Automatic Inference of Ranking Functions by Abstract Interpretation

Caterina Urban

Département d'Informatique
École Normale Supérieure

Project-Team ANTIQUE
INRIA Paris-Rocqencourt

Dagstuhl Seminar 14352
"Next Generation Static Software Analysis Tools"
Schloss Dagstuhl, Wadern, Germany

Proving Program Termination? Why?

30GB Zunes all over the world fail en masse | TechCrunch

techcrunch.com/2008/12/31/all-zune-30s-crapping

News TCTV Events Search

ANNOUNCEMENT Crunchies voting ends on Sunday, January 26, 2014, at 11:59 p.m. PST. Vote for your favorite startup here.

Gadgets Headline zune 30 Zune Feature

30GB Zunes all over the world fail en masse

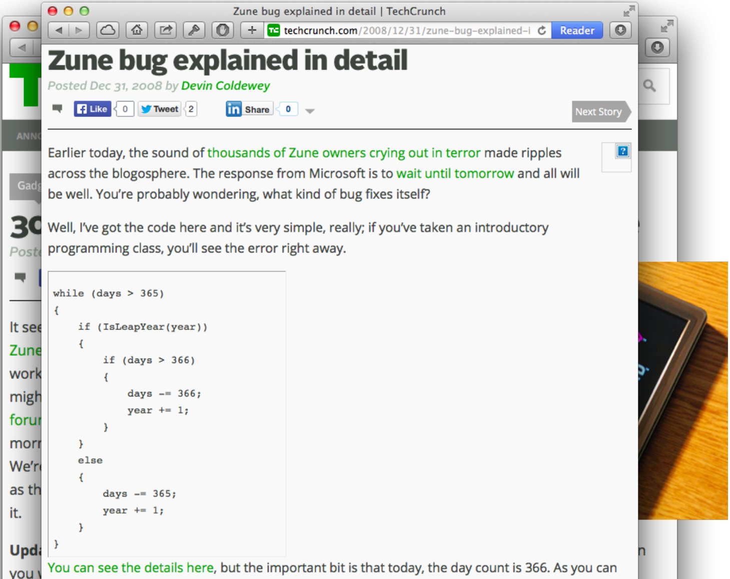
Posted Dec 31, 2008 by Matt Burns

Like 0 Tweet 0 Share 0

It seems that a random bug is affecting a bunch, if not every, 30GB Zunes. Real early this morning, a bunch of Zune 30s just stopped working. No official word from Redmond on this one yet but we might have a gadget Y2K going on here. Fan boards and support forums all have the same mantra saying that at 2:00 AM this morning, the Zune 30s reset on their own and doesn't fully reboot. We're sure Microsoft will get flooded with angry Zune owners as soon as the phone lines open up for the last time in 2008. More as we get it.

Update 2: The solution is ... kind of weak: let your Zune run out of battery and it'll be fixed when you wake up tomorrow and charge it.

Proving Program Termination? Why?



Zune bug explained in detail | TechCrunch

techcrunch.com/2008/12/31/zune-bug-explained-i

Zune bug explained in detail

Posted Dec 31, 2008 by Devin Coldewey

Like 0 Tweet 2 Share 0 Next Story

Earlier today, the sound of thousands of Zune owners crying out in terror made ripples across the blogosphere. The response from Microsoft is to wait until tomorrow and all will be well. You're probably wondering, what kind of bug fixes itself?

Well, I've got the code here and it's very simple, really; if you've taken an introductory programming class, you'll see the error right away.

```
while (days > 365)
{
    if (IsLeapYear(year))
    {
        if (days > 366)
        {
            days -= 366;
            year += 1;
        }
    }
    else
    {
        days -= 365;
        year += 1;
    }
}
```

You can see the details here, but the important bit is that today, the day count is 366. As you can

Outline

- **ranking functions**[1]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below
- **idea**: computation of ranking functions by abstract interpretation[2]
- family of **abstract domains** for program termination[3]
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on **ordinal-valued ranking functions**[4]
- instances based on **decision trees**[5]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)
[5] Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)

Outline

- **ranking functions**[1]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below
- **idea**: computation of ranking functions by abstract interpretation[2]

- family of **abstract domains** for program termination[3]
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on **ordinal-valued ranking functions**[4]
- instances based on **decision trees**[5]

---

1 Floyd - *Assigning Meanings to Programs* (1967)
2 Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
3 Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
4 Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)
5 Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)

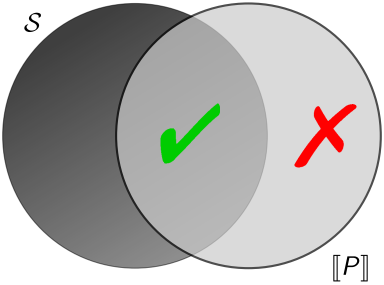
Outline

- **ranking functions**[1]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below
- **idea**: computation of ranking functions by abstract interpretation[2]
- family of **abstract domains** for program termination[3]
  - piecewise-defined ranking functions
  - backward invariance analysis
  - sufficient conditions for termination
- instances based on **ordinal-valued ranking functions**[4]
- instances based on **decision trees**[5]

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)
[3] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)
[4] Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)
[5] Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)

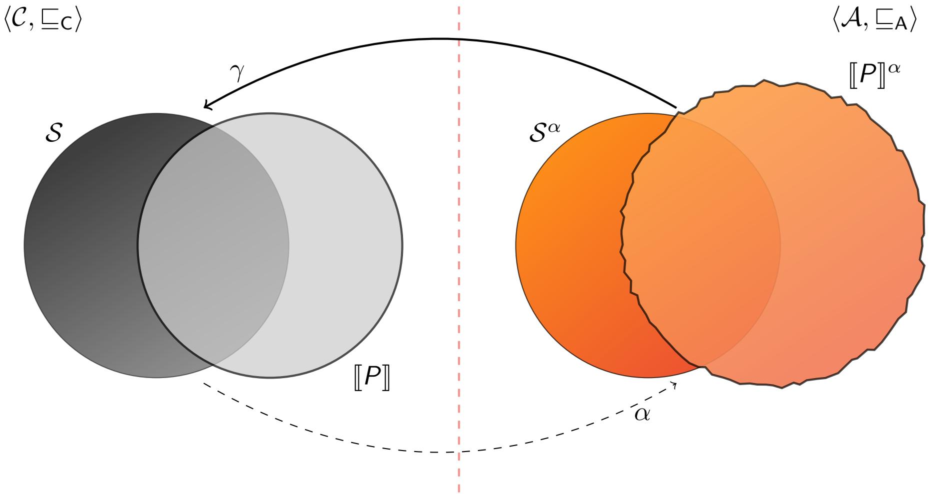
Abstract Interpretation[6]

$\langle \mathcal{C}, \sqsubseteq_{\mathsf{C}} \rangle$



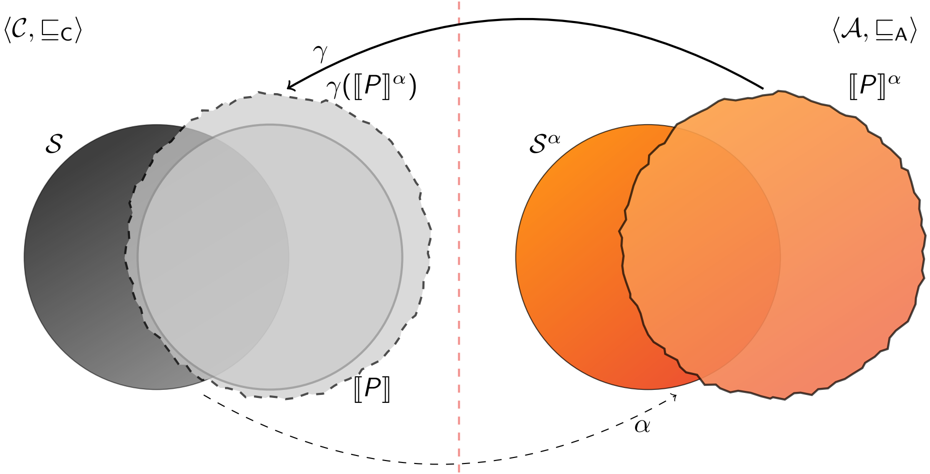
[6] Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)

Abstract Interpretation[6]

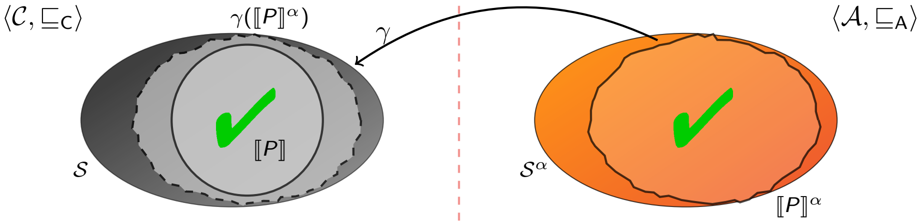


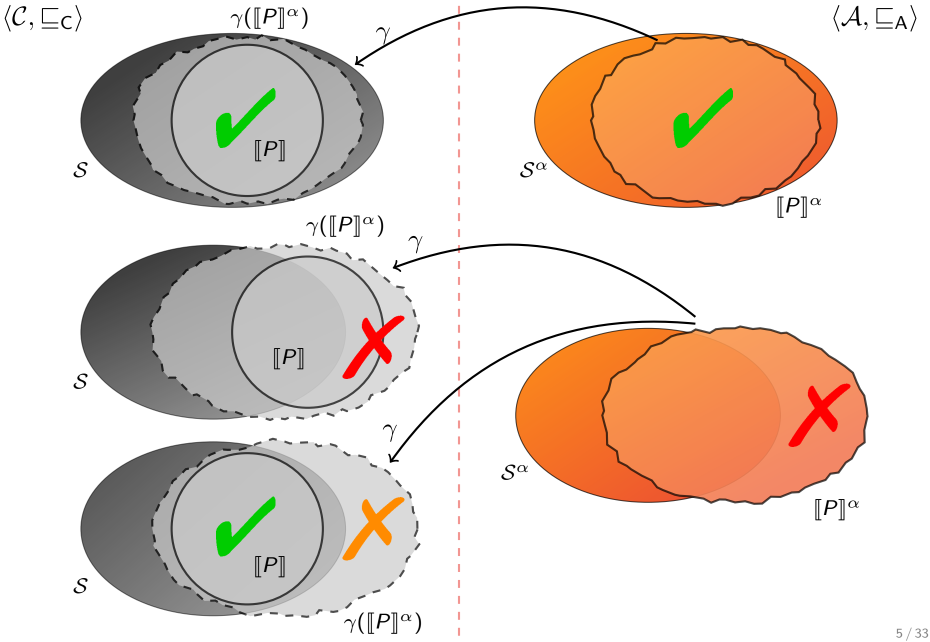
[6]Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)

Abstract Interpretation[6]



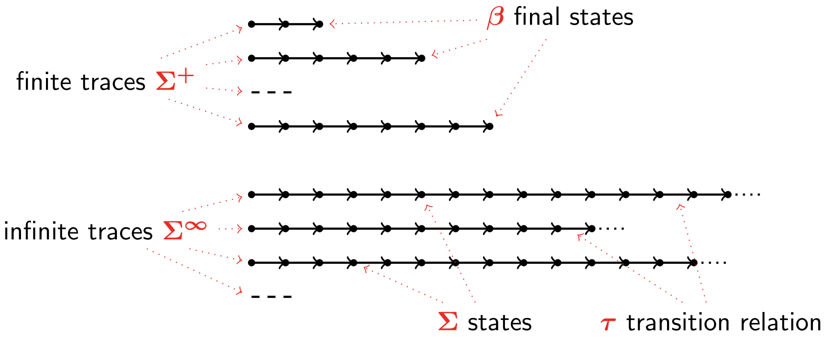
[6] Cousot&Cousot - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.* (POPL 1977)


$\langle \mathcal{C}, \sqsubseteq_C \rangle$
$\gamma(\llbracket P \rrbracket^\alpha)$
γ
$\langle \mathcal{A}, \sqsubseteq_A \rangle$
$\llbracket P \rrbracket$
$\mathcal{S}$
$\mathcal{S}^\alpha$
$\llbracket P \rrbracket^\alpha$


$\langle \mathcal{C}, \sqsubseteq_C \rangle$
$\gamma(\llbracket P \rrbracket^\alpha)$
γ
$\llbracket P \rrbracket$
$\mathcal{S}$
$\langle \mathcal{A}, \sqsubseteq_A \rangle$
$\mathcal{S}^\alpha$
$\llbracket P \rrbracket^\alpha$
$\gamma(\llbracket P \rrbracket^\alpha)$
γ
$\llbracket P \rrbracket$
$\mathcal{S}$
γ
$\llbracket P \rrbracket$
$\mathcal{S}$
$\gamma(\llbracket P \rrbracket^\alpha)$
$\mathcal{S}^\alpha$
$\llbracket P \rrbracket^\alpha$

Termination Semantics

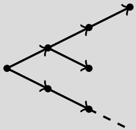
program $\mapsto$ **trace semantics**



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

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

Example



Theorem (Soundness and Completeness)

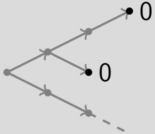
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

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

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

Example



Theorem (Soundness and Completeness)

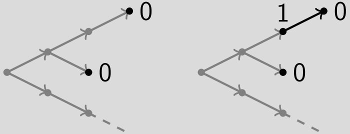
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

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

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

Example



Theorem (Soundness and Completeness)

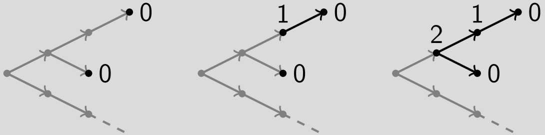
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

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

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

Example



Theorem (Soundness and Completeness)

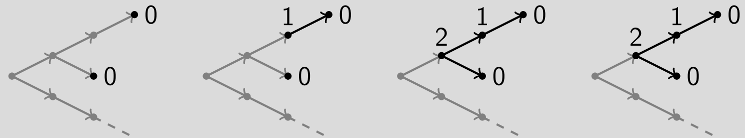
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

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

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

Example



Theorem (Soundness and Completeness)

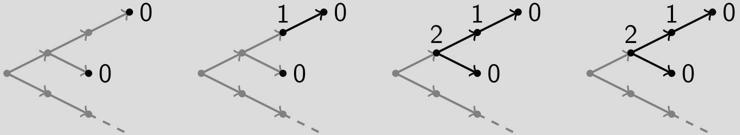
the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

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

idea = define a ranking function **counting the number of program steps** from the end of the program and **extracting the well-founded part** of the program transition relation

Example



Theorem (Soundness and Completeness)

the termination semantics is **sound** *and* **complete** *to prove the termination of programs*

Cousot&Cousot - *An Abstract Interpretation Framework for Termination* (POPL 2012)

- **remark**: the <u>termination semantics</u> is **not computable**!

Example

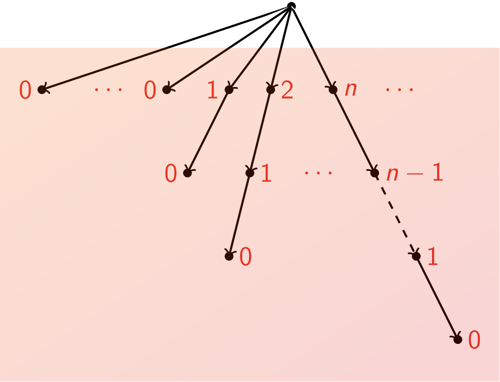
int : x

$x :=\ ?$

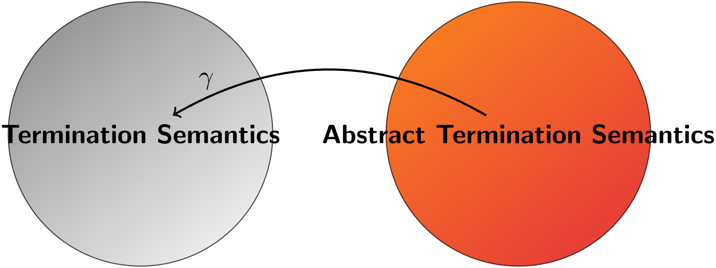
while $(x > 0)$ do

  $x := x - 1$

od

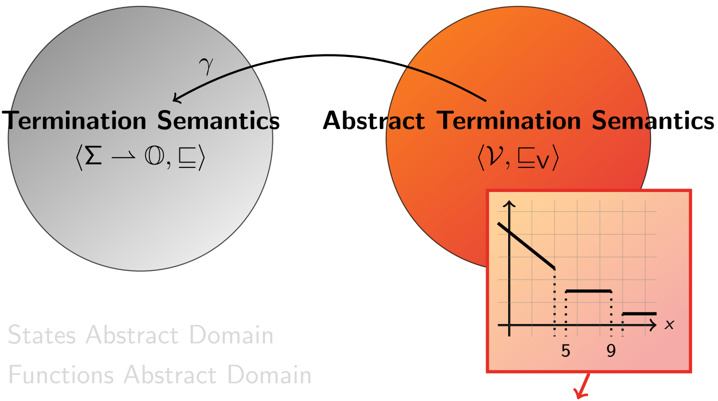


Piecewise-Defined Ranking Functions



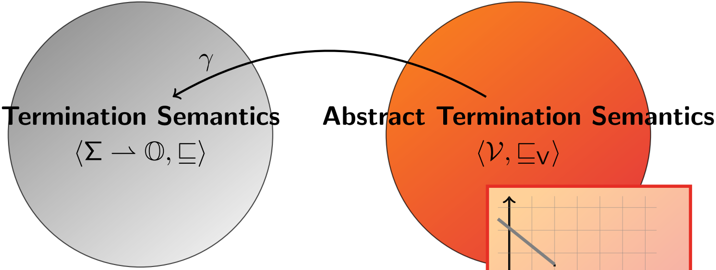
- States Abstract Domain $\mathcal{S}$
- Functions Abstract Domain $\mathcal{F}$
- Piecewise-Defined Ranking Functions Abstract Domain $\mathcal{V}(\mathcal{S}, \mathcal{F})$

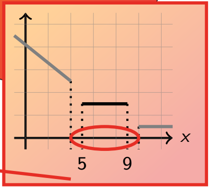
Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)



- States Abstract Domain S
- Functions Abstract Domain F
- Piecewise-Defined Ranking Functions Abstract Domain V(S, F)

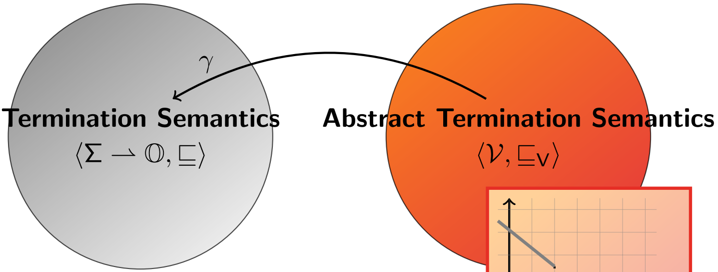
Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)

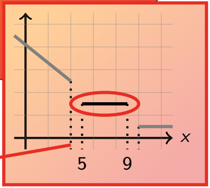




- States Abstract Domain — S
- Functions Abstract Domain — F
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, F)

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

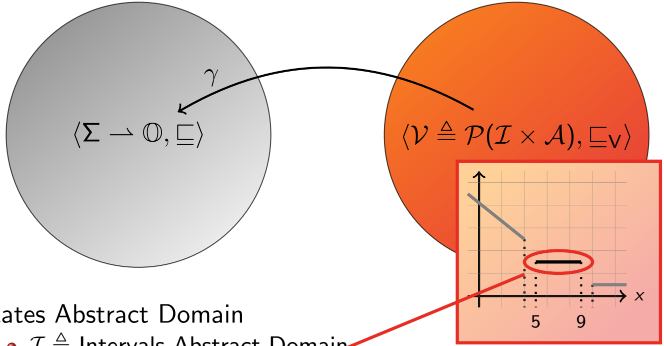




- States Abstract Domain S
- Functions Abstract Domain F
- Piecewise-Defined Ranking Functions Abstract Domain $\mathsf{V}(\mathsf{S}, \mathsf{F})$

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

Affine Ranking Functions Abstract Domain



- States Abstract Domain
  - $\mathcal{I} \triangleq$ Intervals Abstract Domain
- Functions Abstract Domain
  - $\mathcal{A} \triangleq \{\perp_A\} \cup \{f \mid f \in \mathbb{Z}^n \to \mathbb{N}\} \cup \{\top_A\}$
    where $f \equiv f(x_1, \ldots, x_n) = m_1 x_1 + \cdots + m_n x_n + q$

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

Example

int : x

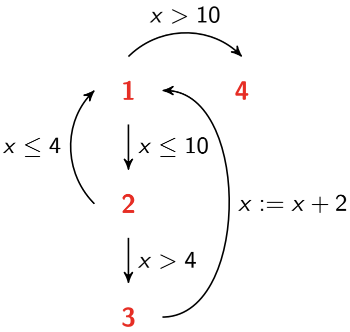
while [1]$(x \leq 10)$ do

  if [2]$(x > 4)$ then

    [3]$x := x + 2$

  fi

od[4]



we map each point to a function of x giving an **upper bound** on the steps before termination

Example

int : x

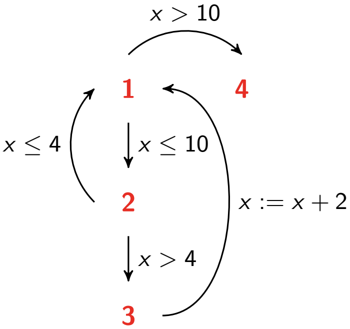
while [1]$(x \leq 10)$ do

  if [2]$(x > 4)$ then

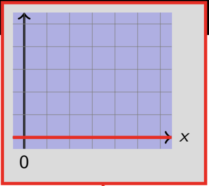
    [3]$x := x + 2$

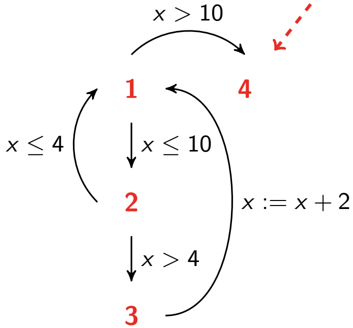
  fi

od[4]



we start at the end with 0 steps before termination





Example

int : x

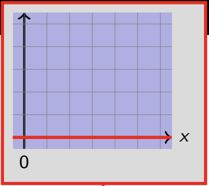
while $^{1}(x \leq 10)$ do

  if $^{2}(x > 4)$ then

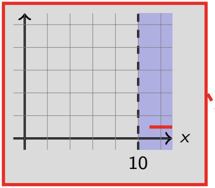
    $^{3}x := x + 2$

  fi

od^{4}



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



Example

int : x

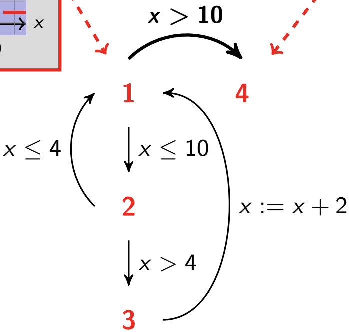
while [1]$(x \leq 10)$ do

  if [2]$(x > 4)$ then

    [3]$x := x + 2$

  fi

od[4]



Example

int : x

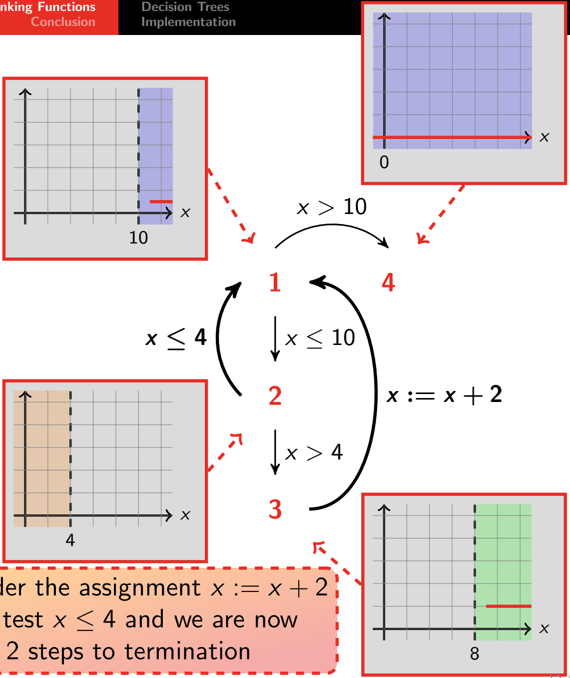
while [1]$(x \leq 10)$ do

  if [2]$(x > 4)$ then

    [3]$x := x + 2$

  fi

od[4]



we consider the assignment $x := x + 2$ or the test $x \leq 4$ and we are now at 2 steps to termination

Example

int : x

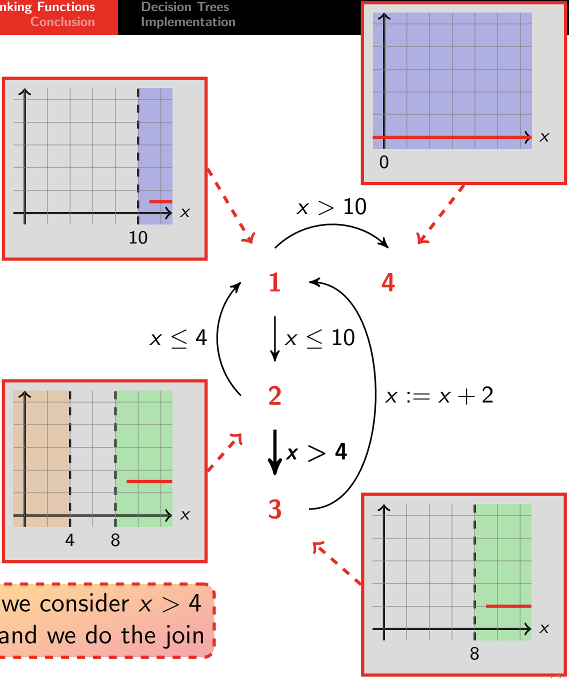
while $^1(x \leq 10)$ do

  if $^2(x > 4)$ then

    $^3x := x + 2$

  fi

od^4



we consider $x > 4$
and we do the join

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

Example

int : x

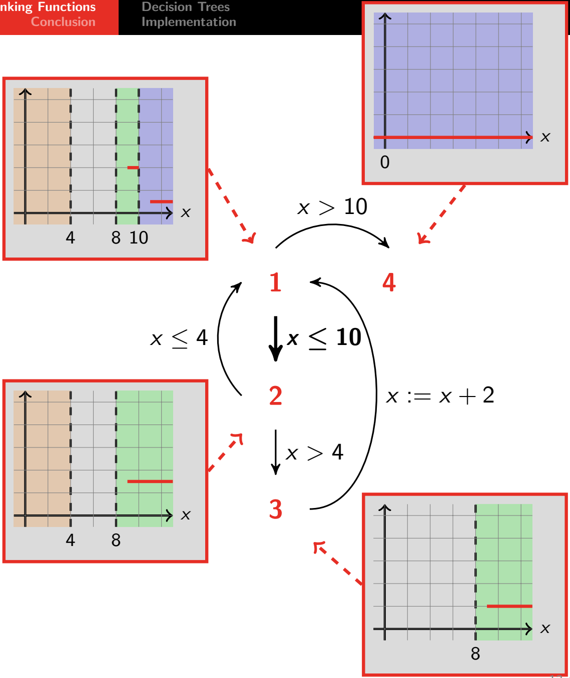
while [1]$(x \leq 10)$ do

if [2]$(x > 4)$ then

[3]$x := x + 2$

fi

od[4]



Example

int : x

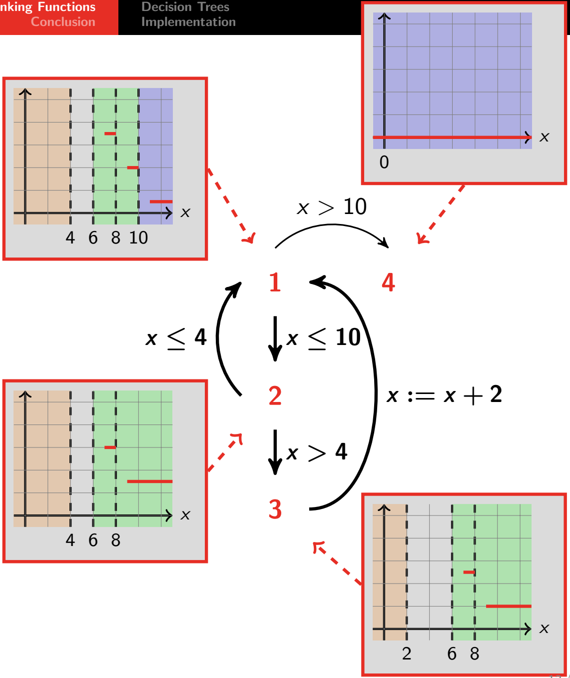
while $^{1}(x \leq 10)$ do

  if $^{2}(x > 4)$ then

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

  fi

od^{4}



we do the widening

Example

int : x

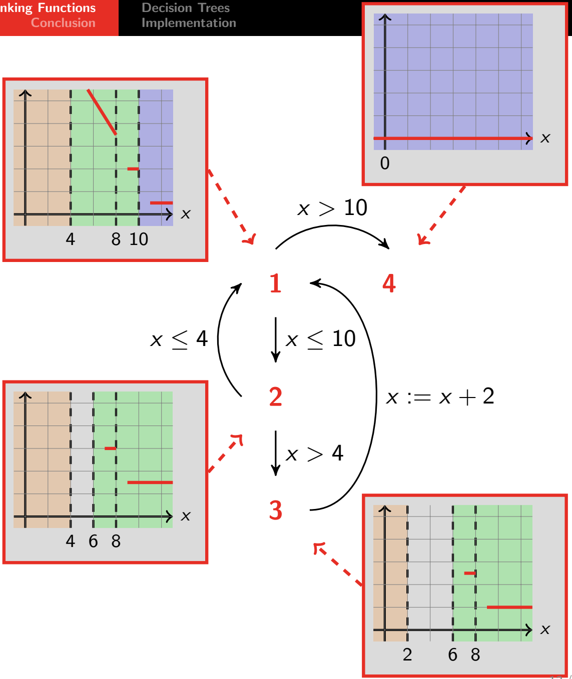
while ${}^{1}(x \leq 10)$ do

  if ${}^{2}(x > 4)$ then

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

  fi

od^{4}



Example

int : x

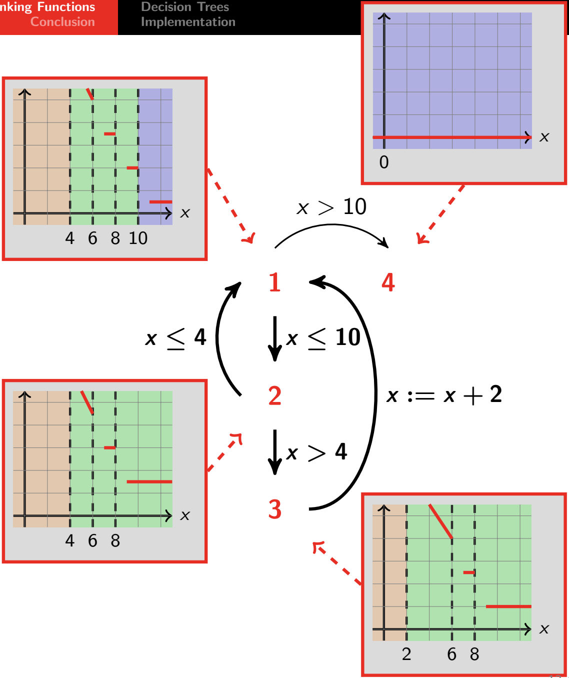
while [1]$(x \leq 10)$ do

  if [2]$(x > 4)$ then

    [3]$x := x + 2$

  fi

od[4]



the analysis provides $x > 4$ as
sufficient precondition
for termination

Example

int : x

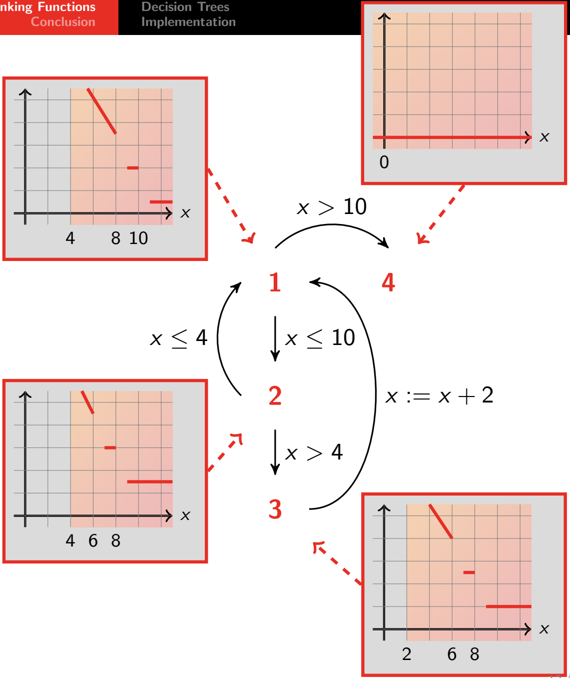
while $^{1}(x \leq 10)$ do

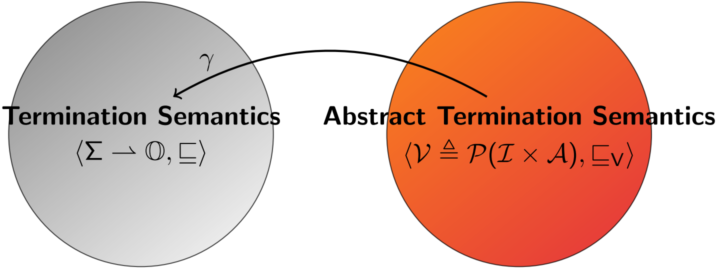
  if $^{2}(x > 4)$ then

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

  fi

od^{4}





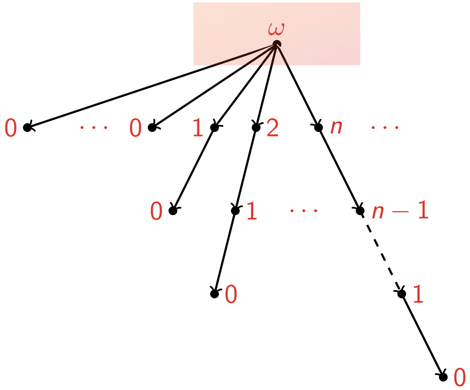
Theorem (Soundness)

the abstract termination semantics is **sound**
to prove the termination of programs

- **remark**: natural-valued ranking functions are **not sufficient**!

Example

```
int : x
x :=  ?
while (x > 0) do
   x := x − 1
od
```



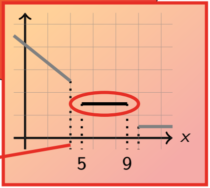
γ

Termination Semantics

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

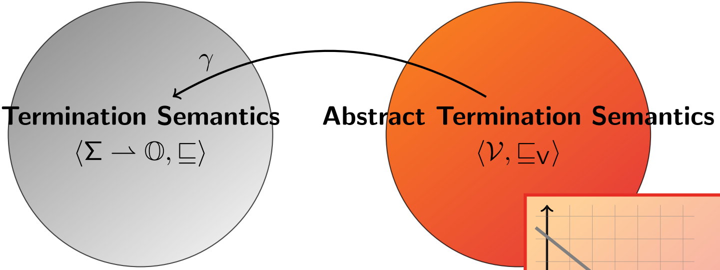
Abstract Termination Semantics

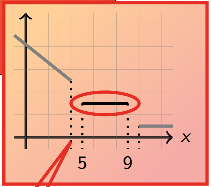
$\langle \mathcal{V}, \sqsubseteq_{\mathsf{V}} \rangle$



- States Abstract Domain — S
- Functions Abstract Domain — F
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, F)

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

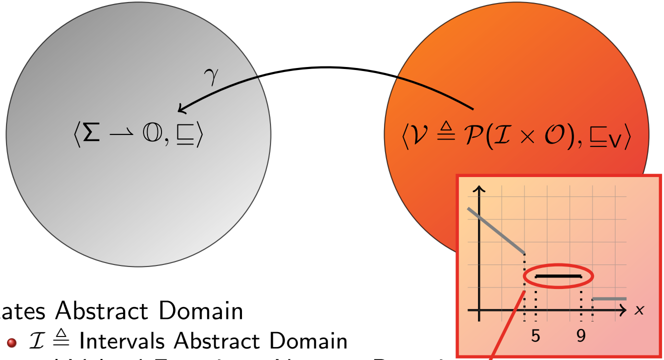




- States Abstract Domain $\quad$ S
- **Natural-Valued** Functions Abstract Domain $\quad$ F
- **Ordinal-Valued** Functions Abstract Domain $\quad$ **O**(F)
- Piecewise-Defined Ranking Functions Abstract Domain $\quad$ V(S, **O**(F))

Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)

Ordinal-Valued Ranking Functions Domain



- States Abstract Domain
  - $\mathcal{I} \triangleq$ Intervals Abstract Domain
- Natural-Valued Functions Abstract Domain
  - $\mathcal{A} \triangleq$ Affine Ranking Functions Abstract Domain
- Ordinal-Valued Functions Abstract Domain
  - $\mathcal{O} \triangleq \{\perp_O\} \cup \{\sum_i \omega^i \cdot f_i \mid f_i \in \mathcal{A} \setminus \{\perp_A, \top_A\}\} \cup \{\top_O\}$

Urban&Miné - *An Abstract Domain to Infer Ordinal-Valued Ranking Functions* (ESOP 2014)

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad x_1 := ?$$

$$[-\infty, +\infty] \mapsto o \triangleq ?$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad x_1 := ?$$

$$[-\infty, +\infty] \mapsto o \triangleq + 1$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad x_1 := ?$$

$$[-\infty, +\infty] \mapsto o \triangleq + x_2 + 1$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad x_1 := ?$$

$$[-\infty, +\infty] \mapsto o \triangleq \omega^2 \cdot 1 + \omega \cdot 0 + x_2 + 1$$

$$\boxed{\omega \cdot \omega = \omega^2 \cdot 1 + \omega \cdot 0}$$

Backward Non-Deterministic Assignments

- non-deterministic assignments are carried out in ascending powers of ω

Example

$$[-\infty, +\infty] \mapsto o \triangleq \omega \cdot x_1 + x_2$$

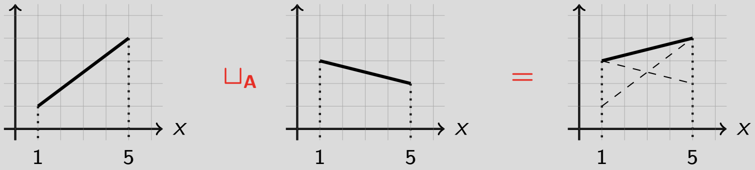
$$\Downarrow \quad x_1 := ?$$

$$[-\infty, +\infty] \mapsto o \triangleq \omega^2 + x_2 + 1$$

Join

- join: $\sqcup_A$

Example



- join: $\sqcup_O$

Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of ω

Example

$$\begin{array}{rcccccccccccc}
[-\infty, +\infty] & \mapsto & o_1 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & x_2 & + & 3 \\
[-\infty, +\infty] & \mapsto & o_2 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
[-\infty, +\infty] & \mapsto & o_1 \sqcup_O o_2 & \triangleq & ? & & & & & & & &
\end{array}$$

Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of ω

Example

$$
\begin{array}{lcccccccccccc}
[-\infty,+\infty] & \mapsto & o_1 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & x_2 & + & \mathbf{3} \\
[-\infty,+\infty] & \mapsto & o_2 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & (-x_2) & + & \mathbf{4} \\
\hline
[-\infty,+\infty] & \mapsto & o_1 \sqcup_O o_2 & \triangleq & & & & & & & & + & \mathbf{4}
\end{array}
$$

Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of ω

Example

$$
\begin{array}{lclcl}
[-\infty, +\infty] \mapsto & o_1 & \triangleq & \omega^2 \cdot x_1 + \omega \cdot x_2 + 3 \\
[-\infty, +\infty] \mapsto & o_2 & \triangleq & \omega^2 \cdot x_1 + \omega \cdot (-x_2) + 4 \\
\hline
[-\infty, +\infty] \mapsto & o_1 \sqcup_O o_2 & \triangleq & \omega^2 \cdot 1 + \omega \cdot 0 + 4
\end{array}
$$

$\omega \cdot \omega = \omega^2 \cdot 1 + \omega \cdot 0$

Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of ω

Example

$$\begin{array}{lcccccccccccc}
[-\infty, +\infty] & \mapsto & o_1 & \triangleq & \omega^2 & \cdot & x_1 & & + & \omega & \cdot & x_2 & + & 3 \\
[-\infty, +\infty] & \mapsto & o_2 & \triangleq & \omega^2 & \cdot & x_1 & & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
[-\infty, +\infty] & \mapsto & o_1 \sqcup_O o_2 & \triangleq & \omega^2 & \cdot & x_1 & \omega^2 \cdot 1 & & & & & + & 4
\end{array}$$

Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of ω

Example

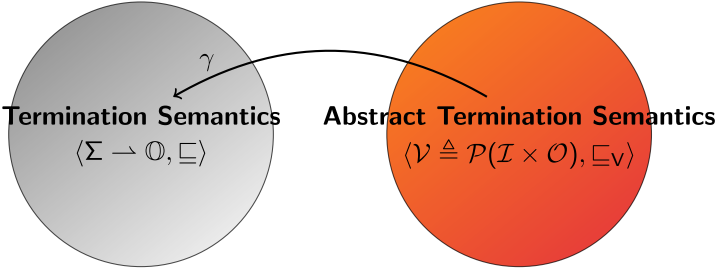
$$\begin{array}{lcccccccc}
[-\infty,+\infty] \mapsto & o_1 & \triangleq & \omega^2 \cdot \mathbf{x_1} & + & \omega \cdot x_2 & + & 3 \\
[-\infty,+\infty] \mapsto & o_2 & \triangleq & \omega^2 \cdot \mathbf{x_1} & + & \omega \cdot (-x_2) & + & 4 \\
\hline
[-\infty,+\infty] \mapsto & o_1 \sqcup_O o_2 & \triangleq & \omega^2 \cdot \mathbf{(x_1+1)} & & & + & 4
\end{array}$$

Join

- join: $\sqcup_A$
- join: $\sqcup_O$
  - $\sqcup_A$ in ascending powers of ω

Example

$$
\begin{array}{lllllllll}
[-\infty, +\infty] \mapsto & o_1 & \triangleq & \omega^2 \cdot x_1 & + & \omega \cdot x_2 & + & 3 \\
[-\infty, +\infty] \mapsto & o_2 & \triangleq & \omega^2 \cdot x_1 & + & \omega \cdot (-x_2) & + & 4 \\
\hline
[-\infty, +\infty] \mapsto & o_1 \sqcup_O o_2 & \triangleq & \omega^2 \cdot (x_1 + 1) & & & + & 4
\end{array}
$$



Theorem (Soundness)

the abstract termination semantics is **sound**
to prove the termination of programs

Example

int : x_1, x_2

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

  if $^2($? $)$ then

    $^3 x_1 := x_1 - 1$

    $^4 x_2 :=$?

  else

    $^5 x_2 := x_2 - 1$

od^6

$$f_1(x_1, x_2) = \begin{cases} 1 & x_1 \leq 0 \lor x_2 \leq 0 \\ \omega \cdot (x_1 - 1) + 7x_1 + 3x_2 - 5 & x_1 > 0 \land x_2 > 0 \end{cases}$$

Example

int : x_1, x_2
while $^1(x_1 \neq 0 \wedge x_2 > 0)$ do
  if $^2(x_1 > 0)$ then
    if $^3($? $)$ then
      $^4 x_1 := x_1 - 1$
      $^5 x_2 :=$?
    else
      $^6 x_2 := x_2 - 1$
  else /* $x_1 < 0$ */
    if $^7($? $)$ then
      $^8 x_1 := x_1 + 1$
    else
      $^9 x_2 := x_2 - 1$
      $^{10} x_1 :=$?
od^{11}

$$f_1(x_1, x_2) = \begin{cases} \omega^2 + \omega \cdot (x_2 - 1) - 4x_1 + 9x_2 - 2 & x_1 < 0 \wedge x_2 > 0 \\ 1 & x_1 = 0 \vee x_2 \leq 0 \\ \omega \cdot (x_1 - 1) + 9x_1 + 4x_2 - 7 & x_1 > 0 \wedge x_2 > 0 \end{cases}$$

Example

int : x_1, x_2

while [1]$(x_1 \neq 0 \wedge x_2 > 0)$ do

  if [2]$(x_1 > 0)$ then

    if [3](?) then

      [4]$x_1 := x_1 - 1$

      [5]$x_2 :=$?

    else

      [6]$x_2 := x_2 - 1$

else /* $x_1 < 0$ */

  if [7](?) then

    [8]$x_1 := x_1 + 1$

  else

    [9]$x_2 := x_2 - 1$

    [10]$x_1 :=$?

the coefficients and their **order** are **inferred by the analysis**

$$f_1(x_1, x_2) = \begin{cases} \omega^2 + \omega \cdot (x_2 - 1) - 4x_1 + 9x_2 - 2 & x_1 < 0 \wedge x_2 > 0 \\ 1 & x_1 = 0 \vee x_2 \leq 0 \\ \omega \cdot (x_1 - 1) + 9x_1 + 4x_2 - 7 & x_1 > 0 \wedge x_2 > 0 \end{cases}$$

Non-Linear Ranking Functions

Example

int : N, x_1, x_2
[1]$x_1 := N$
while [2]$(x_1 \geq 0)$ do
  [3]$x_2 := N$
  while [4]$(x_2 \geq 0)$ do
    [5]$x_2 := x_2 - 1$
  od[6]
  [7]$x_1 := x_1 - 1$
od[8]

$$f_1(x_1, x_2, N) = \begin{cases} 1 & x_1 < 0 \\ \omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0 \end{cases}$$

Non-Linear Ranking Functions

Example

int : $N,\ x_1,\ x_2$

[1] $x_1 := N$

while [2] $(x_1 \geq 0)$ do

  [3] $x_2 := N$

  while [4] $(x_2 \geq 0)$ do

    [5] $x_2 := x_2 - 1$

  od [6]

  [7] $x_1 := x_1 - 1$

od [8]

$$f_1(x_1, x_2, N) = \begin{cases} 1 & x_1 < 0 \\ \omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0 \end{cases}$$

the loop terminates in a
finite number of iterations

- **remark**: the intervals abstract domain is **not sufficient**!

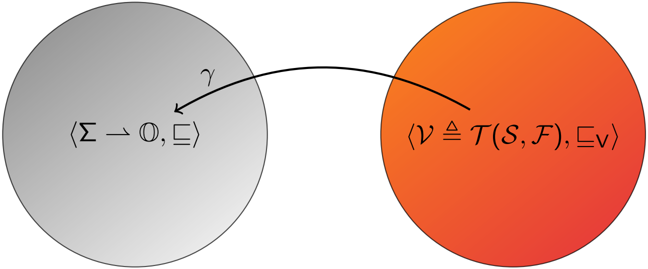
Example

int : $x, \ y, \ r$
while $(r > 0)$ do
  $r := r + x$
  $r := r - y$
od

$$f(x, y, r) = \begin{cases} 1 & r \leq 0 \\ 3r + 1 & r > 0 \land x \leq y \\ \text{undefined} & r > 0 \land x \geq y \end{cases}$$

- $\mathcal{V} \triangleq \mathcal{P}(\mathcal{S} \times \mathcal{F})$ when $\mathcal{S} \triangleq$ Octagons/Polyhedra **does not scale**!

Decision Tree Abstract Domain



- States Abstract Domain
  - $\mathcal{S} \triangleq$ Intervals/Octagons/Polyhedra Abstract Domain
- Functions Abstract Domain
  - $\mathcal{F} \triangleq$ Natural/Ordinal-Valued Ranking Functions Abstract Domain
- Piecewise-Defined Ranking Functions Abstract Domain
  - $\mathcal{T} \triangleq \{\mathrm{LEAF} : f \mid f \in \mathcal{F}\} \cup \{\mathrm{NODE}\{s\} : t_1, t_2 \mid s \in \mathcal{S} \wedge t_1, t_2 \in \mathcal{T}\}$

Urban&Miné - *A Decision Tree Abstract Domain for Proving Conditional Termination* (SAS 2014)

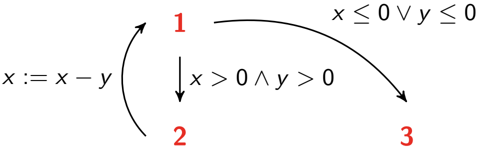
Example

int : x

while [1]$(x > 0 \land y > 0)$ do

[2]$x := x - y$

od[3]



we map each point
to a function of x and y giving
an **upper bound** on the
steps before termination

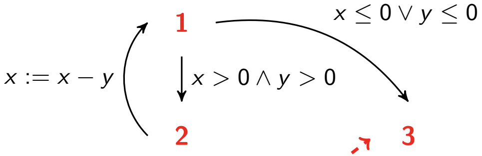
Example

int : x

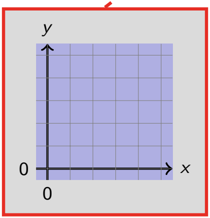
while $^{1}(x > 0 \wedge y > 0)$ do

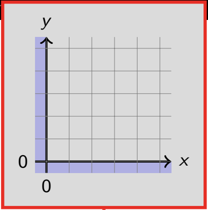
$\quad ^{2}x := x - y$

od^{3}



we start at the end
with 0 steps
before termination





we take into account $x \leq 0$ and we have 1 step to termination

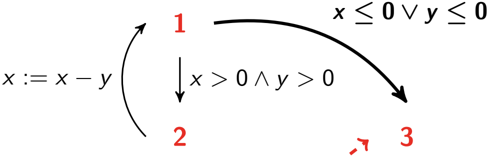
Example

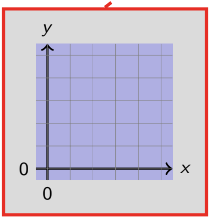
int : x

while $^1(x > 0 \wedge y > 0)$ do

$^2x := x - y$

od^3





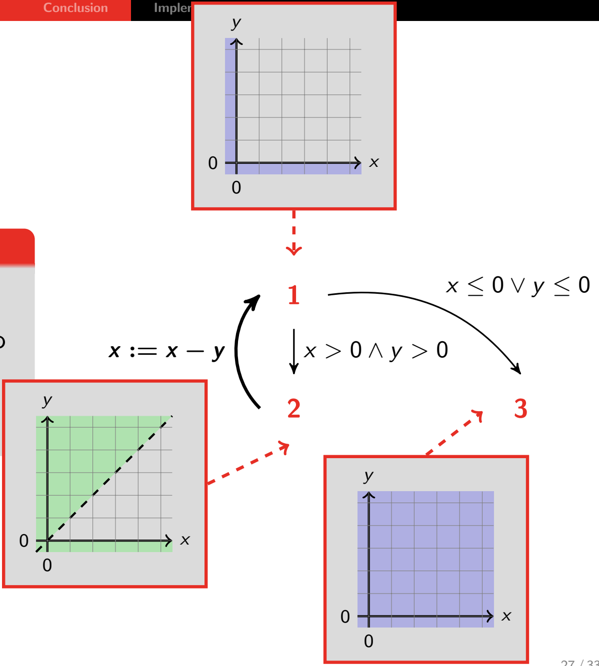
Example

int : x

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

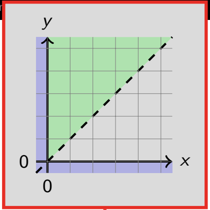
$^{2}x := x - y$

od^{3}



we consider the assignment $x := x - 1$ and we are at 2 steps to termination

we consider $x > 0$
and we do the join



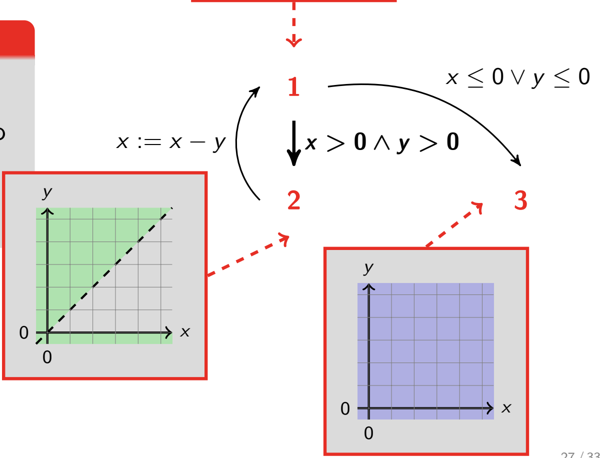
Example

int : x

while [1]$(x > 0 \wedge y > 0)$ do

[2]$x := x - y$

od[3]



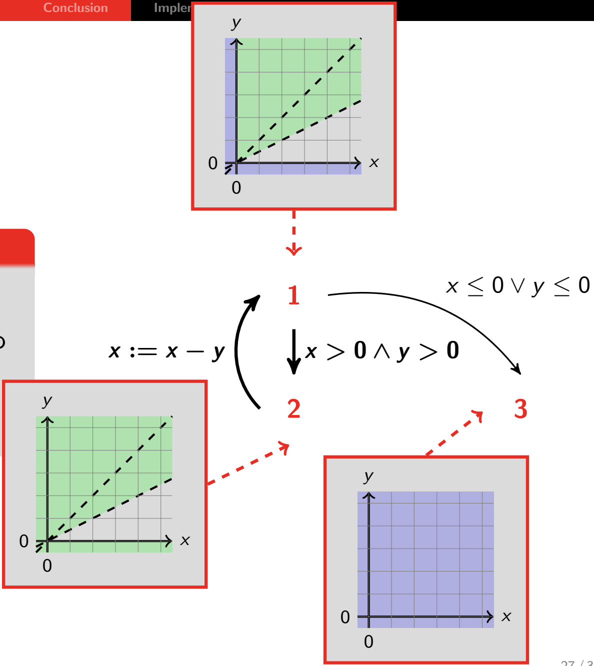
Example

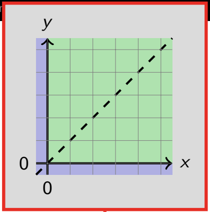
int : x

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

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

od^{3}





we do the widening

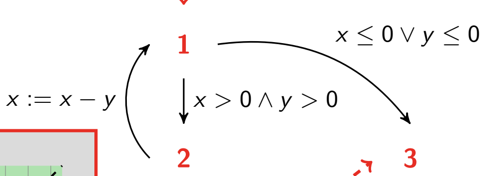
Example

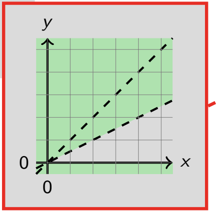
int : x

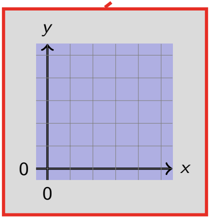
while [1]$(x > 0 \land y > 0)$ do

[2]$x := x - y$

od[3]







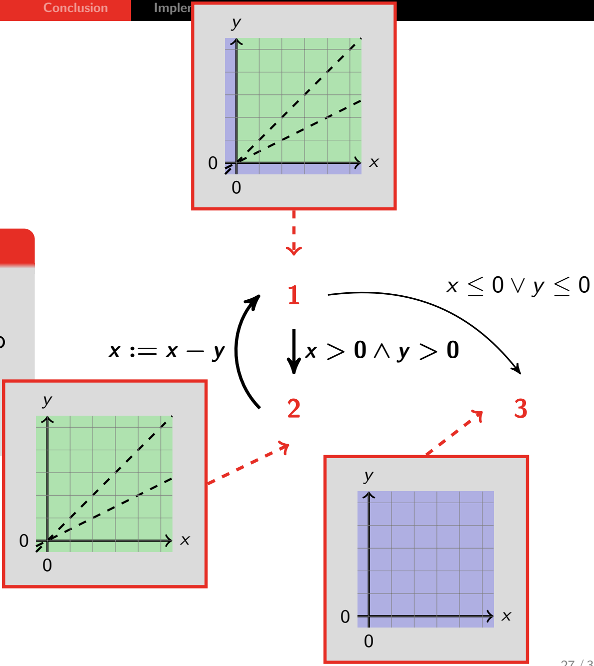
Example

int : x

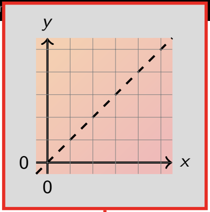
while $^{1}(x > 0 \wedge y > 0)$ do

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

od^{3}



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



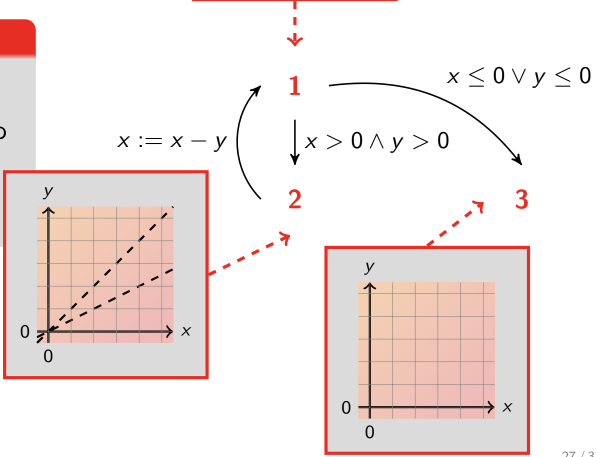
Example

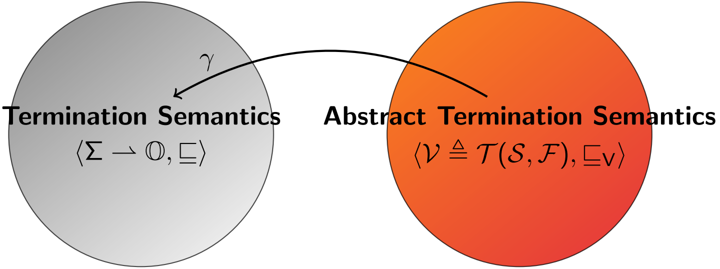
int : x

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

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

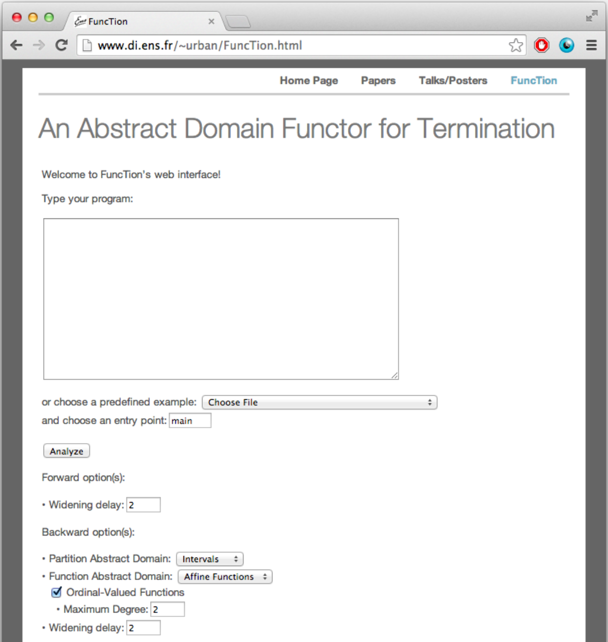
od^{3}



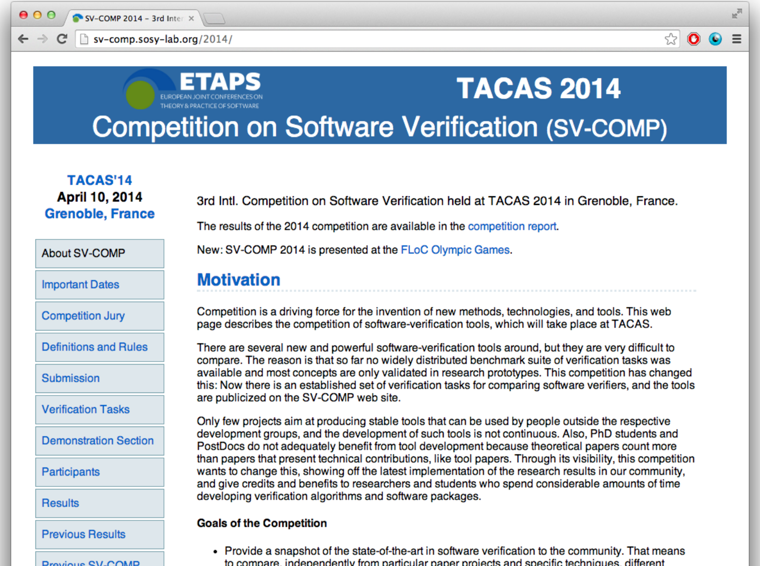


Theorem (Soundness)

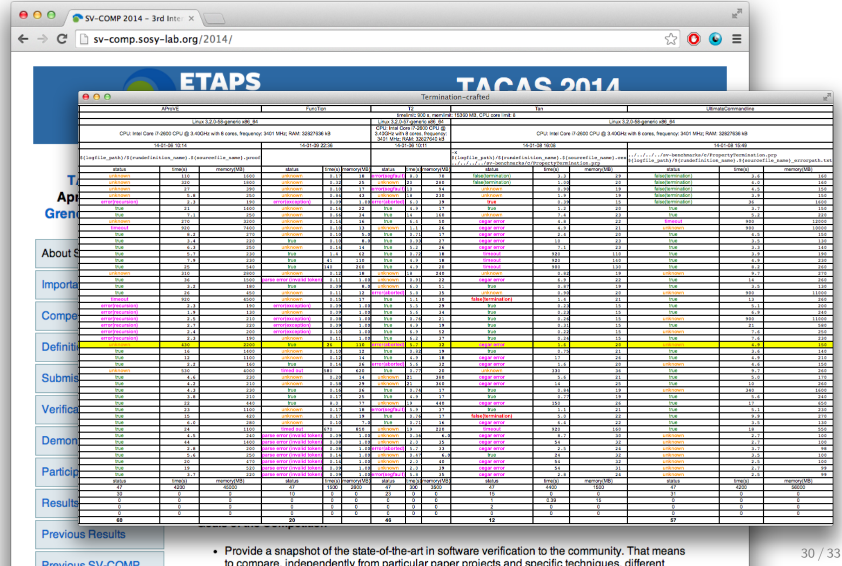
the abstract termination semantics is **sound**
to prove the termination of programs


FuncTion
www.di.ens.fr/~urban/FuncTion.html
Home Page
Papers
Talks/Posters
FuncTion
An Abstract Domain Functor for Termination
Welcome to FuncTion's web interface!
Type your program:
or choose a predefined example:
Choose File
and choose an entry point:
main
Analyze
Forward option(s):
• Widening delay:
2
Backward option(s):
• Partition Abstract Domain:
Intervals
• Function Abstract Domain:
Affine Functions
Ordinal-Valued Functions
• Maximum Degree:
2
• Widening delay:
2

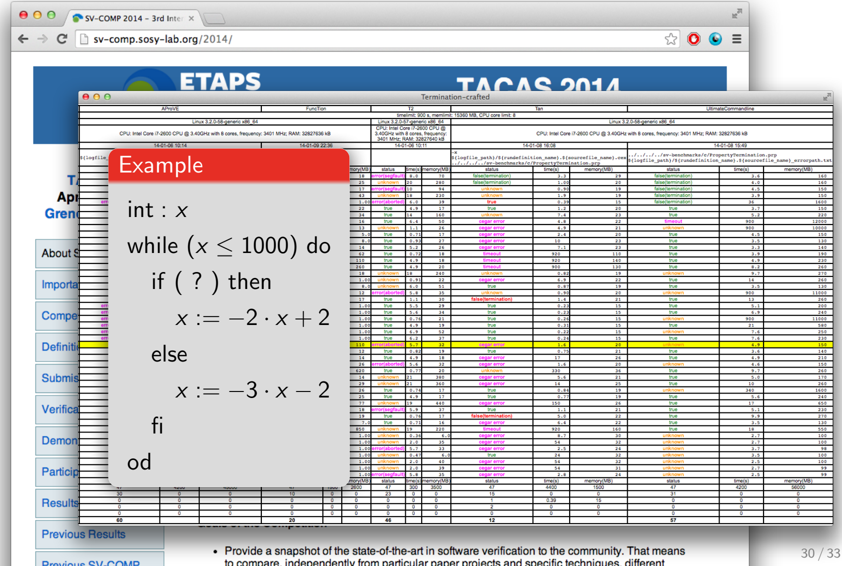
SV-COMP 2014


SV-COMP 2014 – 3rd Inter
sv-comp.sosy-lab.org/2014/
ETAPS
EUROPEAN JOINT CONFERENCES ON THEORY & PRACTICE OF SOFTWARE
TACAS 2014
Competition on Software Verification (SV-COMP)
TACAS'14
April 10, 2014
Grenoble, France
About SV-COMP
Important Dates
Competition Jury
Definitions and Rules
Submission
Verification Tasks
Demonstration Section
Participants
Results
Previous Results
3rd Intl. Competition on Software Verification held at TACAS 2014 in Grenoble, France.
The results of the 2014 competition are available in the competition report.
New: SV-COMP 2014 is presented at the FLoC Olympic Games.
Motivation
Competition is a driving force for the invention of new methods, technologies, and tools. This web page describes the competition of software-verification tools, which will take place at TACAS.
There are several new and powerful software-verification tools around, but they are very difficult to compare. The reason is that so far no widely distributed benchmark suite of verification tasks was available and most concepts are only validated in research prototypes. This competition has changed this: Now there is an established set of verification tasks for comparing software verifiers, and the tools are publicized on the SV-COMP web site.
Only few projects aim at producing stable tools that can be used by people outside the respective development groups, and the development of such tools is not continuous. Also, PhD students and PostDocs do not adequately benefit from tool development because theoretical papers count more than papers that present technical contributions, like tool papers. Through its visibility, this competition wants to change this, showing off the latest implementation of the research results in our community, and give credits and benefits to researchers and students who spend considerable amounts of time developing verification algorithms and software packages.
Goals of the Competition
Provide a snapshot of the state-of-the-art in software verification to the community. That means

SV-COMP 2014


SV-COMP 2014 – 3rd Inter
sv-comp.sosy-lab.org/2014/
ETAPS
TACAS 2014
Termination-crafted
AProVE
FuncTion
T2
Tan
UltimateCommandline
timelimit: 900 s, memlimit: 15360 MB, CPU core limit: 8
About
Importa
Compe
Definiti
Submis
Verifica
Demon
Particip
Results
Previous Results
Previous SV-COMP
• Provide a snapshot of the state-of-the-art in software verification to the community. That means
to compare, independently from particular paper projects and specific techniques, different

SV-COMP 2014



Example

int : x

while ($x \leq 1000$) do

  if (?) then

    $x := -2 \cdot x + 2$

  else

    $x := -3 \cdot x - 2$

  fi

od

Experiments

Benchmark: 87 terminating C programs collected from the literature

Tools:

- FuncTion
- AProVE
- T2
- Ultimate Büchi Automizer

Result:

	Tot	FuncTion	AProVE	T2	Ultimate	Time	Timeouts
FuncTion	51	–	8	8	3	6s	5
AProVE	60	17	–	7	2	35s	19
T2	73	30	20	–	3	2s	0
Ultimate	79	31	21	9	–	9s	1

Conclusions

- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - sufficient preconditions for termination
- instances based on **ordinal-valued functions**
  - lexicographic orders automatically inferred by the analysis
  - analysis not limited to programs with linear ranking functions
- instances based on **decision trees**

Future Work

- **more abstract domains**
  - non-linear ranking functions
  - better widening
- **fair termination**
- other **liveness** properties

Conclusions

- family of **abstract domains** for program termination
  - piecewise-defined ranking functions
  - sufficient preconditions for termination
- instances based on **ordinal-valued functions**
  - lexicographic orders automatically inferred by the analysis
  - analysis not limited to programs with linear ranking functions
- instances based on **decision trees**

Future Work

- **more abstract domains**
  - non-linear ranking functions
  - better widening
- **fair termination**
- other **liveness** properties

Thank You!