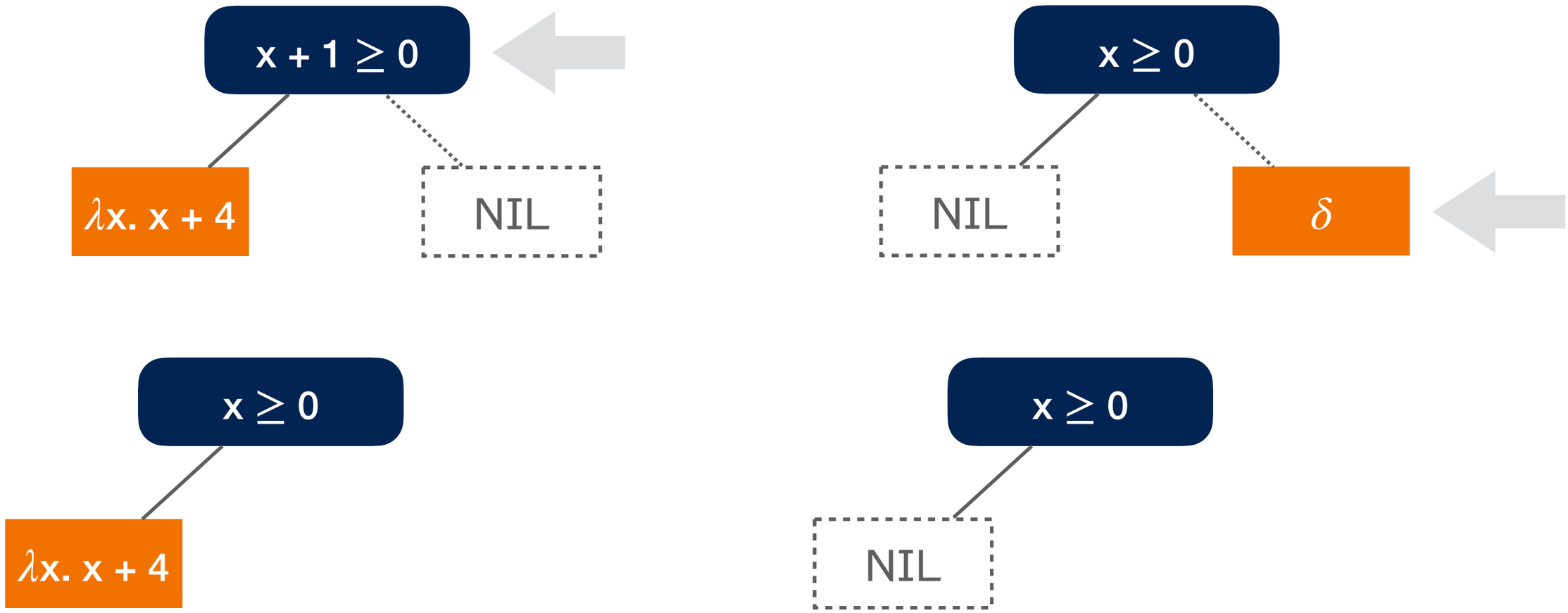
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

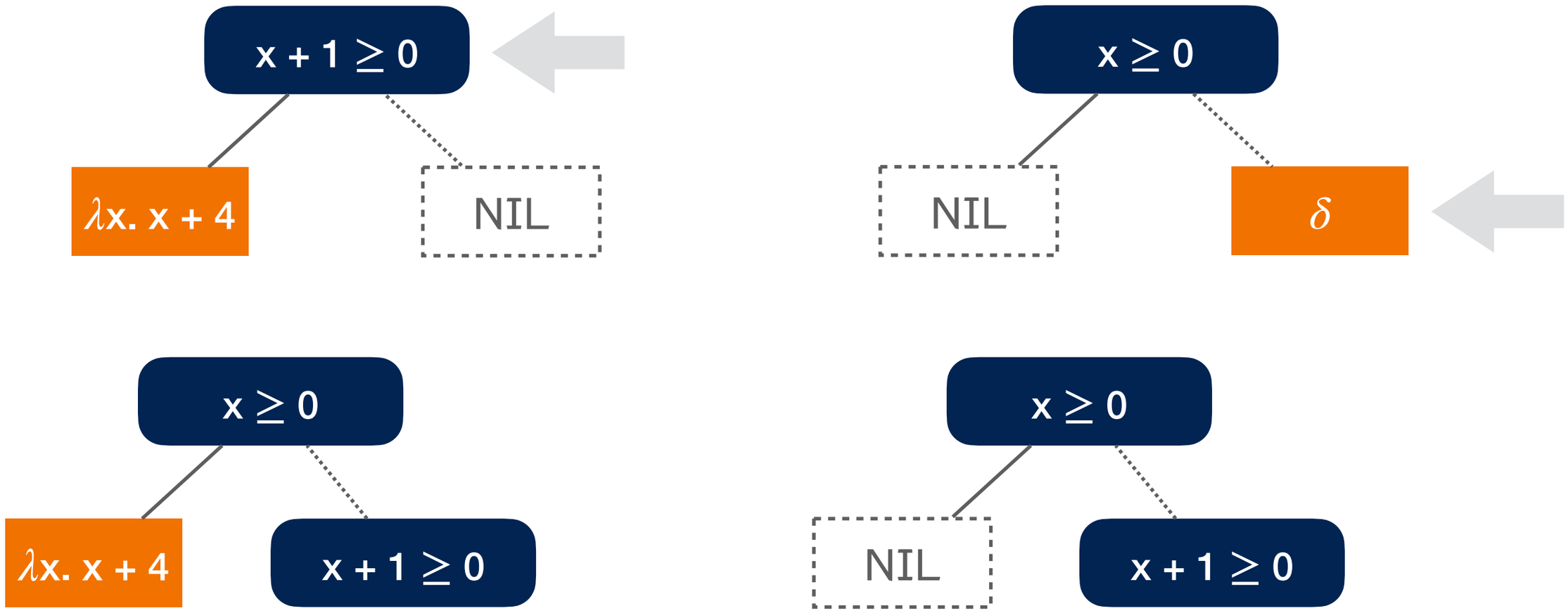
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

Example



Piecewise-Defined Ranking Functions Abstract Domain

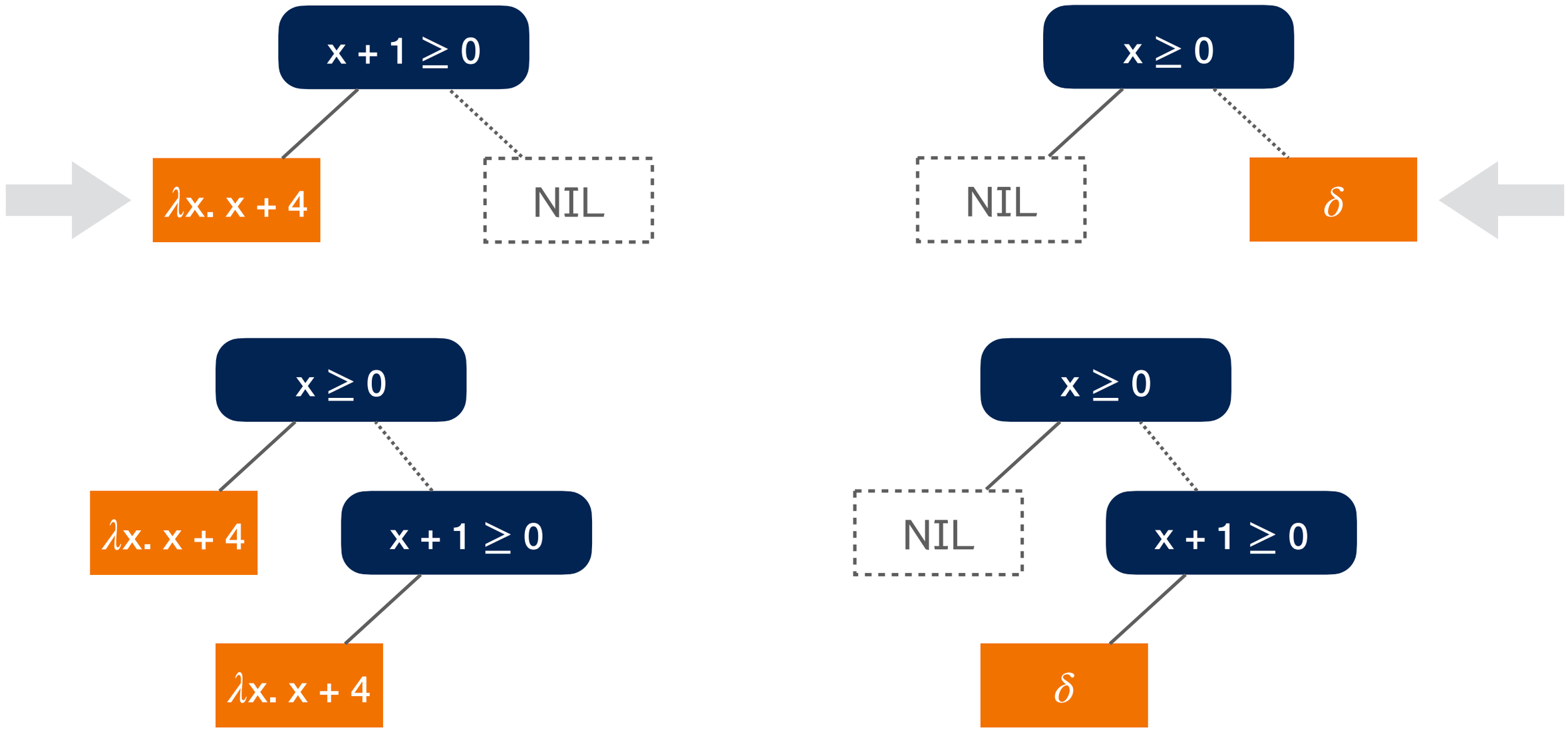
Tree Unification (continue)

Example

Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

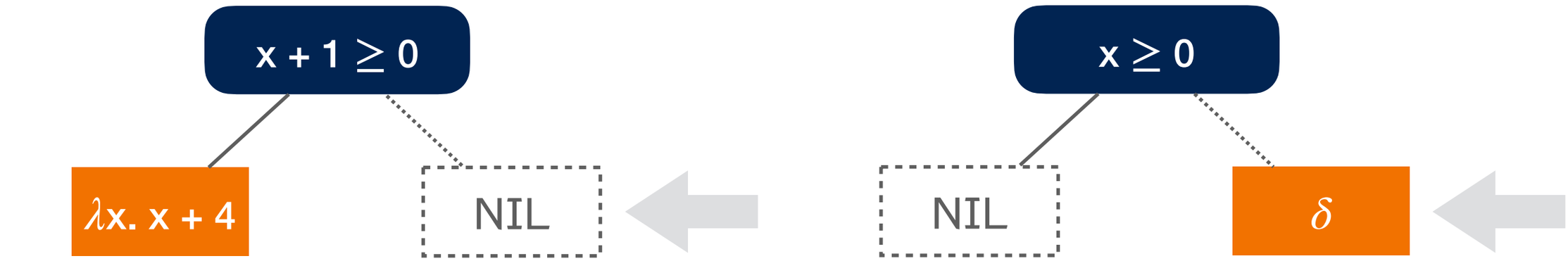
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

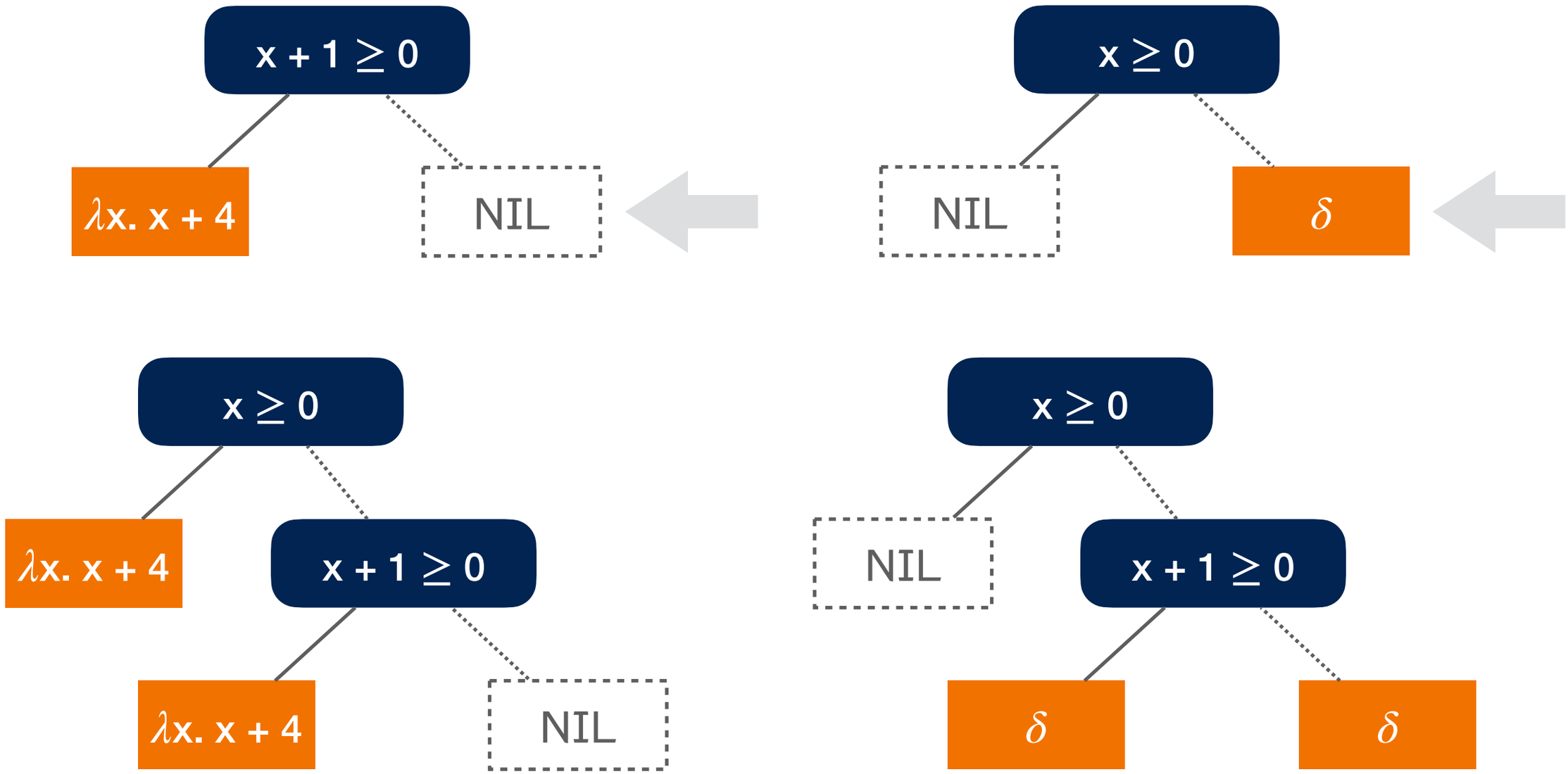
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

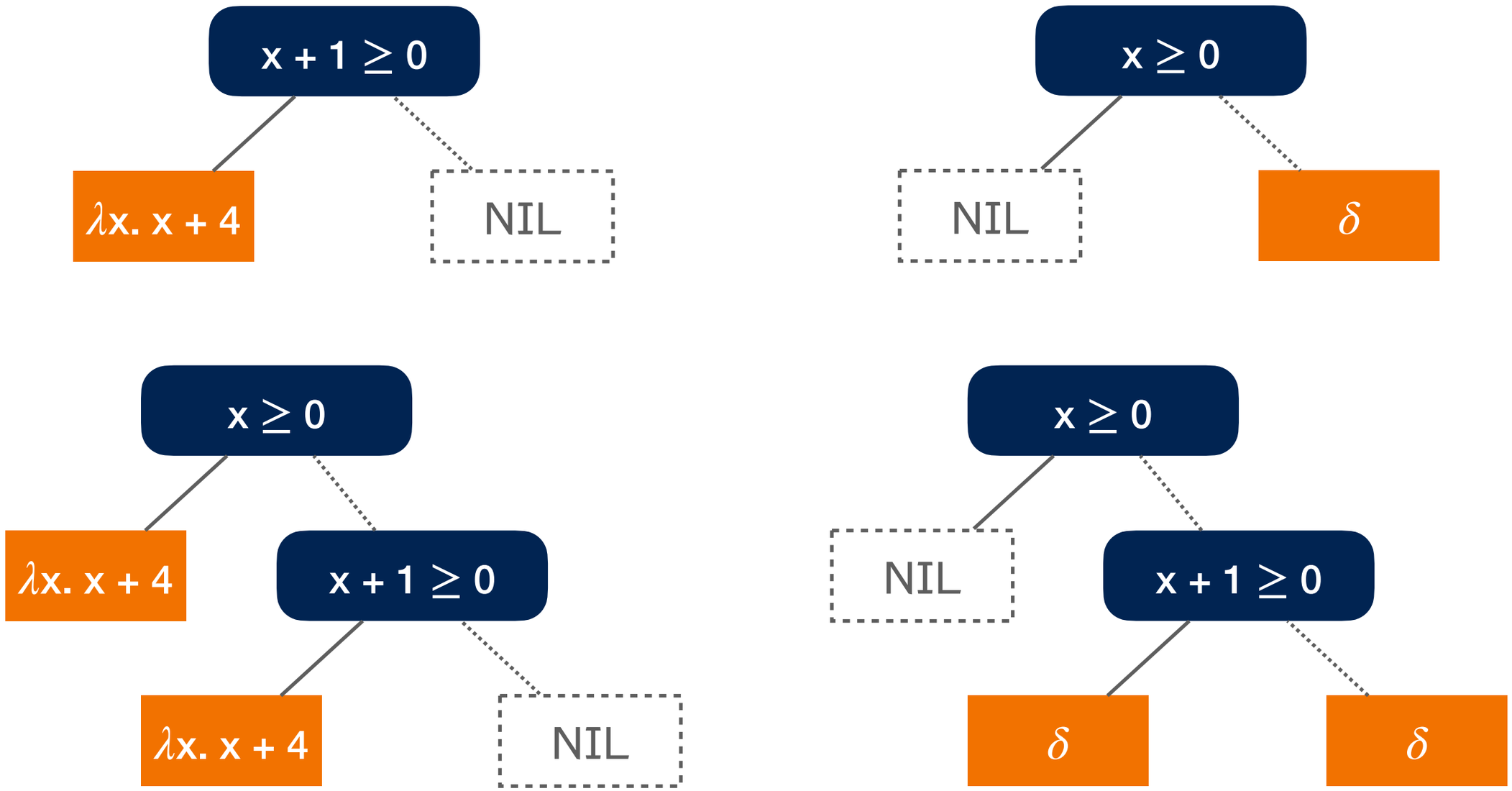
Example



Piecewise-Defined Ranking Functions Abstract Domain

Tree Unification (continue)

Example



Piecewise-Defined Ranking Functions Abstract Domain

Order

1. Perform **tree unification**

2. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C

3. Compare the leaf nodes using the **approximation order** $\preccurlyeq_F [\alpha_C(C)]$ or the **computational order** $\sqsubseteq_F [\alpha_C(C)]$

The concretization function γ_A is monotonic with respect to $\preccurlyeq_A$:

Lemma

$$\forall t_1, t_2 \in \mathscr{A} : t_1 \preccurlyeq_A t_2 \Rightarrow \gamma_A(t_1) \preccurlyeq \gamma_A(t_2)$$

Piecewise-Defined Ranking Functions Abstract Domain

Join

1. Perform **tree unification**

2. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C

3. $\text{NIL} \curlyvee_A t \stackrel{\text{def}}{=} t$
   $t \curlyvee_A \text{NIL} \stackrel{\text{def}}{=} t$

4. Join the leaf nodes using the **approximation join** $\curlyvee_F [\alpha_C(C)]$ or the **computational join** $\sqcup_F [\alpha_C(C)]$

Piecewise-Defined Ranking Functions Abstract Domain

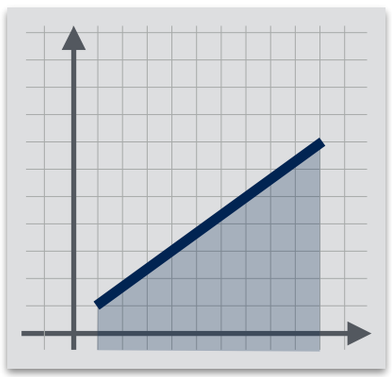
Join (continue)

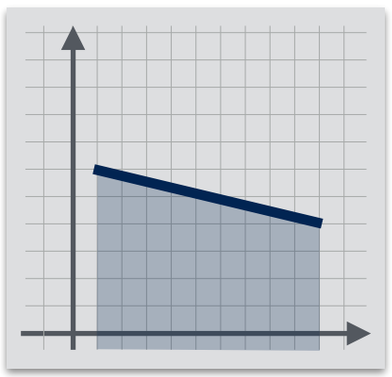
- **approximation join** $\curlyvee_F [D]$, where $D \in \mathscr{D}$:
  - between <u>defined</u> leaf nodes:

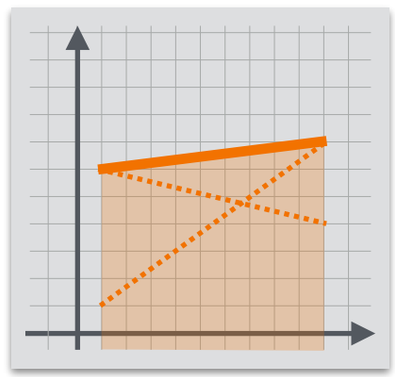
$$f_1 \curlyvee_F [D]\ f_2 \stackrel{\text{def}}{=} \begin{cases} f & f \in \mathscr{F} \backslash \{\ \bot_F, \top_F\ \} \\ \top_F & \text{otherwise} \end{cases}$$

$$\text{where } f \stackrel{\text{def}}{=} \lambda\rho \in \gamma_D(D) : \max(f_1(\ldots, \rho(X_i), \ldots), f_2(\ldots, \rho(X_i), \ldots))$$

Example:







Piecewise-Defined Ranking Functions Abstract Domain

Join (continue)

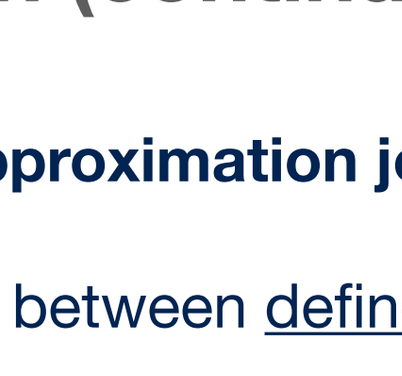
- **approximation join** $\curlyvee_F [D]$, where L

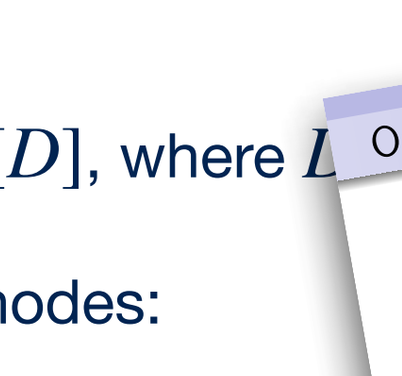
  - between <u>defined</u> leaf nodes:

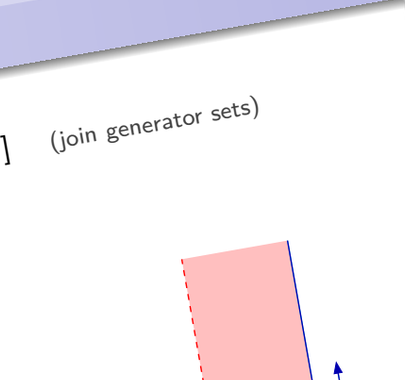
$$f_1 \curlyvee_F [D] f_2 \stackrel{\text{def}}{=} \begin{cases} f & f \in \mathscr{F} \backslash \{ \\ \top_F & \text{otherwise} \end{cases}$$

$$\text{where } f \stackrel{\text{def}}{=} \lambda\rho \in \gamma_D(D) : \max(f_1(\ldots$$

Example:







Piecewise-Defined Ranking Functions Abstract Domain

Join (continue)

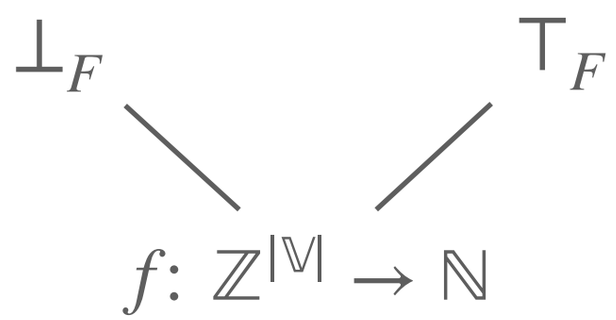
- **approximation join** $\curlyvee_F [D]$, where $D \in \mathscr{D}$:
  - between <u>defined</u> leaf nodes:

$$f_1 \curlyvee_F [D]\ f_2 \overset{\text{def}}{=} \begin{cases} f & f \in \mathscr{F} \backslash \{\ \bot_F, \top_F\ \} \\ \top_F & \text{otherwise} \end{cases}$$

$$\text{where } f \overset{\text{def}}{=} \lambda\rho \in \gamma_D(D) : \max(f_1(\ldots, \rho(X_i), \ldots), f_2(\ldots, \rho(X_i), \ldots))$$

  - otherwise (i.e., when one or both leaf nodes are <u>undefined</u>):

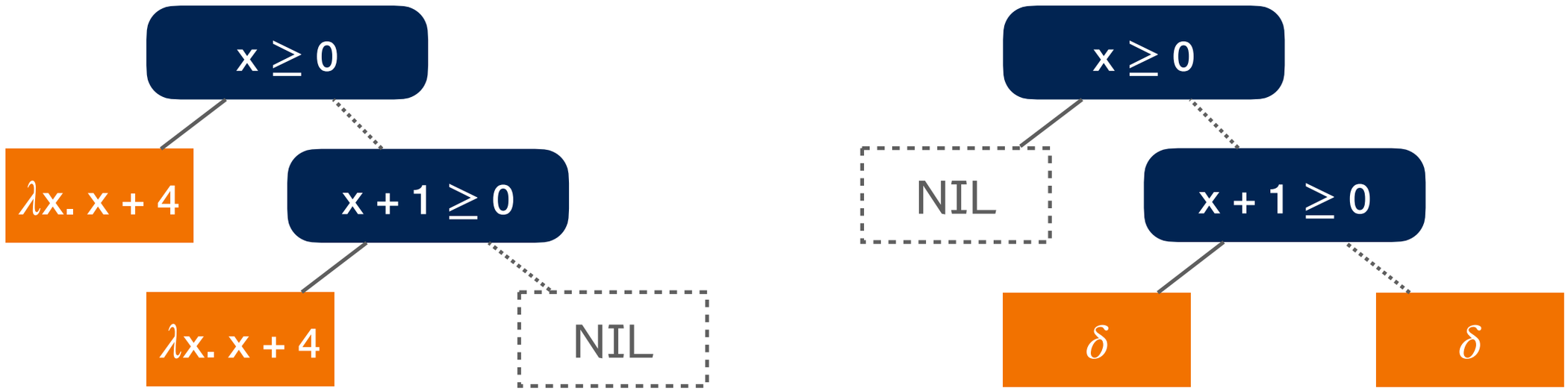
$$\begin{aligned} \bot_F \curlyvee_F [D]\ f &\overset{\text{def}}{=} \bot_F & f \in \mathscr{F} \backslash \{\ \top_F\ \} \\ f \curlyvee_F [D]\ \bot_F &\overset{\text{def}}{=} \bot_F & f \in \mathscr{F} \backslash \{\ \top_F\ \} \\ \top_F \curlyvee_F [D]\ f &\overset{\text{def}}{=} \top_F & f \in \mathscr{F} \backslash \{\ \bot_F\ \} \\ f \curlyvee_F [D]\ \top_F &\overset{\text{def}}{=} \top_F & f \in \mathscr{F} \backslash \{\ \bot_F\ \} \end{aligned}$$

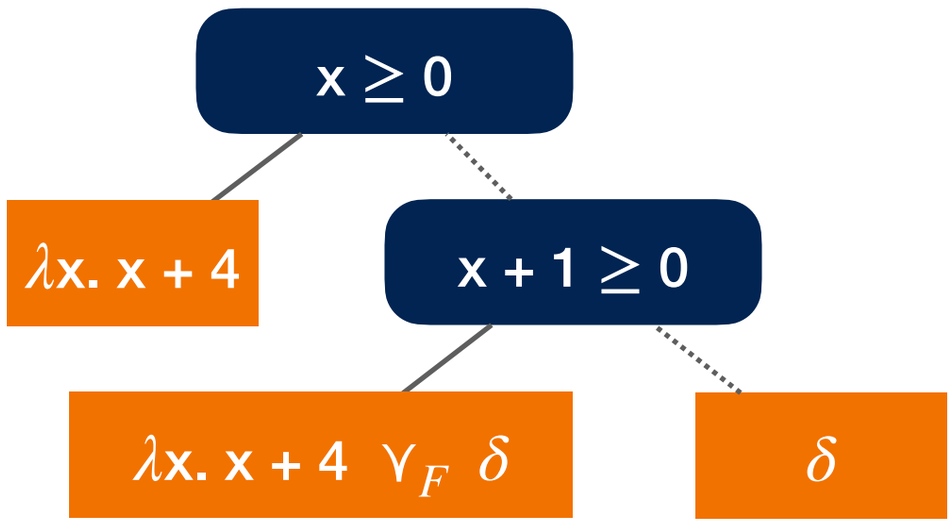


Piecewise-Defined Ranking Functions Abstract Domain

Join (continue)

Example





Piecewise-Defined Ranking Functions Abstract Domain

Join (continue)

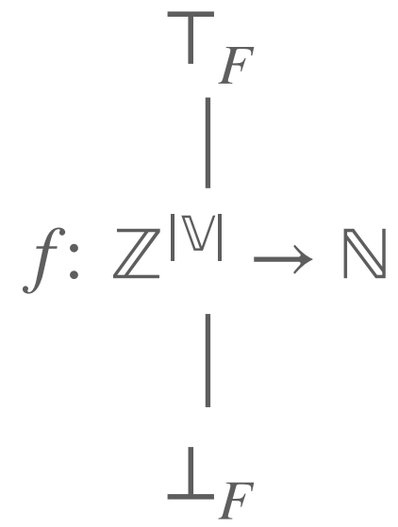
- **computational join** $\sqcup_F [D]$, where $D \in \mathscr{D}$:
  - between <u>defined</u> leaf nodes:

$$f_1 \curlyvee_F [D]\ f_2 \overset{\text{def}}{=} \begin{cases} f & f \in \mathscr{F} \setminus \{\ \bot_F, \top_F\ \} \\ \top_F & \text{otherwise} \end{cases}$$

$$\text{where } f \overset{\text{def}}{=} \lambda\rho \in \gamma_D(D) : \max(f_1(\ldots, \rho(X_i), \ldots), f_2(\ldots, \rho(X_i), \ldots))$$

- otherwise (i.e., when one or both leaf nodes are <u>undefined</u>):

$$\begin{aligned} \bot_F \sqcup_F [D]\ f &\overset{\text{def}}{=} f && f \in \mathscr{F} \\ f \sqcup_F [D]\ \bot_F &\overset{\text{def}}{=} f && f \in \mathscr{F} \\ \top_F \sqcup_F [D]\ f &\overset{\text{def}}{=} \top_F && f \in \mathscr{F} \\ f \sqcup_F [D]\ \top_F &\overset{\text{def}}{=} \top_F && f \in \mathscr{F} \end{aligned}$$



Piecewise-Defined Ranking Functions Abstract Domain

Meet

1. Perform **tree unification**

2. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C

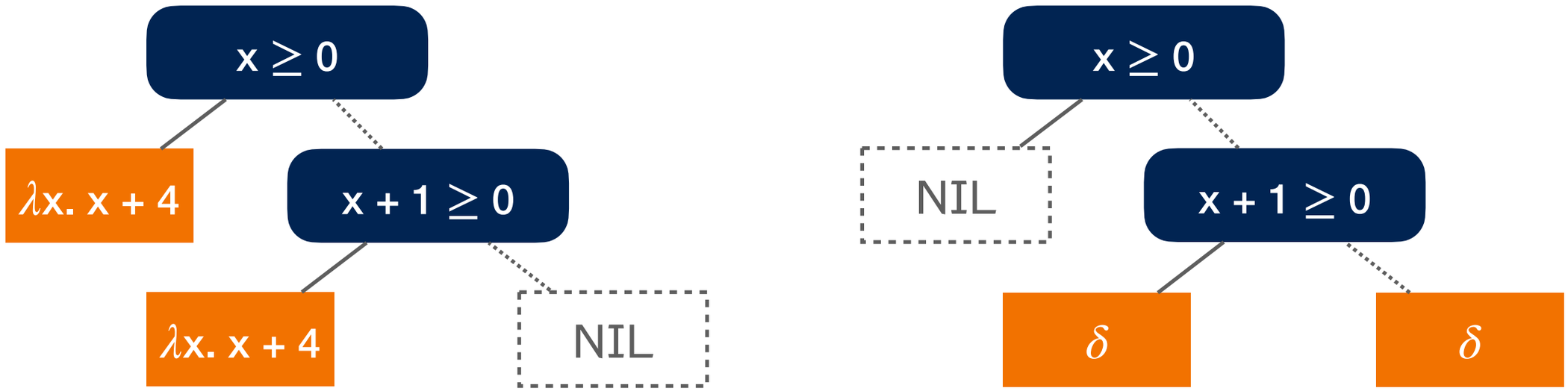
3. $\text{NIL} \curlyvee_A t \stackrel{\text{def}}{=} \text{NIL}$
   $t \curlyvee_A \text{NIL} \stackrel{\text{def}}{=} \text{NIL}$

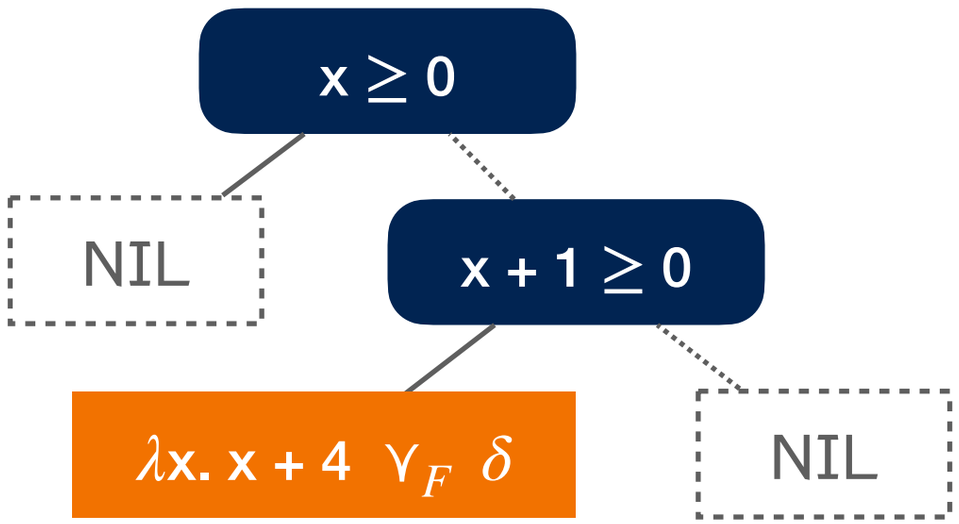
4. Join the leaf nodes using the **approximation join** $\curlyvee_F [\alpha_C(C)]$

Piecewise-Defined Ranking Functions Abstract Domain

Meet (continue)

Example





Piecewise-Defined Ranking Functions Abstract Domain

Widening

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

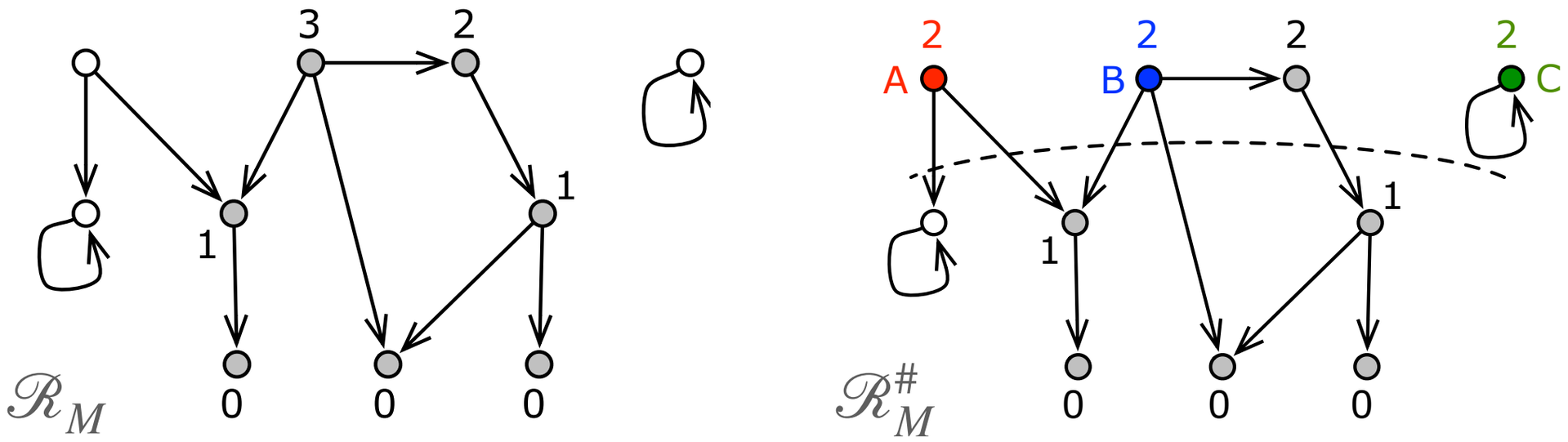
The prediction can (temporarily) be wrong!, i.e.,

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

and/or

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

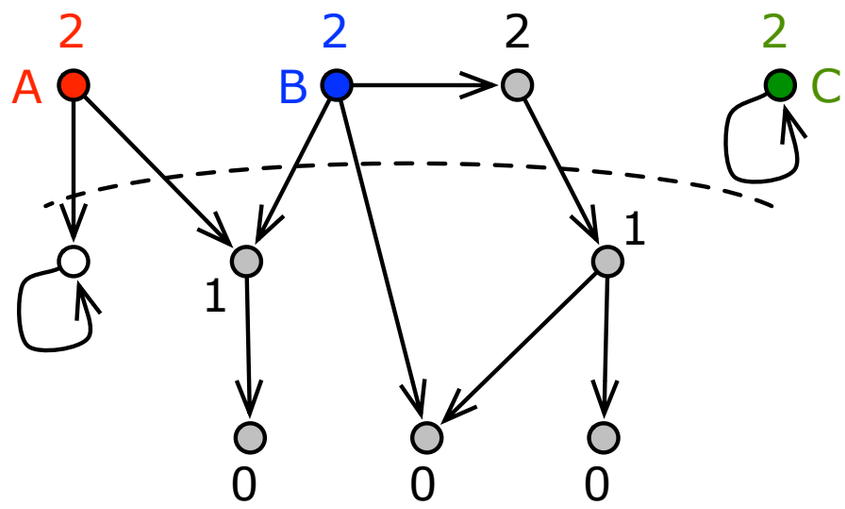
Example



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**



Piecewise-Defined Ranking Functions Abstract Domain

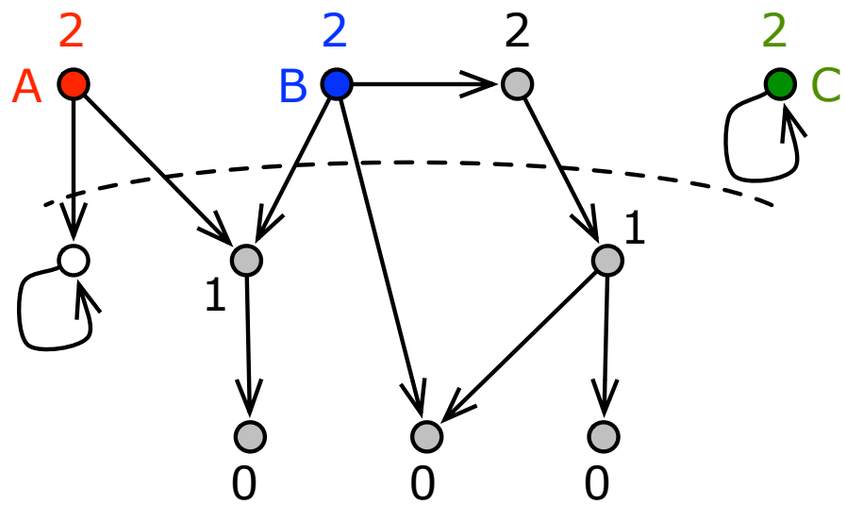
Widening (continue)

Check for Case A

Lemma

Let $\mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell)) \neq \emptyset$. Then, in case A, we have $\mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n+1}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell)) \subset \mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell))$.

(see proof in [Urban15])



Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

Check for Case A

Lemma

Let $\mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell)) \neq \emptyset$. Then, in case A, we have $\mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n+1}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell)) \subset \mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell))$.

(see proof in [Urban15])

1. Perform **tree unification**
2. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C
3. f $\quad \bot_F$ f $\quad \top_F$

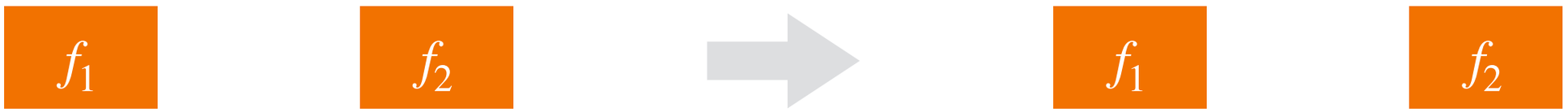
Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue) Domain Widening

Goal: **limit the size** of the decision trees

Left unification: variant of tree unification that forces the structure of t_1 on t_2

- Base case:

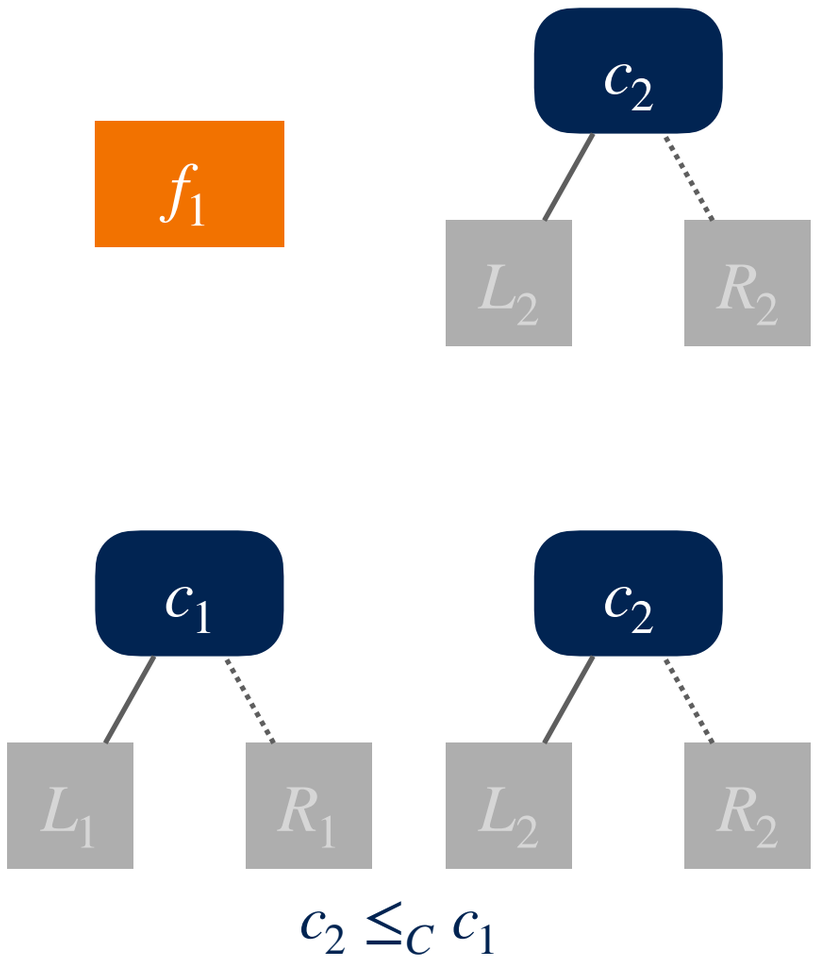


Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

Domain Widening

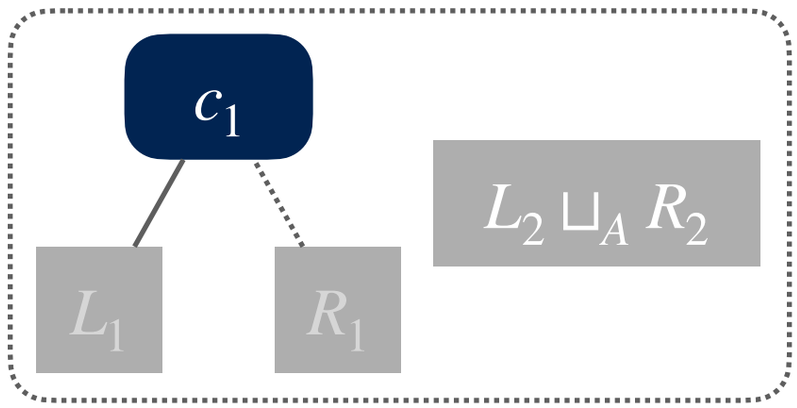
- Case ①



① c_2 is redundant

① $\neg c_2$ is redundant

① c_2 is removed from t_2

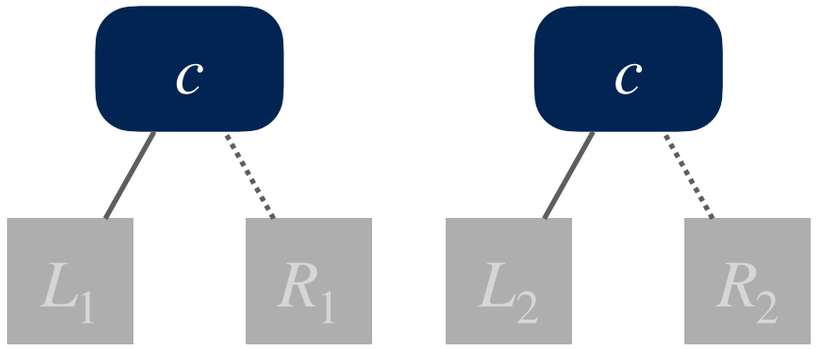


Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

Domain Widening

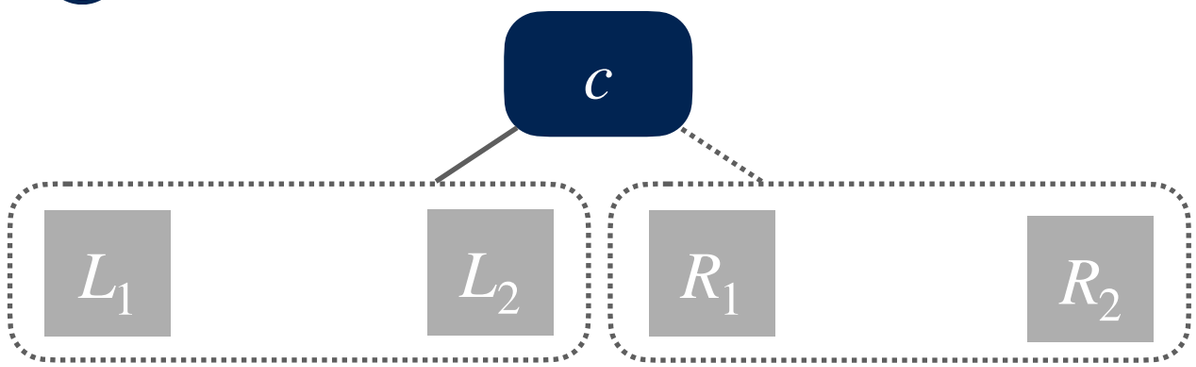
- Case ② (as for tree unification)
- Case ③



1a c is redundant

1b $\neg c$ is redundant

1c c is kept in t_1 and t_2



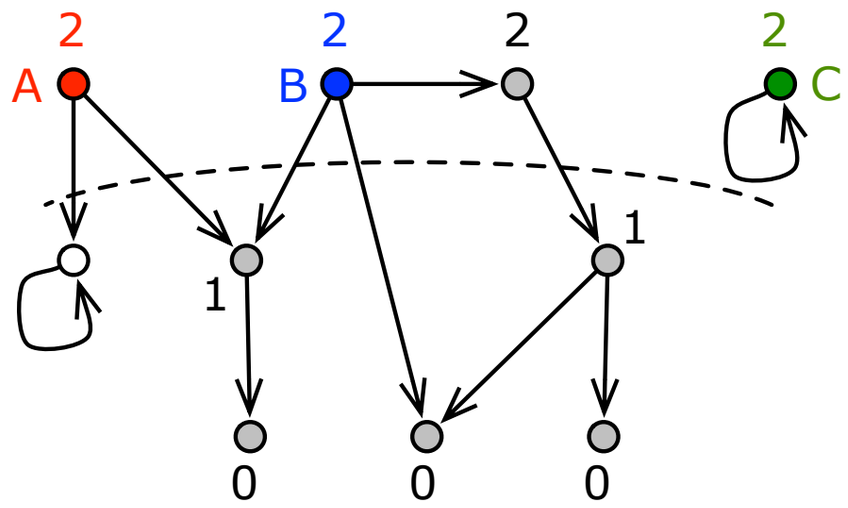
Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

Check for Case B or C

Lemma

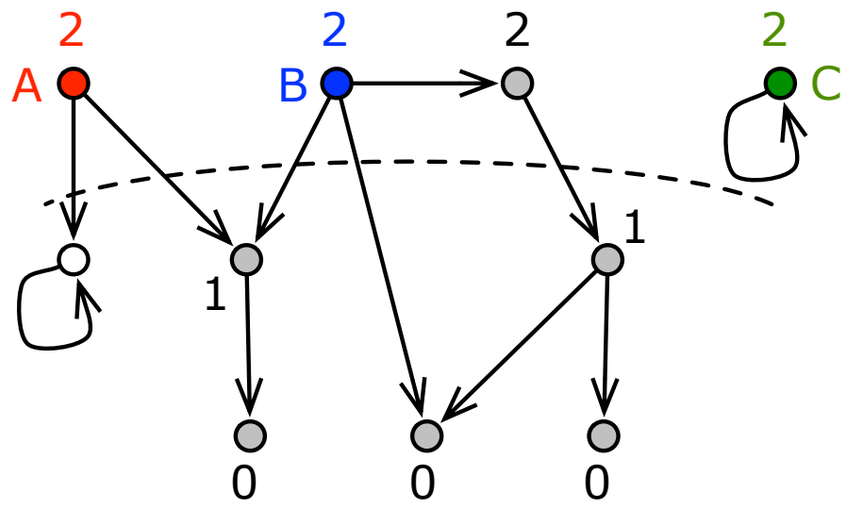
Let $\gamma_A(\mathscr{R}_M^{\#n}(\ell))(\overline{\rho}) < \mathscr{R}_M(\ell)(\overline{\rho})$ for some $\overline{\rho} \in \text{dom}(\mathscr{R}_M(\ell)) \cap \text{dom}(\gamma_A(\mathscr{R}_M^{\#n})(\ell))$ (case B). Then, there exists $\rho \in \text{dom}(\gamma_A(\mathscr{R}_M^{\#n+1}(\ell))) \cap \text{dom}(\mathscr{R}_M^{\#n}(\ell))$ such that $\gamma_A(\mathscr{R}_M^{\#n}(\ell))(\rho) < \gamma_A(\mathscr{R}_M^{\#n+1}(\ell))(\rho)$.



Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

Check for Case B or C



Lemma

Let $\mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell)) \neq \emptyset$. Then, for all $\rho \in \mathrm{dom}(\gamma_A(\mathscr{R}_M^{\#n}(\ell)))\backslash\mathrm{dom}(\mathscr{R}_M(\ell))$ in case C, we have $\gamma_A(\mathscr{R}_M^{\#n}(\ell))(\rho) < \gamma_A(\mathscr{R}_M^{\#n+1}(\ell))(\rho)$.

(see proof in [Urban15])

Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue)

Check for Case B or C

1. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C

2. f_1 f_2 → f $\top_F$

$f_1 \in \mathscr{F} \setminus \{ \bot_F , \top_F \} \wedge f_2 \not\preccurlyeq_F [\alpha_c(C)] f_1$

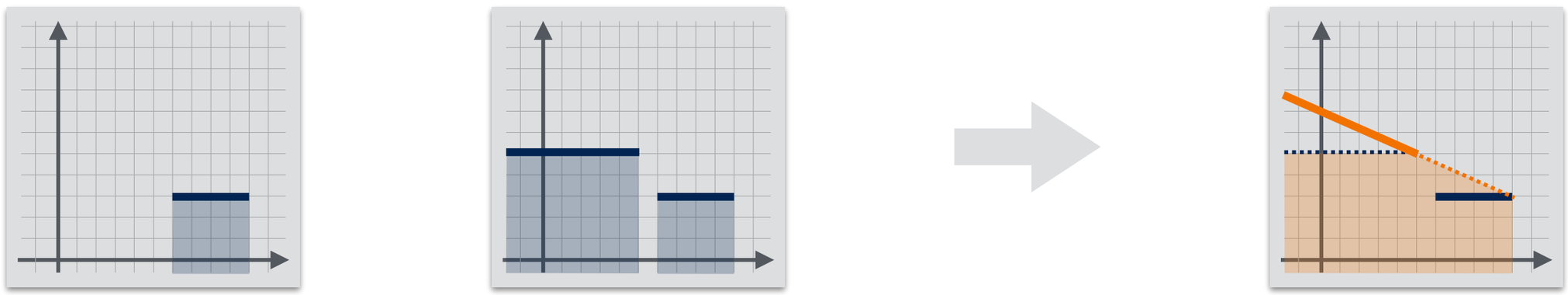
Piecewise-Defined Ranking Functions Abstract Domain

Widening (continue) — Value Widening

1. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C

2. Widen each (defined) leaf node f with respect to each of their adjacent (defined) leaf node $\bar{f}$ using the **extrapolation operator** $\blacktriangledown_F [\alpha_C(\overline{C}), \alpha_C(C)]$, where $\overline{C}$ is the set of constraints along the path to $\bar{f}$

Example:

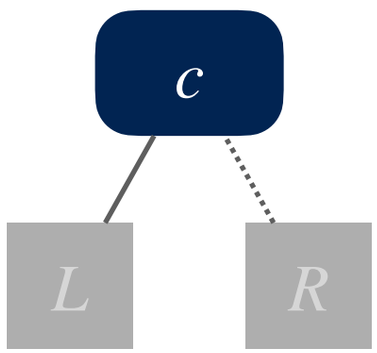


Piecewise-Defined Ranking Functions Abstract Domain

Tree Pruning

Goal: **add** a set J of **linear constraints** to the decision tree

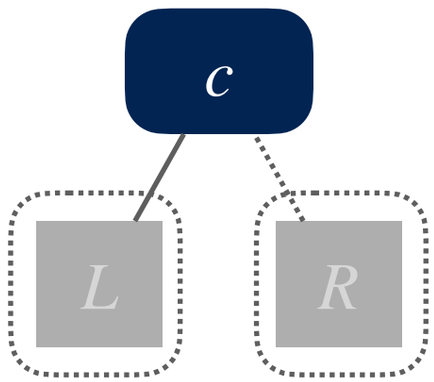
- Base case ($J = \emptyset$)



(a) c is redundant

(b) $\neg c_2$ is redundant

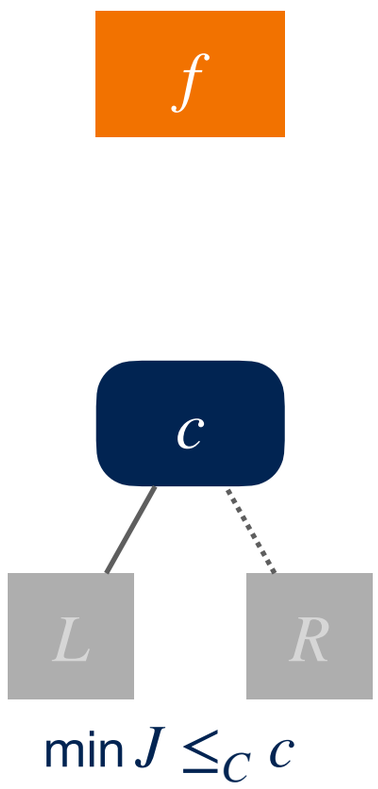
(c) c is kept in t

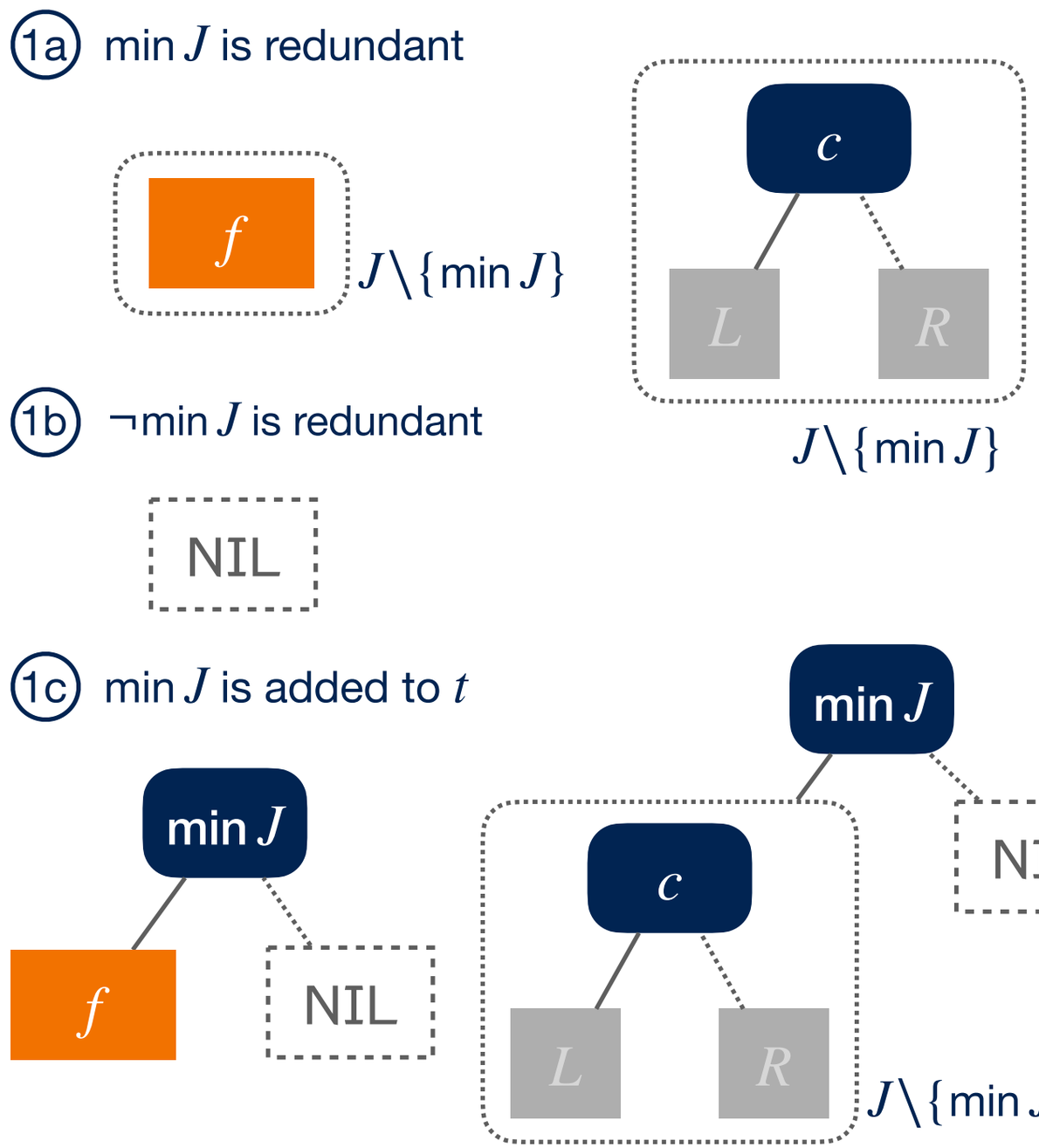


Piecewise-Defined Ranking Functions Abstract Domain

Tree Pruning (continue)

- Case ①

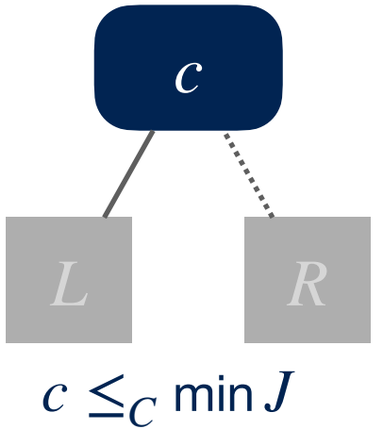




Piecewise-Defined Ranking Functions Abstract Domain

Tree Pruning (continue)

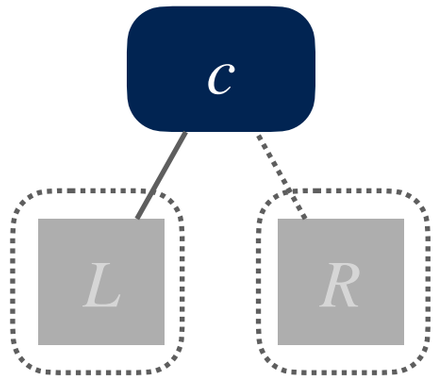
- Case ②



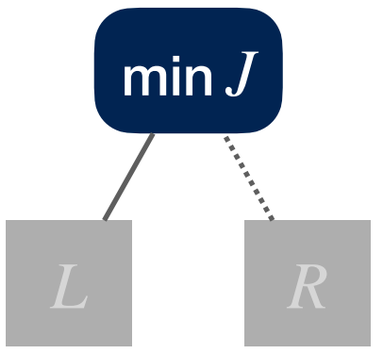
2a c is redundant

2b $\neg c_2$ is redundant

2c c is kept in t



- Case ③



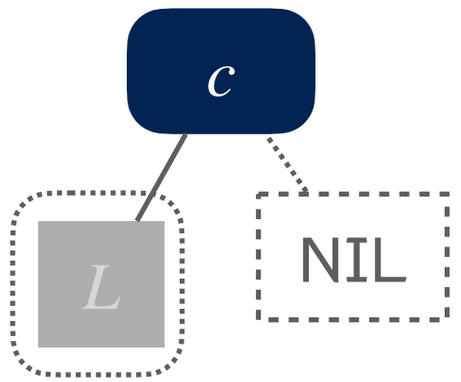
3a $\min J$ is redundant

$J \setminus \{\min J\}$

3b $\neg \min J$ is redundant

NIL

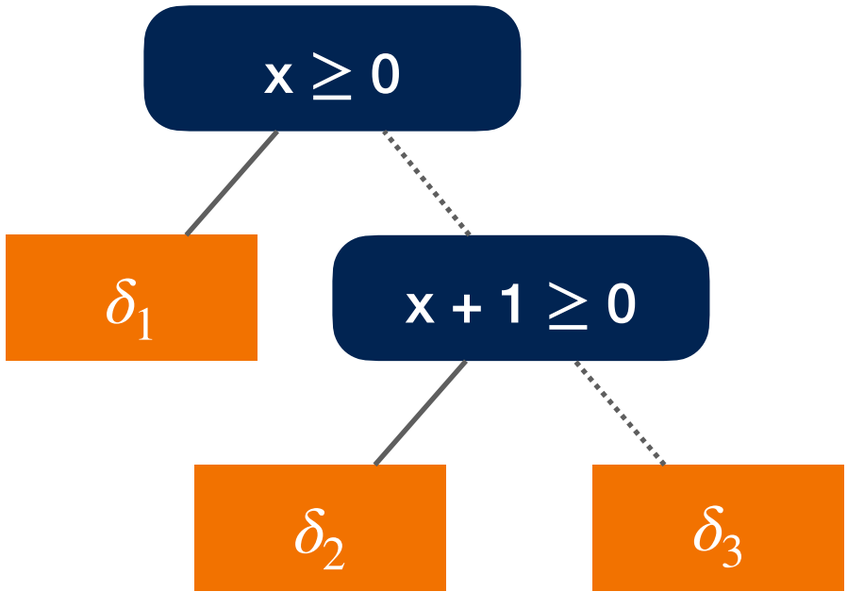
3c $\min J$ is kept in t



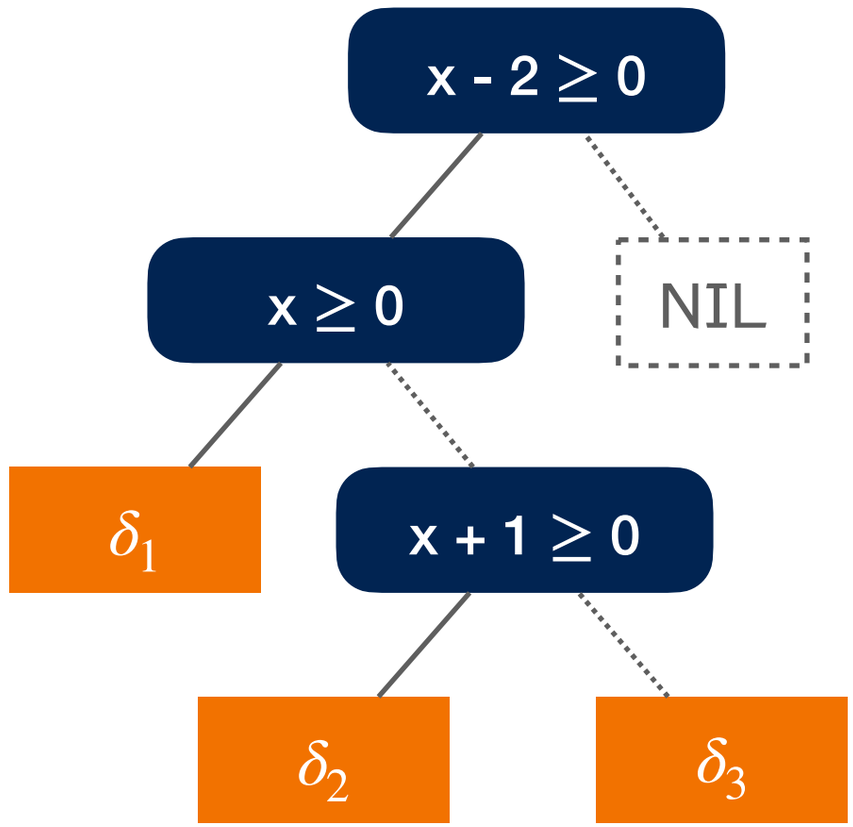
Piecewise-Defined Ranking Functions Abstract Domain

Tree Pruning (continue)

Example



$$J \stackrel{\text{def}}{=} \{x - 2 \geq 0\}$$



Piecewise-Defined Ranking Functions Abstract Domain

Assignments

$\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]$

- Base case (f)

  Apply $\overleftarrow{\text{ASSIGN}}_F[\![X \leftarrow e]\!][\alpha_C(C)]$ on the defined leaf nodes

$$\overleftarrow{\text{ASSIGN}}_F[\![X \leftarrow e]\!][D](f) \stackrel{\text{def}}{=} \begin{cases} \bar{f} & \bar{f} \in \mathscr{F} \setminus \{ \bot_F , \top_F \} \\ \top_F & \text{otherwise} \end{cases} \qquad f \in \mathscr{F} \setminus \{ \bot_F , \top_F \}$$

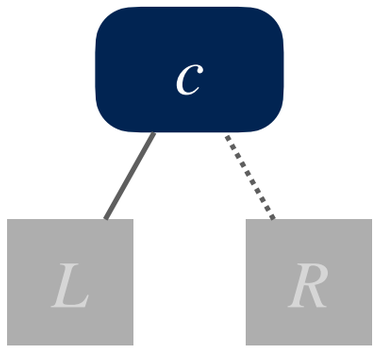
where $\bar{f}(\ldots, X_i, X, \ldots) \stackrel{\text{def}}{=} \max\{f(\ldots, \rho(X_i), v, \ldots) + 1 \mid \rho \in \gamma_D(R) \wedge v \in E[\![e]\!]\rho\}$
and $R \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_D[\![X \leftarrow e]\!]D$

Example:
$\overleftarrow{\text{ASSIGN}}_F[\![x \leftarrow x + [1,2]]\!][\ \top_D\ ](\lambda x\,.\,x + 1) = \lambda x\,.\,x + 4$
(since $f(x + [1,2]) + 1 = x + [1,2] + 1 + 1 = x + [3,4]$ and $\max(3,4) = 4$

Piecewise-Defined Ranking Functions Abstract Domain

Assignments

$\overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]$

- 

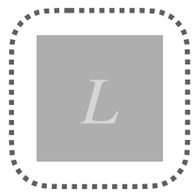
Convert $\overleftarrow{\text{ASSIGN}}_D[\![X \leftarrow e]\!](\alpha_C(\{c\}))$ **and**
$\overleftarrow{\text{ASSIGN}}_D[\![X \leftarrow e]\!](\alpha_C(\{\neg c\}))$
into sets I **and** J **of linear constraints** *in canonical form*

case ① $I = J = \varnothing$

case ② $I = \varnothing \wedge \bot_C \in J$

case ③ $\bot_C \in I \wedge J = \varnothing$

case ④

1. perform **tree pruning** on  and 
2. join the results with Υ_A

Piecewise-Defined Ranking Functions Abstract Domain

Tests

$\mathtt{FILTER}_A[\![e]\!]$

1. Recursively descend the tree and apply STEP_F on the defined leaf nodes to account for one more execution step needed before termination:

$$\mathsf{STEP}_F(f) \stackrel{\text{def}}{=} \lambda X_1, \ldots, X_k \,.\, f(X_1, \ldots, X_k) + 1 \qquad f \in \mathscr{F} \backslash \{ \bot_F , \top_F \}$$

2. **Convert e into a set J of linear constraints** *in canonical form*

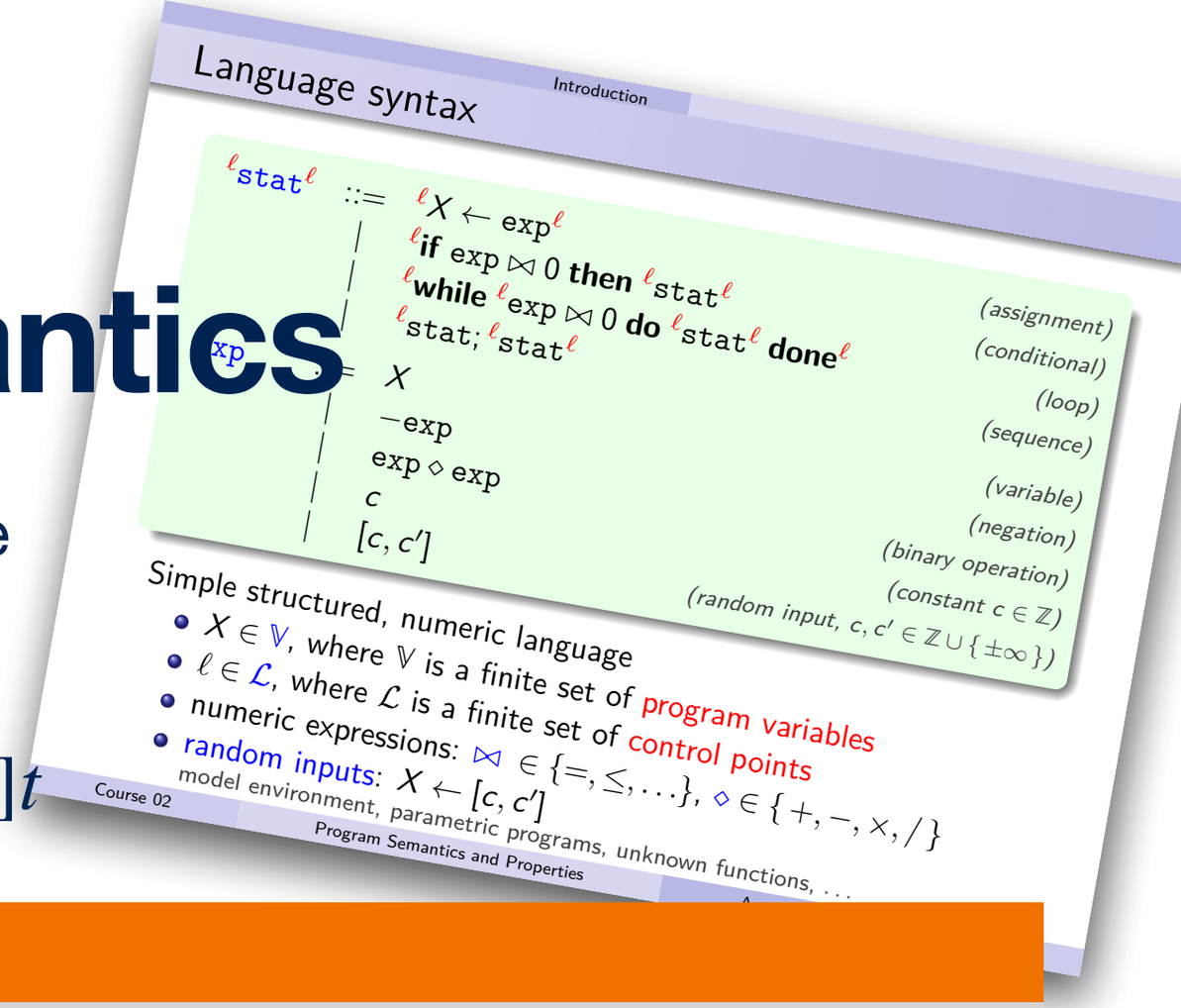
   Example: $\alpha_C(\mathtt{FILTER}_D[\![e]\!] \ \top_D \ )$
   where $\langle \mathscr{D}, \sqsubseteq_D \rangle$ is the underlying numerical domain

3. Perform **tree pruning** with J

Abstract Definite Termination Semantics

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

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

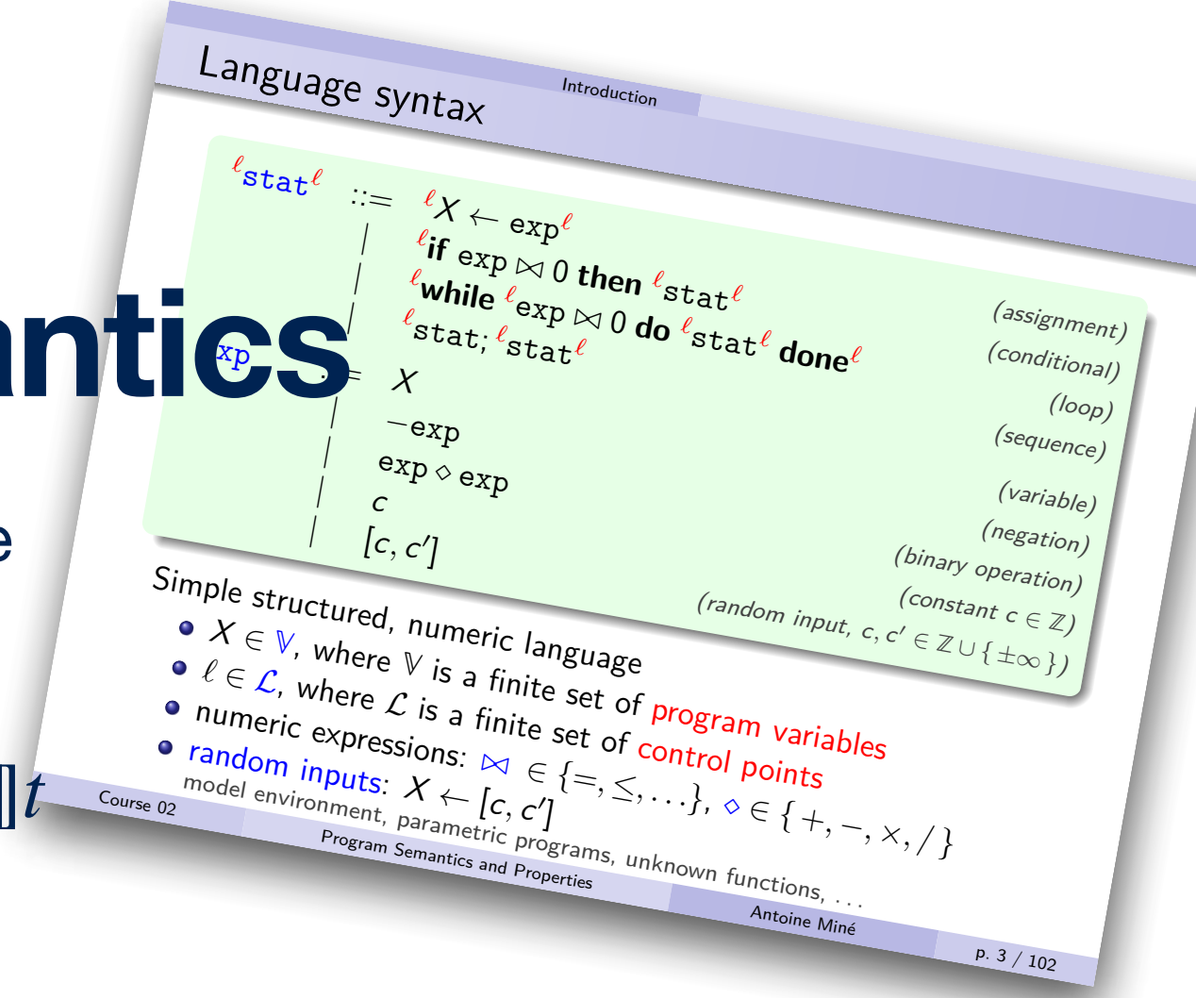


Lemma (Soundness)

$$\mathscr{R}_{M}[\![{}^{\ell}X \leftarrow e]\!]\gamma_{A}(t) \preccurlyeq \gamma_{A}(\mathscr{R}^{\#}_{M}[\![{}^{\ell}X \leftarrow e]\!]t)$$

(see proof in [Urban15])

Abstract Definite Termination Semantics



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

- $\mathscr{R}^\#_M[\![^\ell X \leftarrow e]\!]t \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_A[\![X \leftarrow e]\!]t$
- $\mathscr{R}^\#_M[\![\textbf{if } ^\ell e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=}$
  $\text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^\#_M[\![s]\!]t) \curlyvee_T \text{FILTER}_A[\![e \bowtie\!\!\!\!/\ 0]\!]t$

Lemma (Soundness)

$$\mathscr{R}_M[\![\textbf{if } ^\ell e \bowtie 0 \textbf{ then } s]\!]\gamma_A(t) \preccurlyeq \gamma_A(\mathscr{R}^\#_M[\![\textbf{if } ^\ell e \bowtie 0 \textbf{ then } s]\!]t)$$

(see proof in [Urban15])

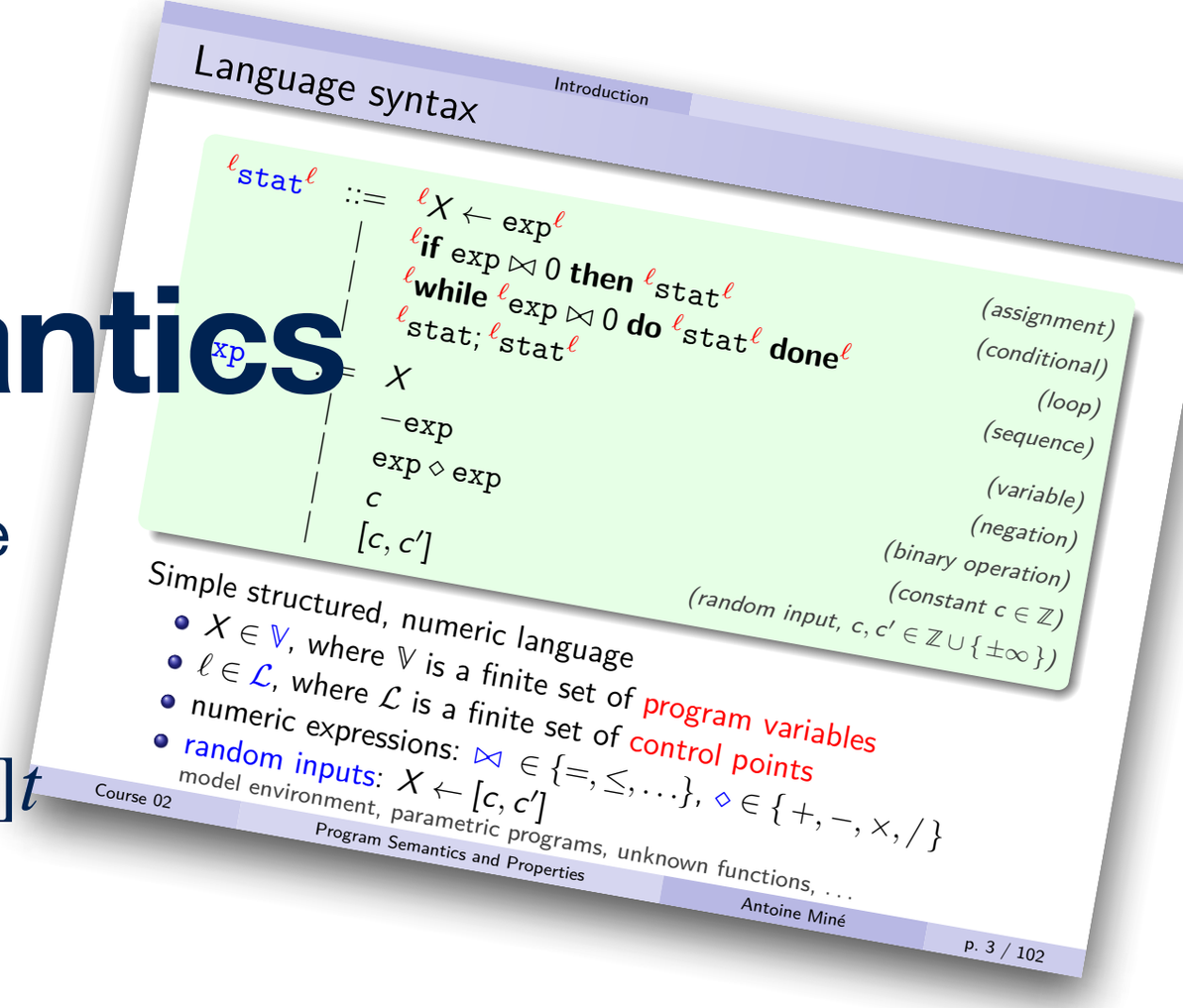
Abstract Definite Termination Semantics

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

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

- $\mathscr{R}^\#_M[\![\textbf{if}\ {}^\ell\ e \bowtie 0\ \textbf{then}\ s]\!]t \overset{\text{def}}{=}$
  $\text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^\#_M[\![s]\!]t) \curlyvee_T \text{FILTER}_A[\![e \bowtie 0]\!]t$

- $\mathscr{R}^\#_M[\![\textbf{while}\ {}^\ell\ e \bowtie 0\ \textbf{do}\ s\ \textbf{done}]\!]t \overset{\text{def}}{=} \text{lfp}^\# \overline{F}^\#_M$
  where $\overline{F}^\#_M(x) \overset{\text{def}}{=} \text{FILTER}_A[\![e \bowtie 0]\!](\mathscr{R}^\#_M[\![s]\!]x) \curlyvee_T \text{FILTER}_A[\![e \bowtie 0]\!](t)$



Lemma (Soundness)

$$\mathscr{R}_M[\![\textbf{while}\ {}^\ell\ e \bowtie 0\ \textbf{do}\ s\ \textbf{done}]\!]\gamma_A(t) \preccurlyeq \gamma_A(\mathscr{R}^\#_M[\![\textbf{while}\ {}^\ell\ e \bowtie 0\ \textbf{do}\ s\ \textbf{done}]\!]t)$$

(see proof in [Urban15])

Abstract Definite Termination Semantics

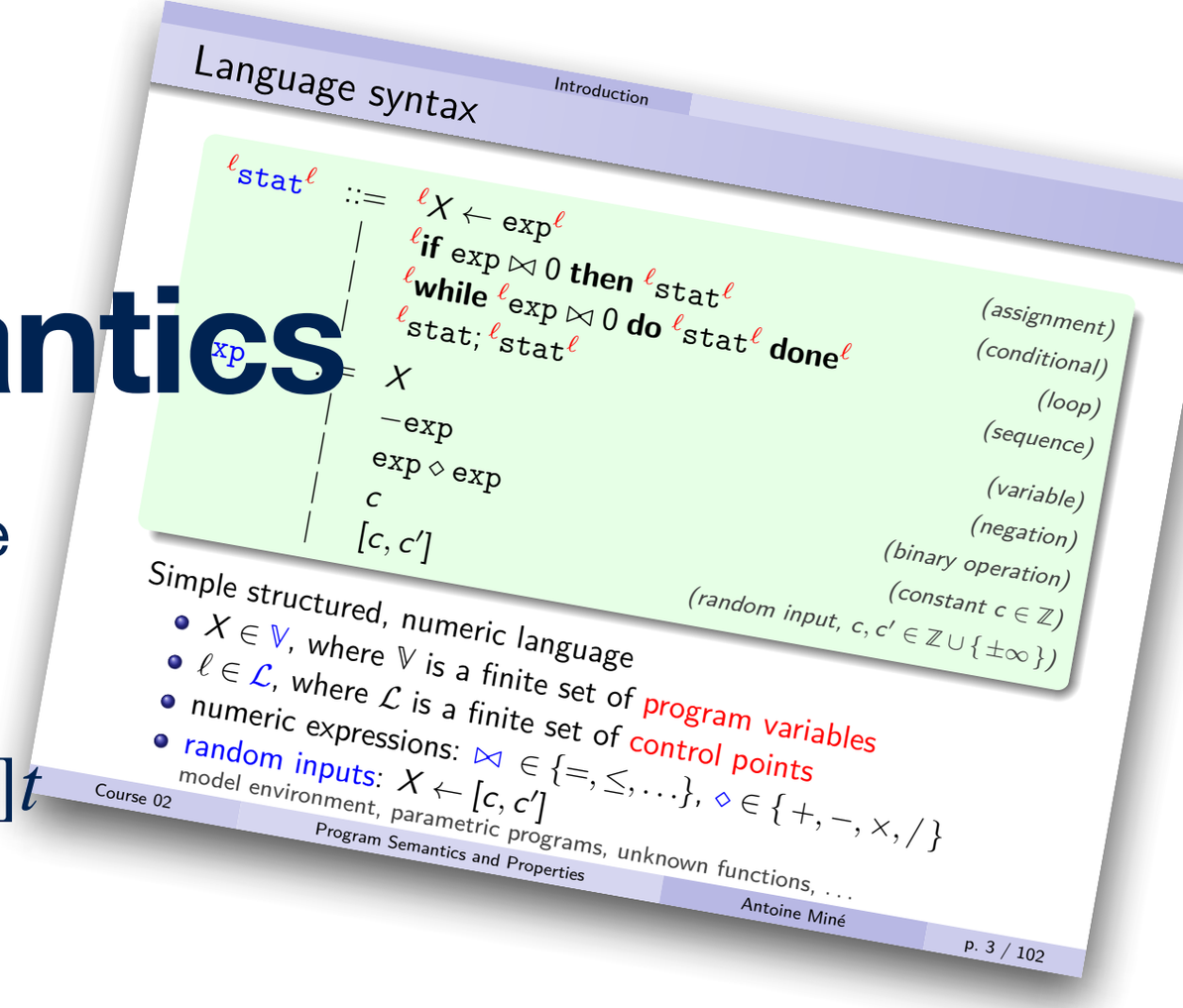
For each program instruction stat, we define a transformer $\mathscr{R}^{\#}_{M}[\![\text{stat}]\!] : \mathscr{A} \to \mathscr{A}$:

- $\mathscr{R}^{\#}_{M}[\![{}^{\ell} X \leftarrow e]\!]t \overset{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_{A}[\![X \leftarrow e]\!]t$

- $\mathscr{R}^{\#}_{M}[\![\textbf{if}\ {}^{\ell}\ e \bowtie 0\ \textbf{then}\ s]\!]t \overset{\text{def}}{=}$
  $\text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\#}_{M}[\![s]\!]t) \curlyvee_{T} \text{FILTER}_{A}[\![e \bowtie 0]\!]t$

- $\mathscr{R}^{\#}_{M}[\![\textbf{while}\ {}^{\ell}\ e \bowtie 0\ \textbf{do}\ s\ \textbf{done}]\!]t \overset{\text{def}}{=} \text{lfp}^{\#}\overline{F}^{\#}_{M}$
  where $\overline{F}^{\#}_{M}(x) \overset{\text{def}}{=} \text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\#}_{M}[\![s]\!]x) \curlyvee_{T} \text{FILTER}_{A}[\![e \bowtie 0]\!](t)$

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



Abstract Definite Termination Semantics

Definition

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

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

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

Theorem (Soundness)

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

Corollary (Soundness)

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

Abstract Definite Termination Semantics

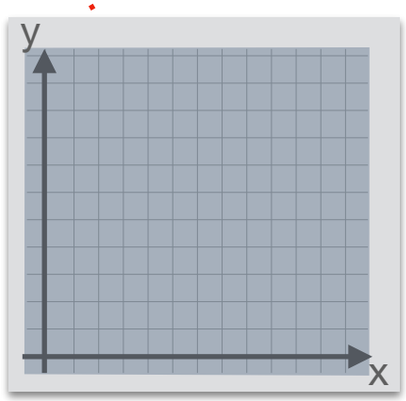
Example

$^{1}x \leftarrow [-\infty, +\infty]$
$^{2}y \leftarrow [-\infty, +\infty]$
while $^{3}(x > 0)$ **do**
$^{4}x \leftarrow x - y$
od^{5}

Abstract Definite Termination Semantics

Example

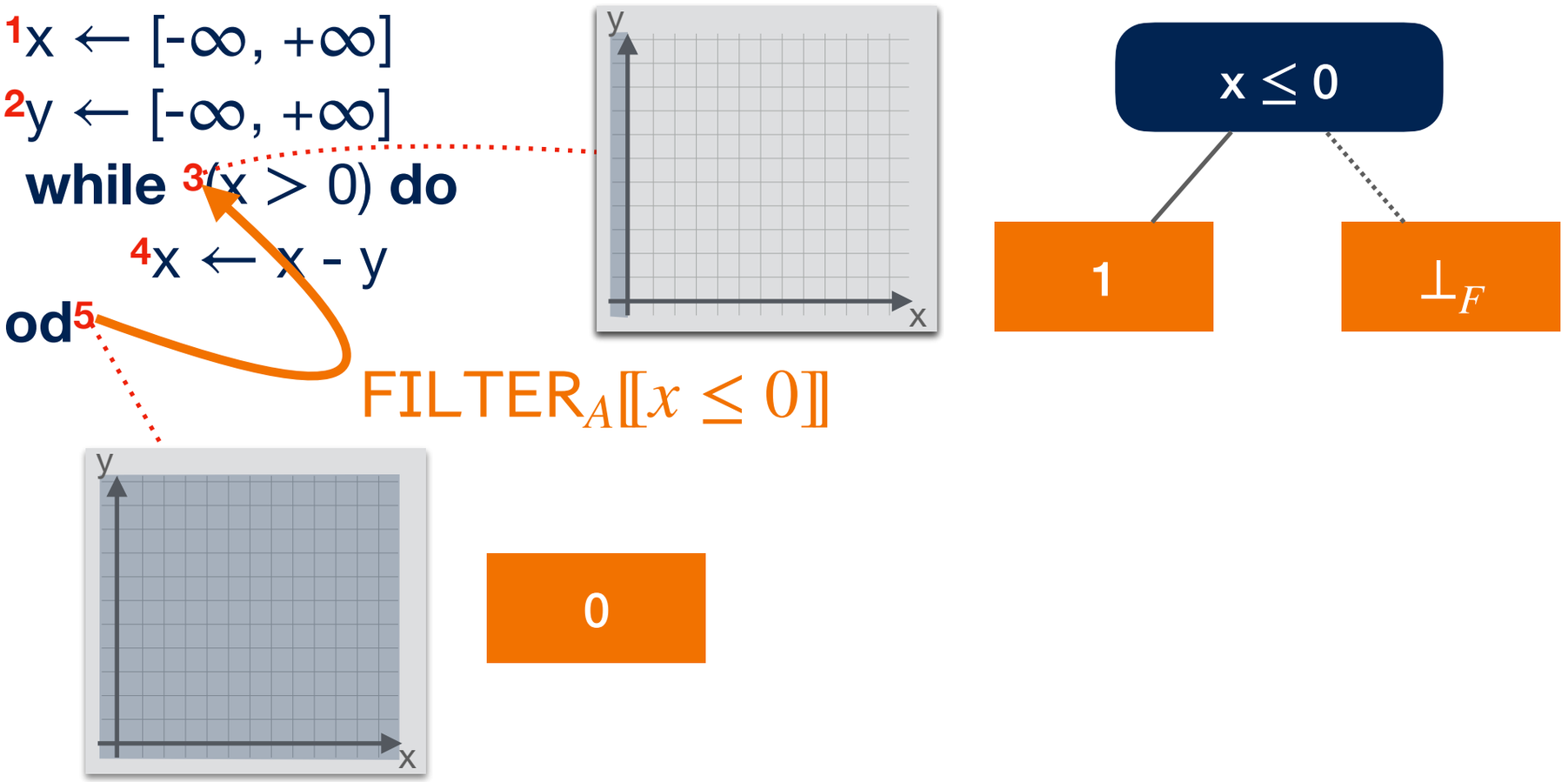
[1]x ← [-∞, +∞]
[2]y ← [-∞, +∞]
while [3](x > 0) **do**
    [4]x ← x - y
od[5]



0

Abstract Definite Termination Semantics

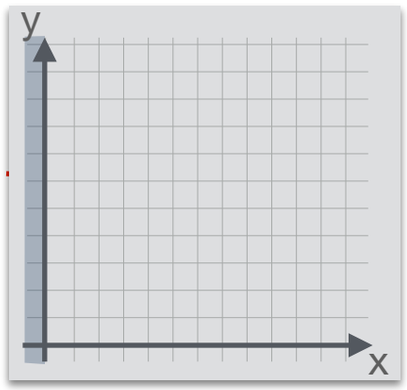
Example

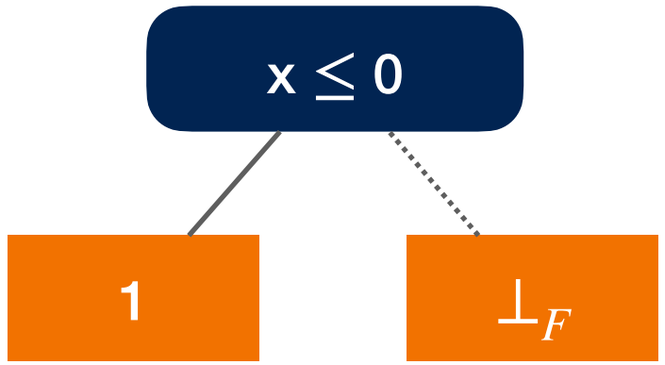


Abstract Definite Termination Semantics

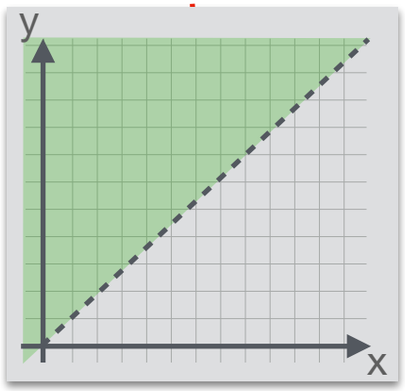
Example

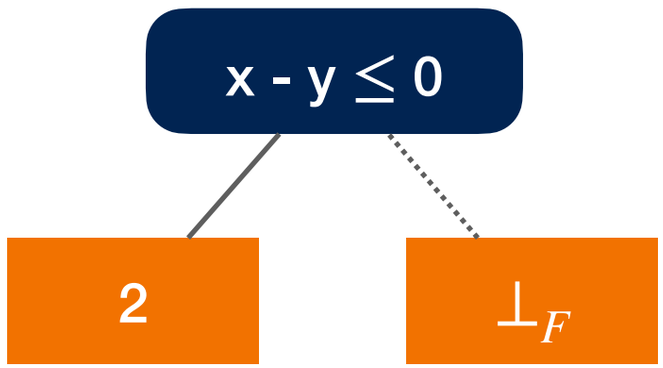
1x ← [-∞, +∞]
2y ← [-∞, +∞]
while 3(x > 0) **do**
4x ← x - y
od5





$\overleftarrow{\text{ASSIGN}}_A[\![x \leftarrow x - y]\!]$



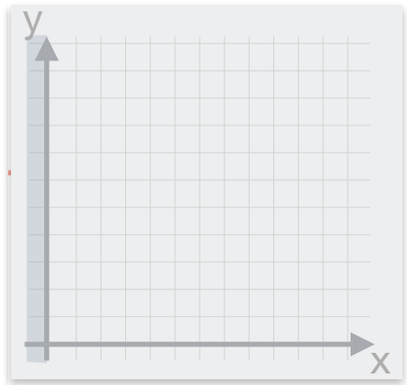


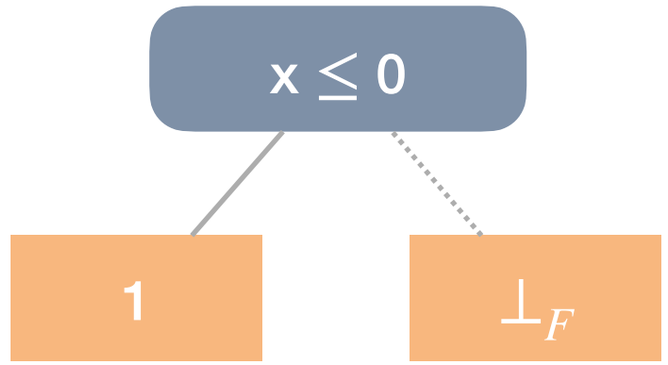
Abstract Definite Termination Semantics

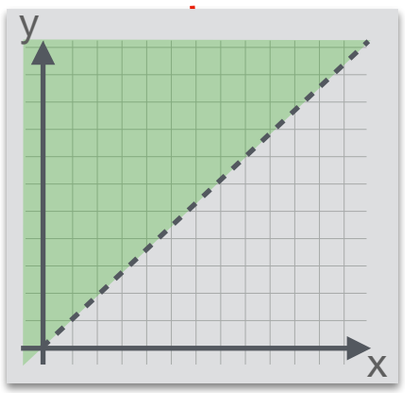
Example

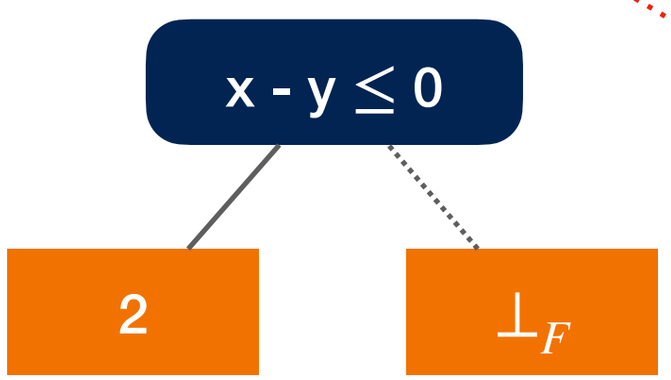
1 x ← [-∞, +∞]
2 y ← [-∞, +∞]
while 3 (x > 0) **do**
  4 x ← x - y
od 5

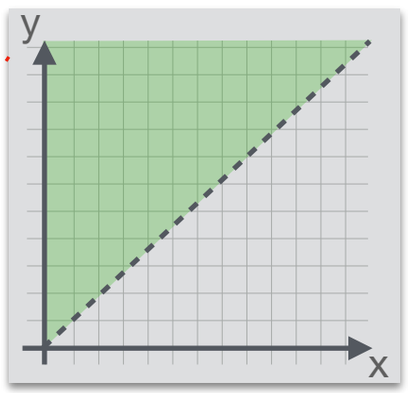
FILTER$_A[\![x > 0]\!]$

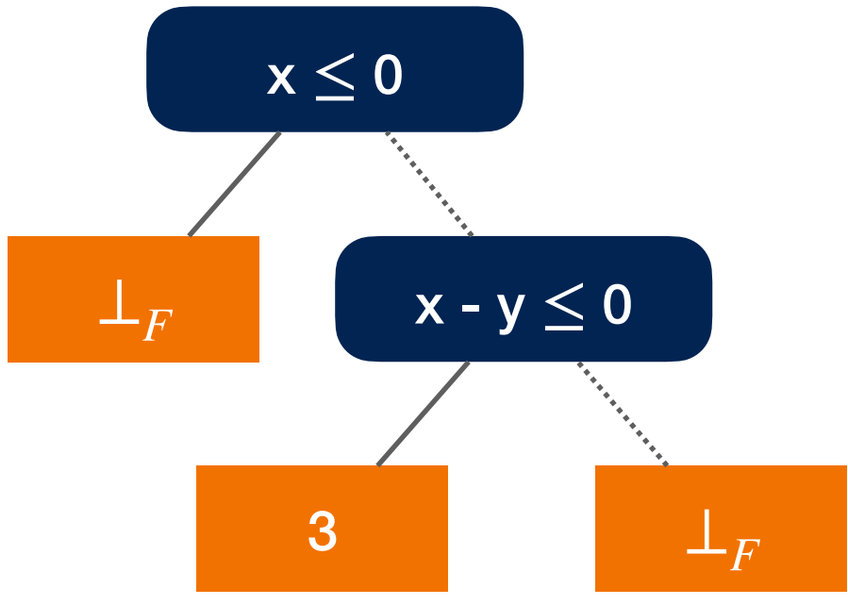








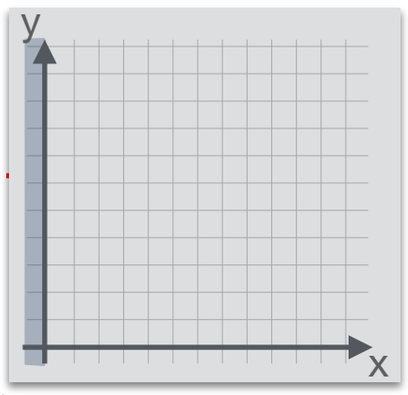


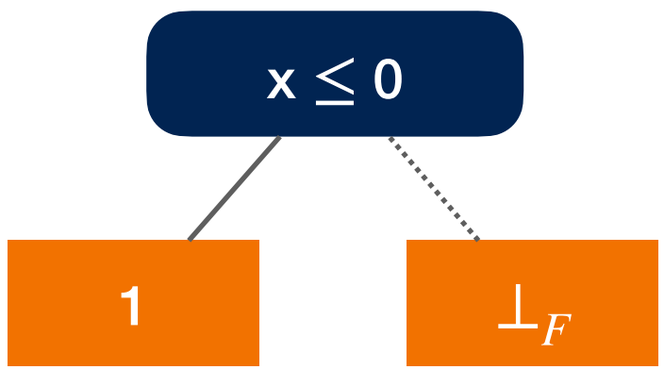


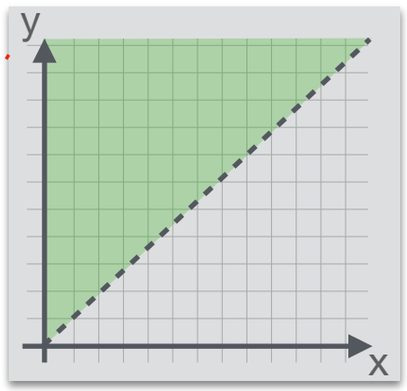
Abstract Definite Termination Semantics

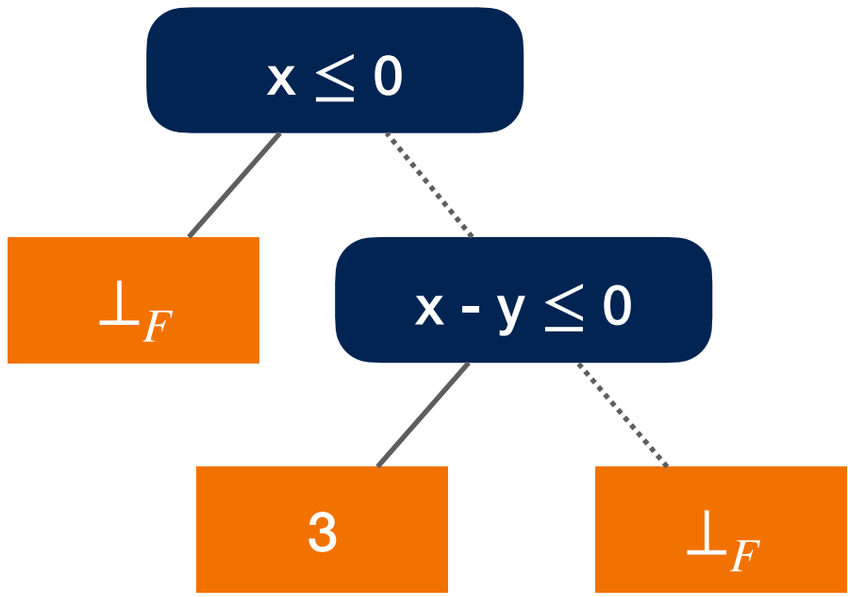
Example

```
1x ← [-∞, +∞]
2y ← [-∞, +∞]
 while 3(x > 0) do
       4x ← x - y
od5
```







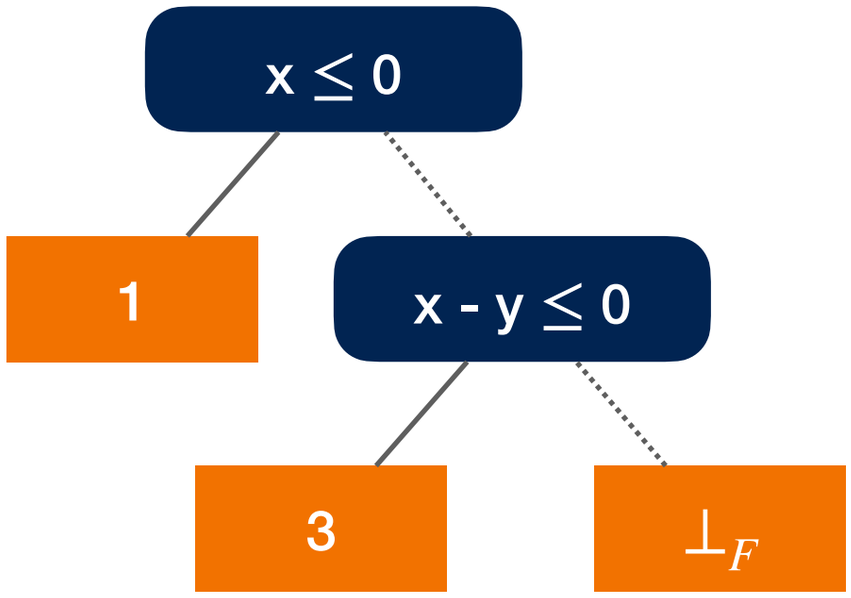


Abstract Definite Termination Semantics

Example

[1]x ← [-∞, +∞]
[2]y ← [-∞, +∞]
while [3](x > 0) **do**
    [4]x ← x - y
od[5]

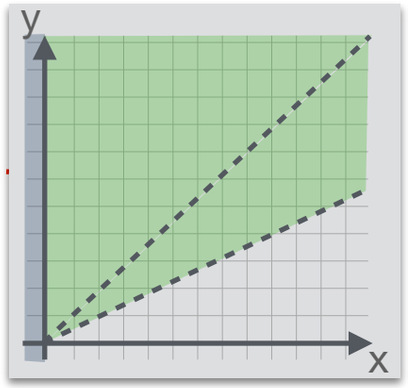
γ_A

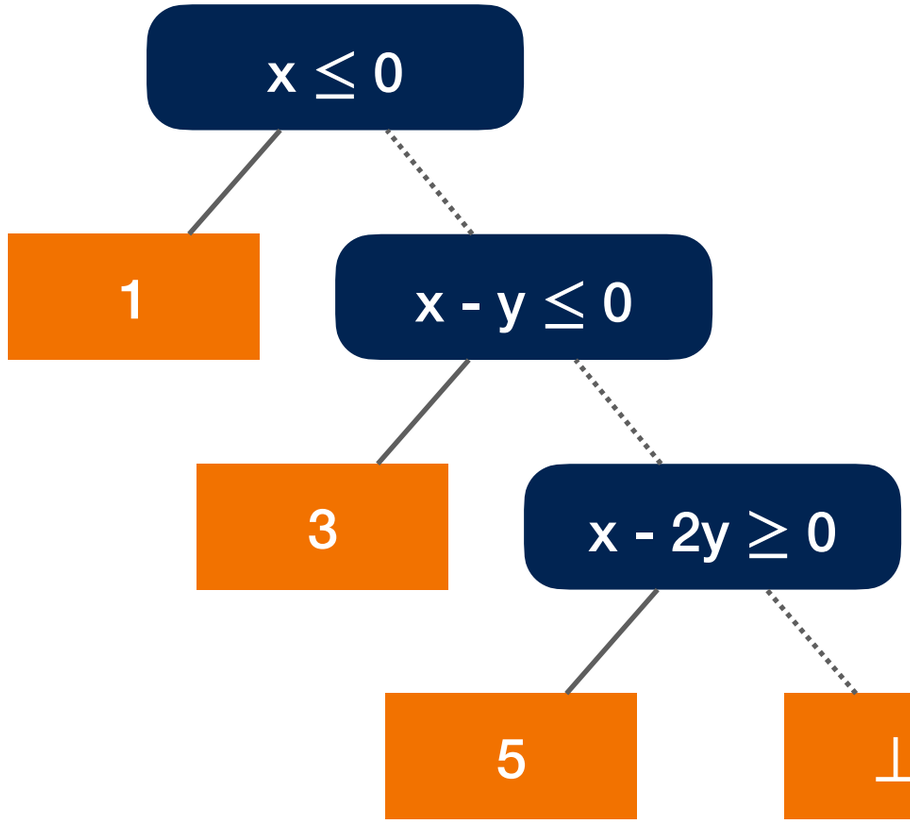


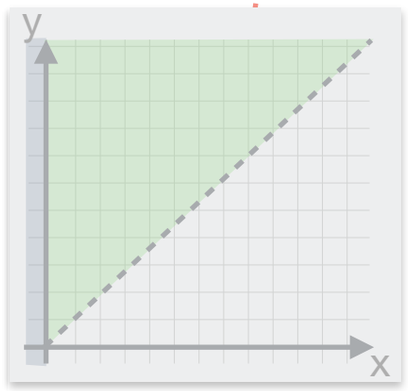
Abstract Definite Termination Semantics

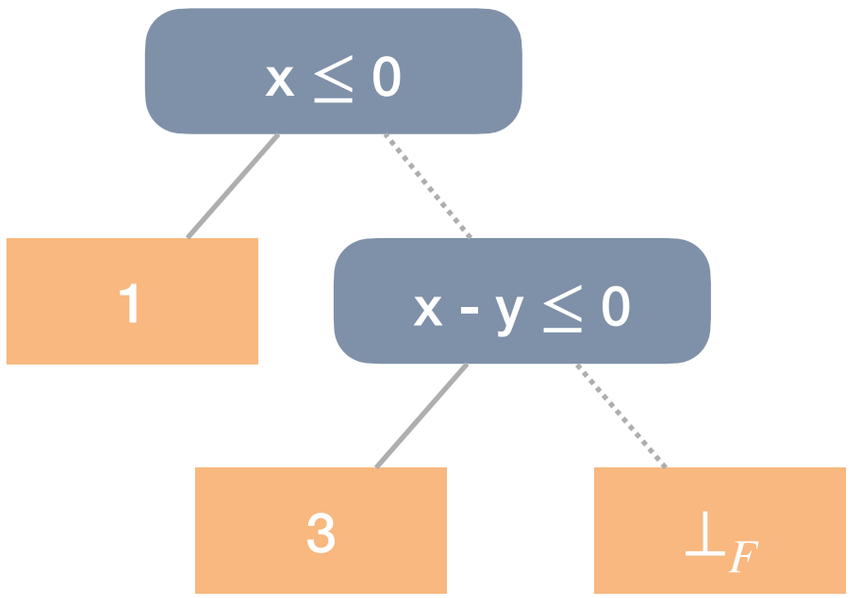
Example

[1]x ← [-∞, +∞]
[2]y ← [-∞, +∞]
while [3](x ≥ 0) **do**
    [4]x ← x - y
od[5]







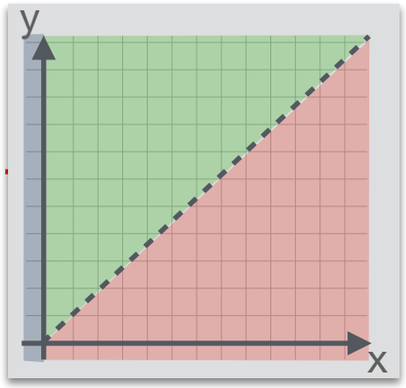


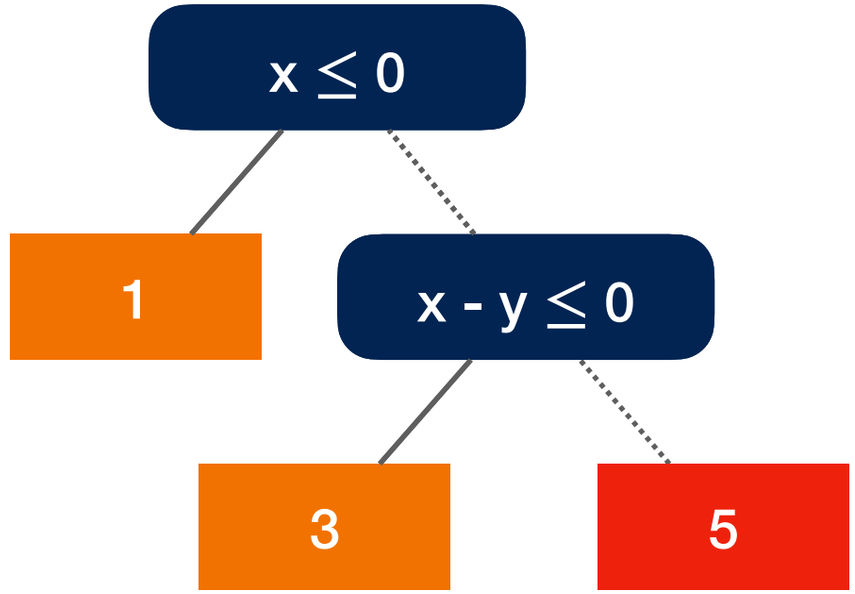
Abstract Definite Termination Semantics

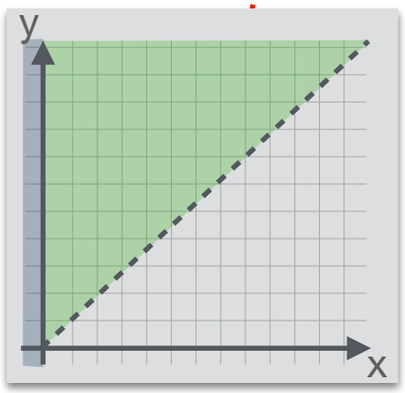
Example

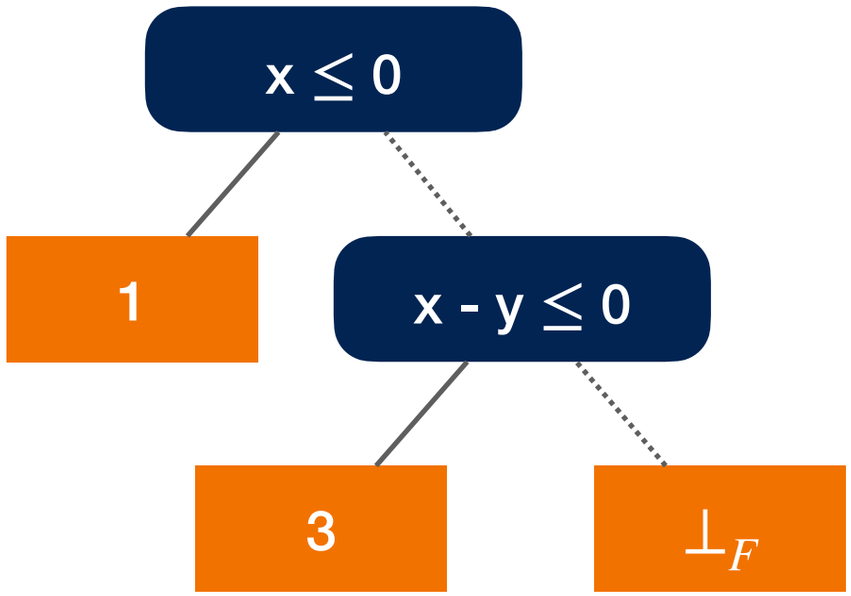
[1]x ← [-∞, +∞]
[2]y ← [-∞, +∞]
while [3](x > 0) **do**
  [4]x ← x - y
od[5]

∇_A





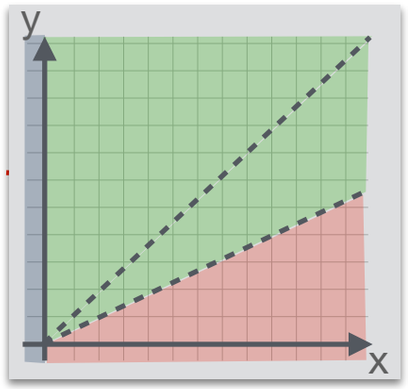


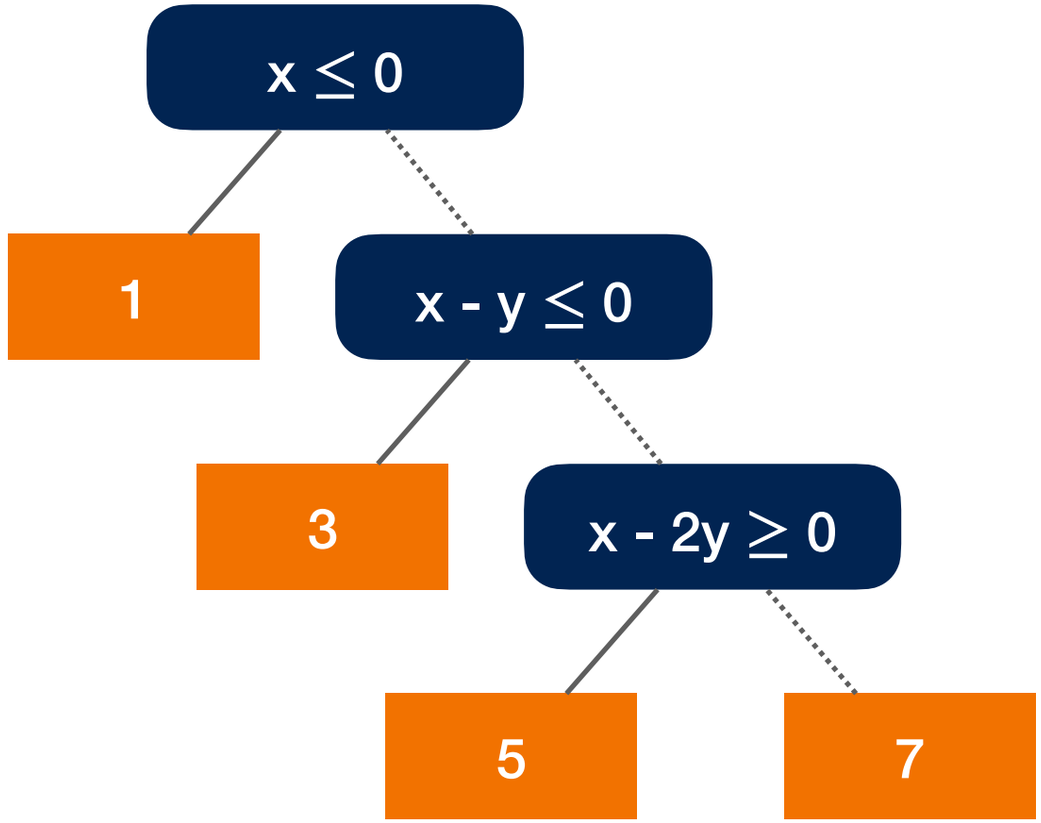


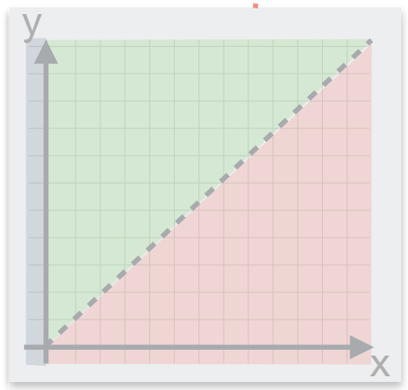
Abstract Definite Termination Semantics

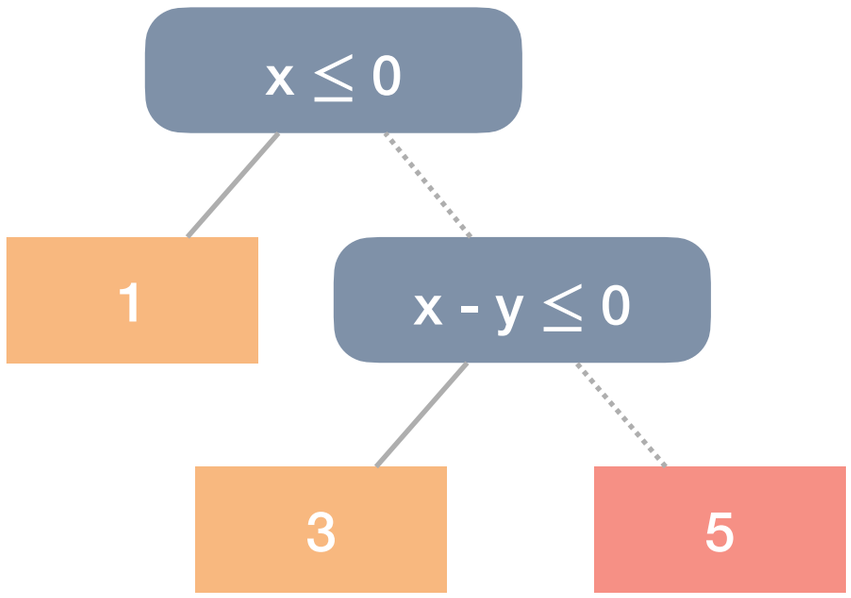
Example

[1]x ← [-∞, +∞]
[2]y ← [-∞, +∞]
while [3](x > 0) **do**
  [4]x ← x - y
od[5]







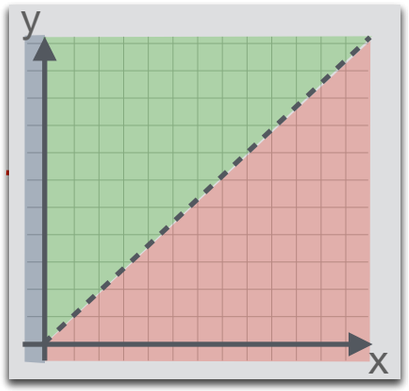


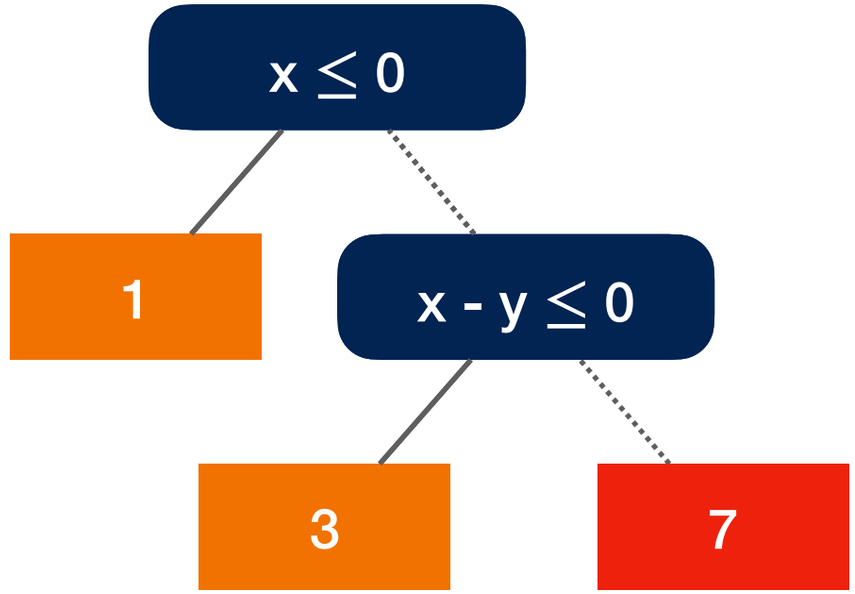
Abstract Definite Termination Semantics

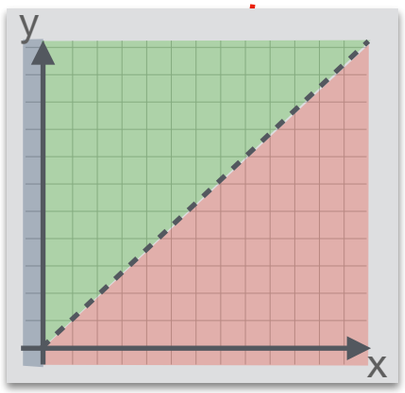
Example

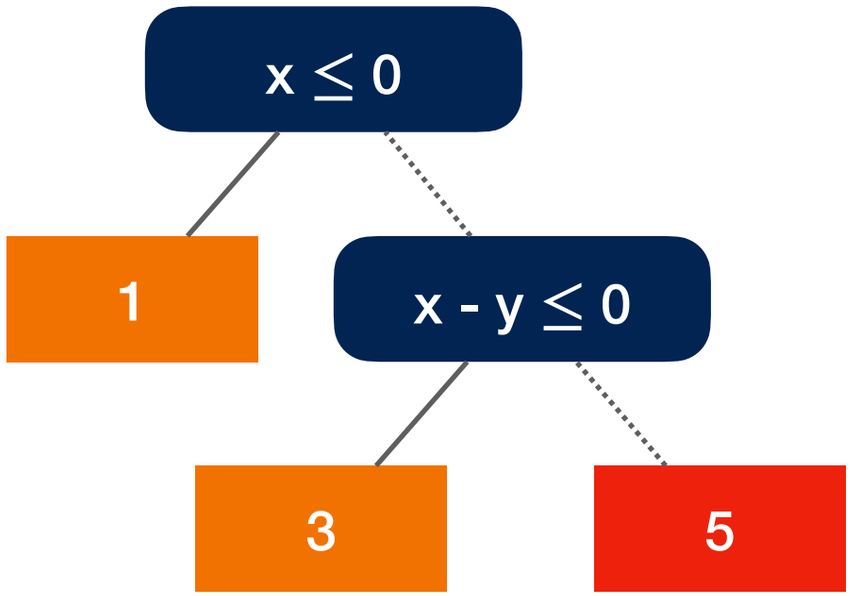
[1]x ← [-∞, +∞]
[2]y ← [-∞, +∞]
while [3](x > 0) **do**
  [4]x ← x - y
od[5]

∇_A







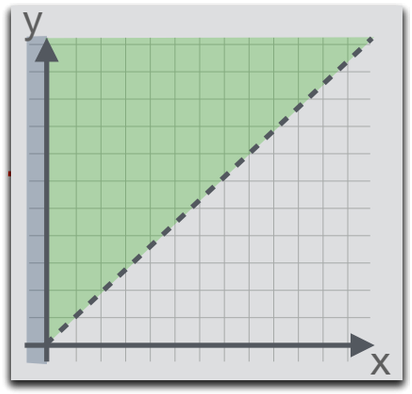


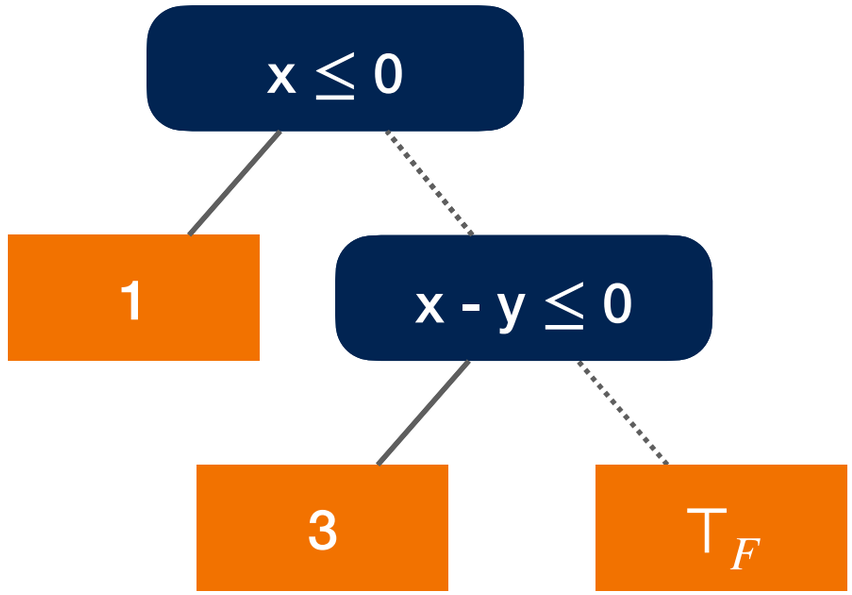
Abstract Definite Termination Semantics

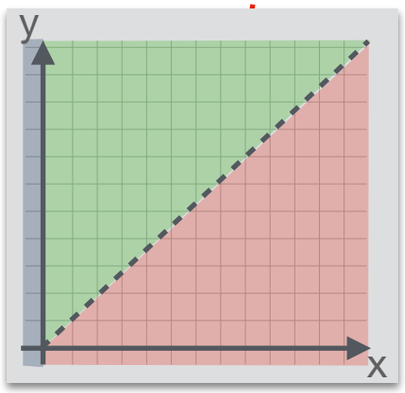
Example

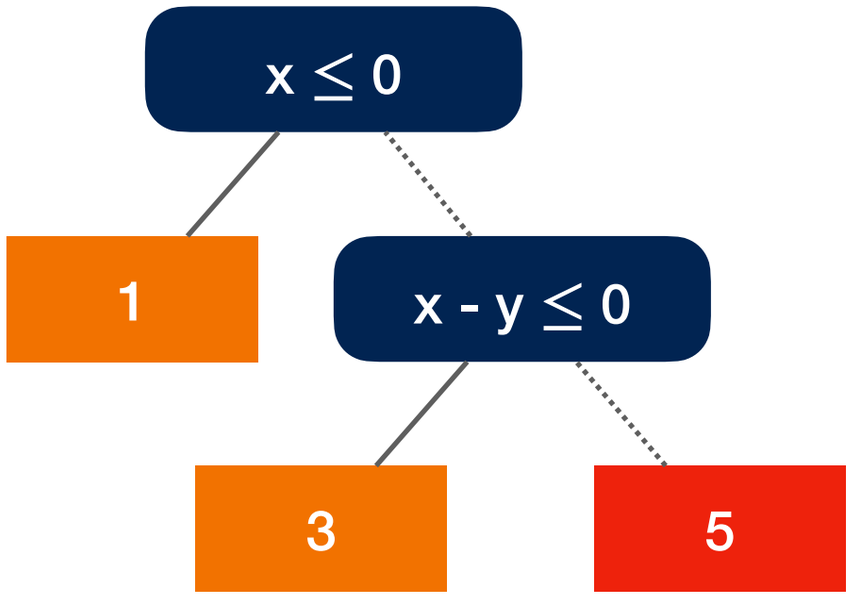
$^{1}x \leftarrow [-\infty, +\infty]$
$^{2}y \leftarrow [-\infty, +\infty]$
while $^{3}(x > 0)$ **do**
$^{4}x \leftarrow x - y$
od^{5}

∇_A





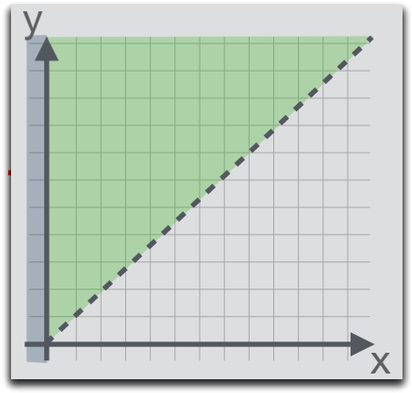


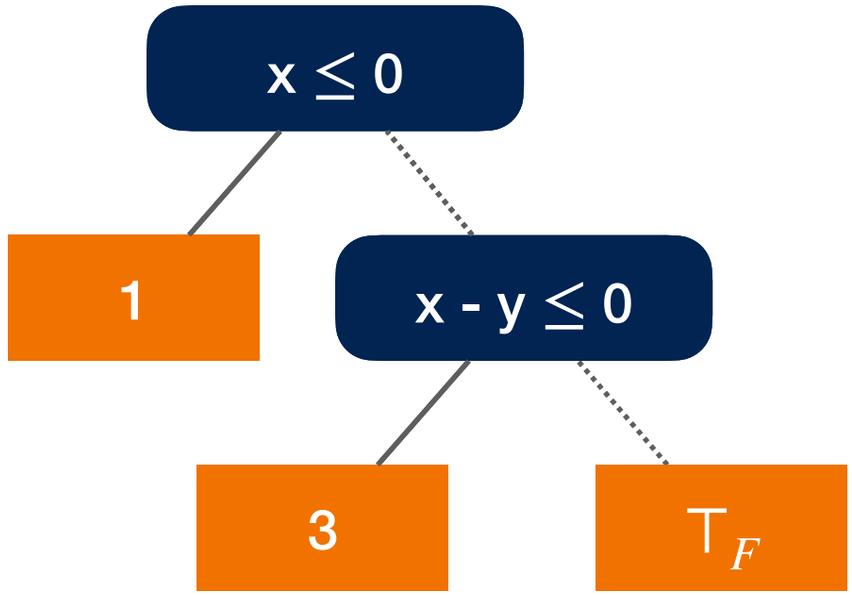


Abstract Definite Termination Semantics

Example

$^{1}x \leftarrow [-\infty, +\infty]$
$^{2}y \leftarrow [-\infty, +\infty]$
while $^{3}(x > 0)$ **do**
    $^{4}x \leftarrow x - y$
od^{5}





Guarantee Properties

Guarantee Properties

"something good eventually happens at least once"

$$\text{AF}\,\varphi$$

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

Example:

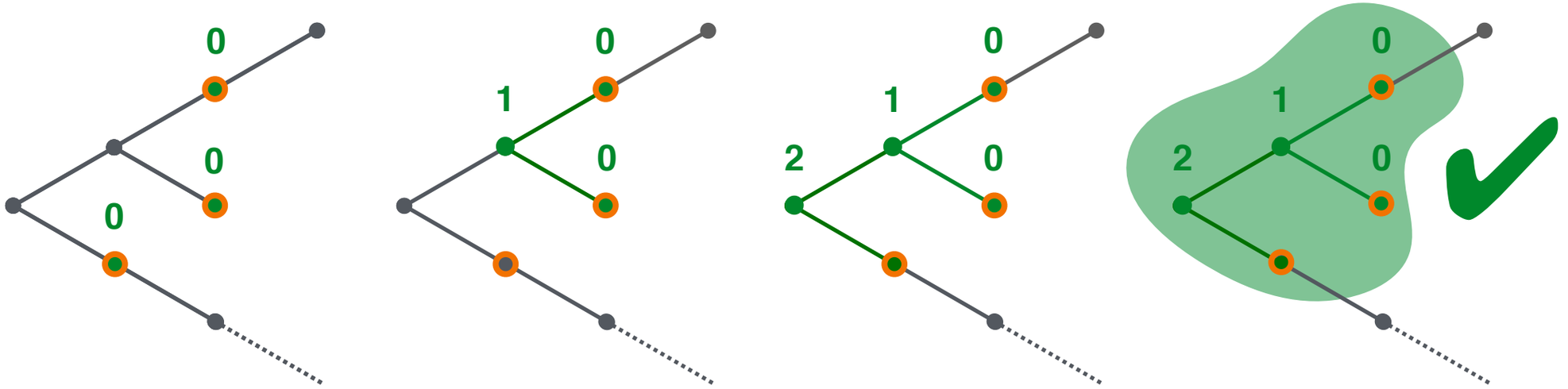
```
[1]x ← [-∞, +∞]
 while [2](x ≥ 0) do
       [3]x ← x + 1
od[4]
 while [5](0 ≥ 0) do
       if [6](x ≤ 10) do
             [7]x ← x + 1
       else
             [8]x ← -x
od[9]
```

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

Guarantee Semantics

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

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



Theorem

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

Abstract Guarantee Semantics

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

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

- $\mathscr{R}^{\varphi\#}_{G}[\![\textbf{if } {}^{\ell} e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=} \text{RESET}^{G}_{A}[\![\varphi]\!](X)$
  where $X \stackrel{\text{def}}{=} \text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_{G}[\![s]\!]t) \curlyvee_{T} \text{FILTER}_{A}[\![e \bowtie\!\!\!\!/\ 0]\!]t)$

- $\mathscr{R}^{\varphi\#}_{G}[\![\textbf{while } {}^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]t \stackrel{\text{def}}{=} \text{lfp}^{\#}\overline{F}^{\varphi\#}_{G}$
  where $\overline{F}^{\varphi\#}_{G}(x) \stackrel{\text{def}}{=} \text{RESET}^{G}_{A}[\![\varphi]\!](X)$
  $X \stackrel{\text{def}}{=} \text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_{G}[\![s]\!]x) \curlyvee_{T} \text{FILTER}_{A}[\![e \bowtie\!\!\!\!/\ 0]\!](t))$

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

Abstract Guarantee Semantics

Definition

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

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

where $\mathscr{R}^{\varphi\#}_{G}[\![\text{stat}]\!] : \mathscr{A} \to \mathscr{A}$ 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 $\mathscr{I}$ if $\mathscr{I} \subseteq \text{dom}(\gamma_{A}(\mathscr{R}^{\varphi\#}_{G}[\![\text{stat}^{\ell}]\!]))$

Recurrence Properties

"something good eventually happens infinitely often"

AG AF φ

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

Example:

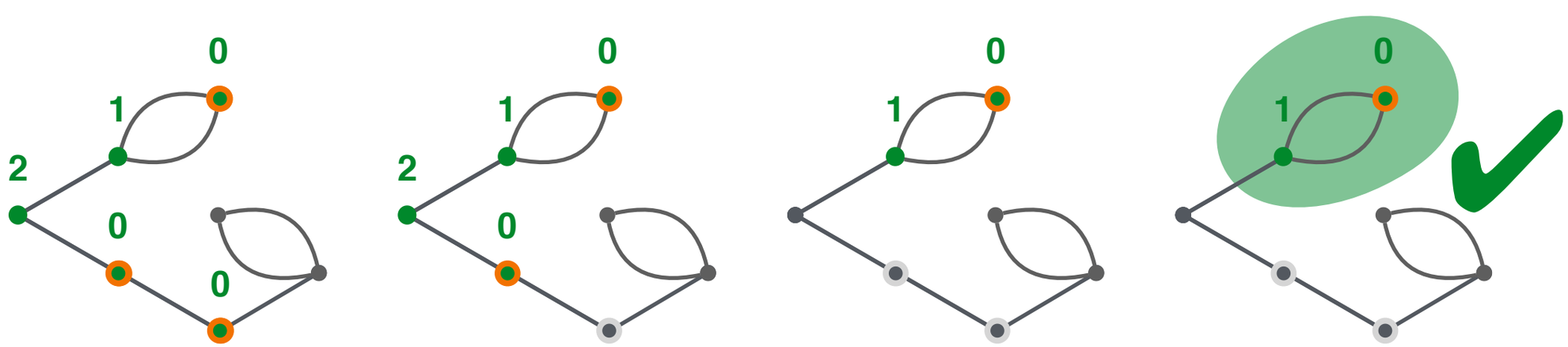
AG AF $(x = 3)$ is satisfied for $\mathscr{I} \stackrel{\text{def}}{=} \{(1,\rho) \in \Sigma \mid \rho(x) < 0\}$

```
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
```

Recurrence Semantics

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

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



Theorem

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

Abstract Recurrence Semantics

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

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

- $\mathscr{R}^{\varphi\#}_{R}[\![\textbf{if } ^{\ell} e \bowtie 0 \textbf{ then } s]\!]t \stackrel{\text{def}}{=} \text{RESET}^{R}_{A}[\![\varphi]\!](X)$
  where $X \stackrel{\text{def}}{=} \text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_{G}[\![s]\!]t) \curlyvee_{T} \text{FILTER}_{A}[\![e \bowtie\!\!\!\!/\ 0]\!]t)$

- $\mathscr{R}^{\varphi\#}_{R}[\![\textbf{while } ^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]t \stackrel{\text{def}}{=} \text{gfp}^{\#}_{G(t)} \overline{F}^{\varphi\#}_{R}$
  where $G \stackrel{\text{def}}{=} \mathscr{R}^{\varphi\#}_{G}[\![\textbf{while } ^{\ell} e \bowtie 0 \textbf{ do } s \textbf{ done}]\!]$
  $\overline{F}^{\varphi\#}_{R}(x) \stackrel{\text{def}}{=} \text{RESET}^{R}_{A}[\![\varphi]\!](X)$
  $X \stackrel{\text{def}}{=} \text{FILTER}_{A}[\![e \bowtie 0]\!](\mathscr{R}^{\varphi\#}_{R}[\![s]\!]x) \curlyvee_{T} \text{FILTER}_{A}[\![e \bowtie\!\!\!\!/\ 0]\!](t))$

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

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 $\mathscr{I} \subseteq \text{dom}(\gamma_A(\mathscr{R}_R^{\varphi\#}[\![\text{stat}^\ell]\!]))$

Bibliography

[Cousot02] **Patrick Cousot**. Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation. In Theoretical Computer Science 277(1-2):47–103, 2002.

[Cousot12] **Patrick Cousot and Radhia Cousot**. An Abstract Interpretation Framework for Termination. In POPL, pages 245–258, 2012.

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

extension with **ordinal-valued ranking functions**

[Urban17] **Nathanaël Courant and Caterina Urban**. Precise Widening Operators for Proving Termination by Abstract Interpretation. In TACAS, 2017.

extensions with **other widening heuristics**

[Urban18] **Caterina Urban, Samuel Ueltschi, and Peter Müller**. Abstract Interpretation of CTL Properties. In SAS, pages 402–422, 2018.

extension to properties in expressed in **computation tree logic**