An Abstract Domain to Infer Ordinal-Valued Ranking Functions

”to infinity… and beyond!”

Caterina Urban and **Antoine Miné**

École Normale Supérieure & CNRS & INRIA
Paris, France

ESOP 2014
Grenoble, France

Outline

- **ranking functions**[1]
  - functions that strictly decrease at each program step. . .
  - . . . and that are bounded from below
- **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)
- family of **abstract domains** for program termination[2]
  - piecewise-defined ranking functions
- instances based on **ordinal-valued ranking functions**

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)

Outline

- **ranking functions**[1]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below
- **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)
- family of **abstract domains** for program termination[2]
  - piecewise-defined ranking functions
- instances based on **ordinal-valued ranking functions**

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)

Outline

- **ranking functions**[1]
  - functions that strictly decrease at each program step...
  - ...and that are bounded from below

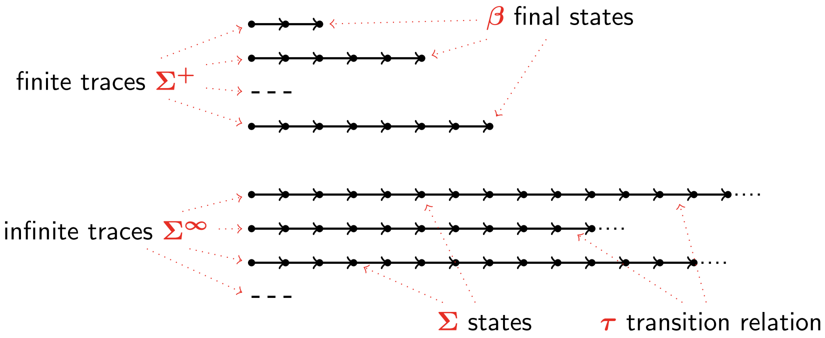
- **remark**: natural-valued ranking functions are not sufficient (e.g., programs with unbounded non-determinism)

- family of **abstract domains** for program termination[2]
  - piecewise-defined ranking functions
- instances based on **ordinal-valued ranking functions**

[1] Floyd - *Assigning Meanings to Programs* (1967)
[2] Urban - *The Abstract Domain of Segmented Ranking Functions* (SAS 2013)

Termination Semantics

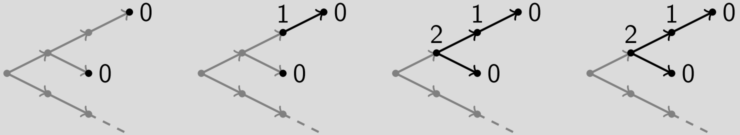
program $\mapsto$ **trace semantics**



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)

Example

int : x

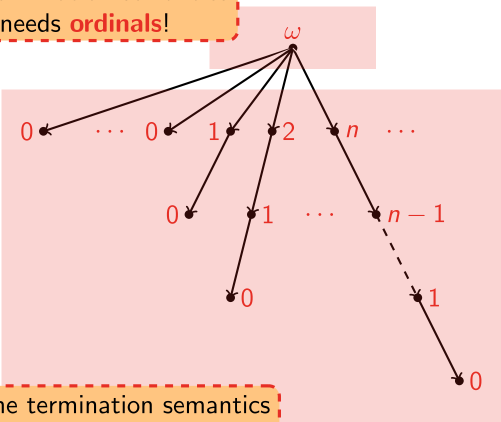
$x :=$?

while $(x > 0)$ do

  $x := x - 1$

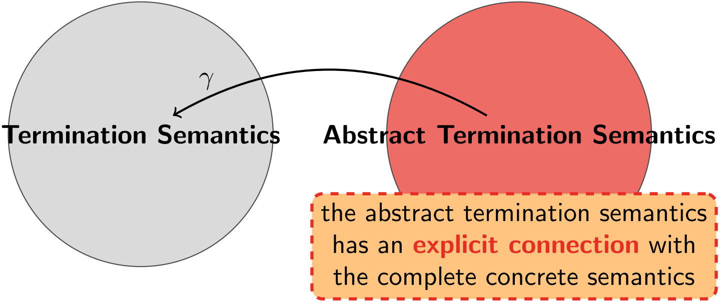
od

the termination semantics needs **ordinals**!



the termination semantics it is **not computable**!

Piecewise-Defined Ranking Functions



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

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

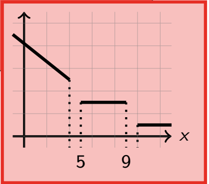
γ

Termination Semantics

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

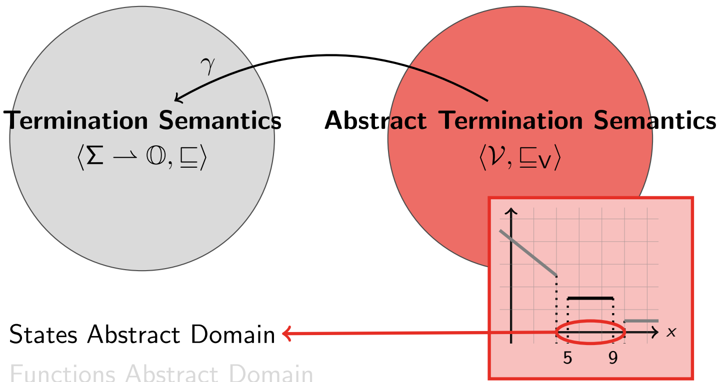
Abstract Termination Semantics

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



- 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)



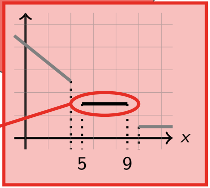
- States Abstract Domain — S
- Functions Abstract Domain — F
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, F)

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

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

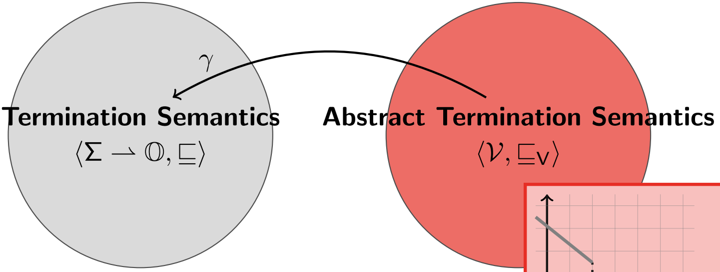
γ

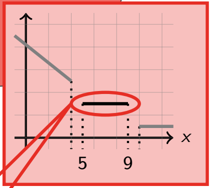
Abstract Termination Semantics
$\langle \mathcal{V}, \sqsubseteq_{\mathsf{V}} \rangle$



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

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

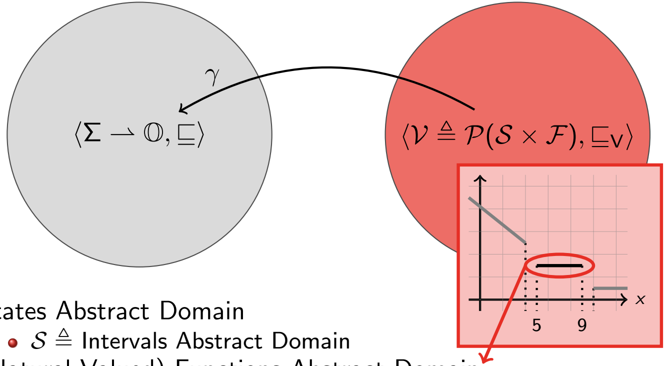




- States Abstract Domain — S
- **Natural-Valued** Functions Abstract Domain — F
- **Ordinal-Valued** Functions Abstract Domain — **O**(F)
- Piecewise-Defined Ranking Functions Abstract Domain — V(S, **O**(F))

Natural-Valued Ranking Functions

Affine Ranking Functions Abstract Domain



- States Abstract Domain
  - $\mathcal{S} \triangleq$ Intervals Abstract Domain
- (Natural-Valued) Functions Abstract Domain
  - $\mathcal{F} \triangleq \{\bot_\mathsf{F}\} \cup \{f \mid f \in \mathbb{Z}^n \to \mathbb{N}\} \cup \{\top_\mathsf{F}\}$
    where $f \equiv f(x_1, \ldots, x_n) = m_1 x_1 + \cdots + m_n x_n + q$
  - join $\sqcup_\mathsf{F}$, widening $\triangledown_\mathsf{F}$, backward assignments $\mathsf{ASSIGN}_\mathsf{F}$, ...

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

Example

int : x

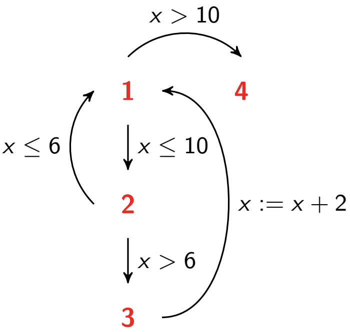
while [1]$(x \leq 10)$ do

  if [2]$(x > 6)$ then

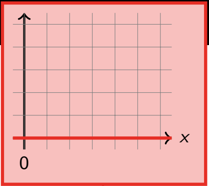
    [3]$x := x + 2$

  fi

od[4]



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



Example

int : x

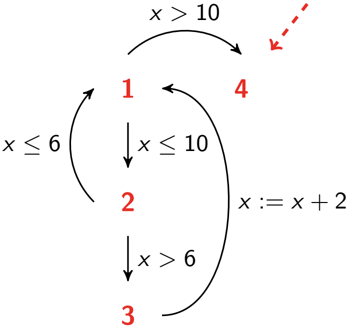
while ${}^{1}(x \leq 10)$ do

  if ${}^{2}(x > 6)$ 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

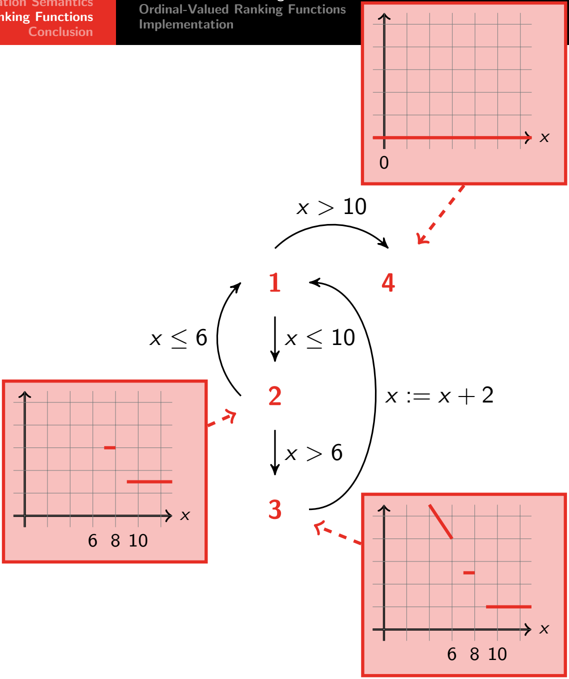
  if [2]$(x > 6)$ then

    [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

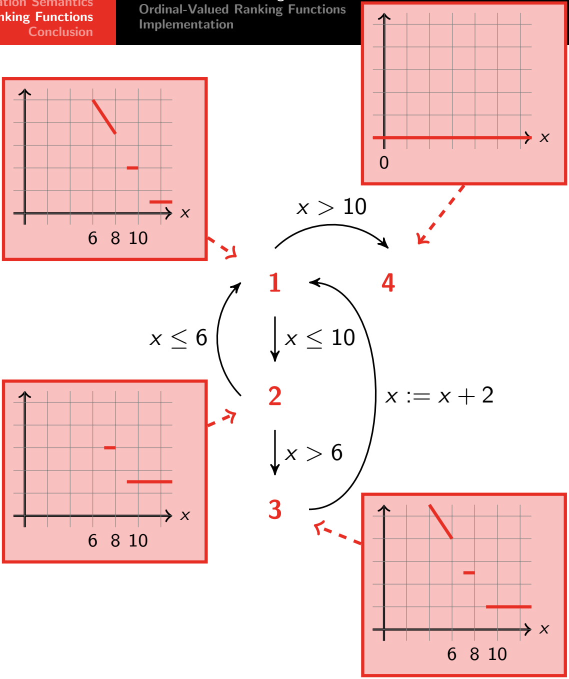
  if $^{2}(x > 6)$ then

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

  fi

od^{4}

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



Example

int : x

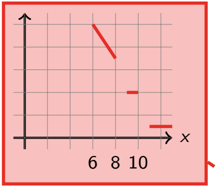
while $^{1}(x \leq 10)$ do

  if $^{2}(x > 6)$ then

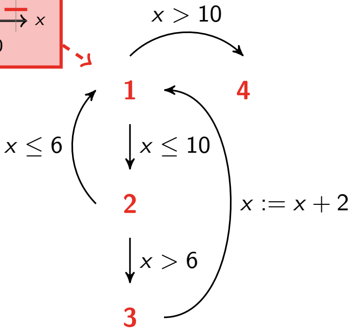
    $^{3}x := x + 2$

  fi

od^{4}



the analysis provides $x > 6$ as sufficient precondition for termination



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

Ordinal-Valued Ranking Functions

Ordinals

$$0,\ 1,\ 2,\ \ldots$$

$$\omega,\ \omega+1,\ \omega+2,\ \ldots$$

$$\omega\cdot 2,\ \omega\cdot 2+1,\ \omega\cdot 2+2,\ \ldots$$

$$\vdots$$

$$\omega^2,\ \ldots$$

$$\vdots$$

$$\omega^\omega,\ \ldots$$

$$\vdots$$

$$\epsilon_0,\ \ldots$$

$$\vdots$$

Ordinal Arithmetic

- **addition**

$$\alpha + 0 = \alpha \qquad \text{(zero case)}$$

$$\alpha + (\beta + 1) = (\alpha + \beta) + 1 \qquad \text{(successor case)}$$

$$\alpha + \beta = \bigcup_{\gamma < \beta} (\alpha + \gamma) \qquad \text{(limit case)}$$

  - associative: $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
  - not commutative: $1 + \omega = \omega \neq \omega + 1$
- **multiplication**

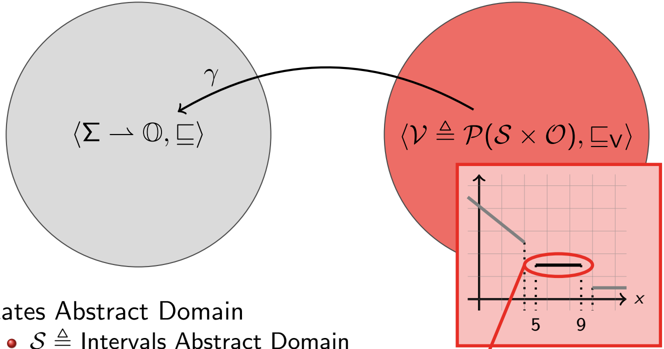
Ordinal Arithmetic

- **addition**
- **multiplication**

$$\alpha \cdot 0 = 0 \qquad \text{(zero case)}$$
$$\alpha \cdot (\beta + 1) = (\alpha \cdot \beta) + \alpha \qquad \text{(successor case)}$$
$$\alpha \cdot \beta = \bigcup_{\gamma < \beta} (\alpha \cdot \gamma) \qquad \text{(limit case)}$$

  - associative: $(\alpha \times \beta) \times \gamma = \alpha \times (\beta \times \gamma)$
  - left distributive: $\alpha \times (\beta + \gamma) = (\alpha \times \beta) + (\alpha \times \gamma)$
  - not commutative: $2 \times \omega = \omega \neq \omega \times 2$
  - not right distributive: $(\omega + 1) \times \omega = \omega \times \omega \neq \omega \times \omega + \omega$

Ordinal-Valued Ranking Functions Domain



- States Abstract Domain
  - $\mathcal{S} \triangleq$ Intervals Abstract Domain
- Natural-Valued Functions Abstract Domain
  - $\mathcal{F} \triangleq$ Affine Ranking Functions Abstract Domain
- Ordinal-Valued Functions Abstract Domain
  - $\mathcal{O} \triangleq \{\bot_O\} \cup \{\sum_i \omega^i \cdot f_i \mid f_i \in \mathcal{F} \setminus \{\bot_F, \top_F\}\} \cup \{\top_O\}$

Backward Assignments: $\mathrm{ASSIGN}_{\mathrm{O}}$

- backward assignments amount to variable substitution

Example

$$o \triangleq \omega \cdot (x_1 - x_2) + x_1$$

$$\Downarrow \quad \mathbf{x_1 := x_1 + x_2}$$

$$o \triangleq ?$$

Backward Assignments: $\mathrm{ASSIGN}_{\mathrm{O}}$

- backward assignments amount to variable substitution

Example

$$o \triangleq \omega \cdot (x_1 - x_2) + x_1$$

$$\Downarrow \quad \mathbf{x_1 := x_1 + x_2}$$

$$o \triangleq \qquad + \; \mathbf{1}$$

Backward Assignments: $\mathrm{ASSIGN}_{\mathrm{O}}$

- backward assignments amount to variable substitution

Example

$$o \triangleq \omega \cdot (x_1 - x_2) + x_1$$

$$\Downarrow \quad \mathbf{x_1 := x_1 + x_2}$$

$$o \triangleq \omega \cdot (\mathbf{x_1 + x_2 - x_2}) + \mathbf{x_1 + x_2} + 1$$

Backward Assignments: $\mathrm{ASSIGN}_{\mathrm{O}}$

- backward assignments amount to variable substitution

Example

$$o \triangleq \omega \cdot (x_1 - x_2) + x_1$$

$$\Downarrow \quad \mathbf{x_1 := x_1 + x_2}$$

$$o \triangleq \omega \cdot x_1 + x_1 + x_2 + 1$$

Backward Non-Deterministic Assignments

- $\mathrm{ASSIGN}_\mathsf{F}$ in ascending powers of ω

Example

$$o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$o \triangleq ?$$

Backward Non-Deterministic Assignments

- ASSIGN_F in ascending powers of ω

Example

$$o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$o \triangleq + \mathbf{1}$$

Backward Non-Deterministic Assignments

- $\mathrm{ASSIGN}_\mathsf{F}$ in ascending powers of ω

Example

$$o \triangleq \omega \cdot x_1 + \mathbf{x_2}$$

$$\Downarrow \quad \mathbf{x_1} := \mathbf{?}$$

$$o \triangleq + \mathbf{x_2} + 1$$

Backward Non-Deterministic Assignments

- $\mathrm{ASSIGN}_{\mathsf{F}}$ in ascending powers of ω

Example

$$o \triangleq \omega \cdot \mathbf{x_1} + x_2$$

$$\Downarrow \quad \mathbf{x_1} := \ ?$$

$$o \triangleq \omega^2 \cdot \mathbf{0}\ {}^{1} + \omega \cdot \mathbf{0} + x_2 + 1$$

$\omega^k \cdot \omega = \omega^{k+1} \cdot 1 + \omega^k \times 0 = \omega^{k+1}$

Backward Non-Deterministic Assignments

- $\mathrm{ASSIGN}_\mathsf{F}$ in ascending powers of ω

Example

$$o \triangleq \omega \cdot x_1 + x_2$$

$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$o \triangleq \omega^2 \cdot 1 + \omega \cdot 0 + x_2 + 1$$

Backward Non-Deterministic Assignments

- $\mathrm{ASSIGN}_{\mathsf{F}}$ in ascending powers of ω

Example

$$o \triangleq \omega \cdot x_1 + x_2$$

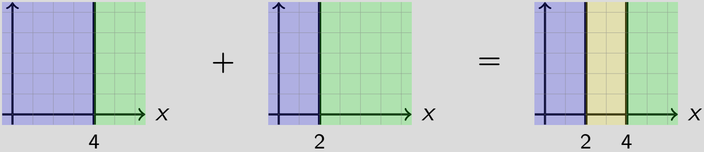
$$\Downarrow \quad \mathbf{x_1 := ?}$$

$$o \triangleq \omega^2 + x_2 + 1$$

Join: $\sqcup_O$

- segmentation unification

Example



- join: $\sqcup_O$

Join: $\sqcup_O$

- segmentation unification
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of ω

Example

$$
\begin{array}{ccccccccccc}
o_1 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & x_2 & + & 3 \\
o_2 & \triangleq & \omega^2 & \cdot & (x_1 - 1) & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
o_1 \sqcup_O o_2 & \triangleq & ? & & & & & & & &
\end{array}
$$

Join: $\sqcup_O$

- segmentation unification
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of ω

Example

$$
\begin{array}{ccccccccccc}
o_1 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & x_2 & + & 3 \\
o_2 & \triangleq & \omega^2 & \cdot & (x_1 - 1) & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
o_1 \sqcup_O o_2 & \triangleq & & & & & & & & + & 4
\end{array}
$$

Join: $\sqcup_O$

- segmentation unification
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of ω

Example

$$
\begin{array}{cccccccccc}
o_1 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & \mathbf{x_2} & + & 3 \\
o_2 & \triangleq & \omega^2 & \cdot & (x_1 - 1) & + & \omega & \cdot & (-\mathbf{x_2}) & + & 4 \\
\hline
o_1 \sqcup_O o_2 & \triangleq & & & \mathbf{1} & + & \omega & \cdot & \mathbf{0} & + & 4
\end{array}
$$

$\omega^k \cdot \omega = \omega^{k+1} \cdot 1 + \omega^k \times 0 = \omega^{k+1}$

Join: $\sqcup_O$

- segmentation unification
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of ω

Example

$$
\begin{array}{ccccccccccc}
o_1 & \triangleq & \omega^2 & \cdot & \mathbf{x_1} & & + & \omega & \cdot & x_2 & + & 3 \\
o_2 & \triangleq & \omega^2 & \cdot & \mathbf{(x_1 - 1)} & & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
o_1 \sqcup_O o_2 & \triangleq & \omega^2 & \cdot & \mathbf{x_1} & \mathbf{1} & + & \omega & \cdot & \mathbf{0} & + & 4
\end{array}
$$

Join: $\sqcup_O$

- segmentation unification
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of ω

Example

$$
\begin{array}{ccccccccccc}
o_1 & \triangleq & \omega^2 & \cdot & \mathbf{x_1} & + & \omega & \cdot & x_2 & + & 3 \\
o_2 & \triangleq & \omega^2 & \cdot & (\mathbf{x_1 - 1}) & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
o_1 \sqcup_O o_2 & \triangleq & \omega^2 & \cdot & (\mathbf{x_1 + 1}) & + & \omega & \cdot & \mathbf{0} & + & 4
\end{array}
$$

Join: $\sqcup_O$

- segmentation unification
- join: $\sqcup_O$
  - $\sqcup_F$ in ascending powers of ω

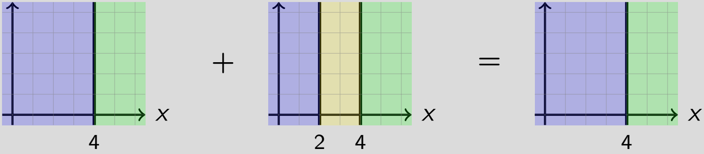
Example

$$
\begin{array}{cccccccccccc}
o_1 & \triangleq & \omega^2 & \cdot & x_1 & + & \omega & \cdot & x_2 & + & 3 \\
o_2 & \triangleq & \omega^2 & \cdot & (x_1 - 1) & + & \omega & \cdot & (-x_2) & + & 4 \\
\hline
o_1 \sqcup_O o_2 & \triangleq & \omega^2 & \cdot & (x_1 + 1) & & & & & + & 4
\end{array}
$$

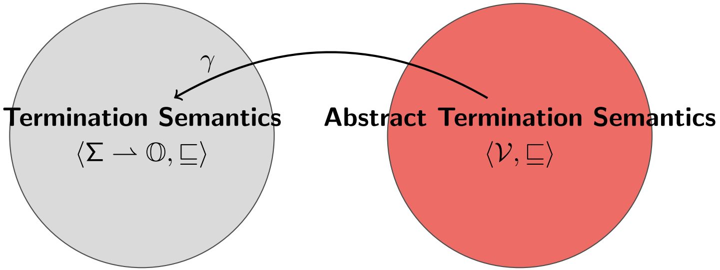
Widening: ∇_O

- segmentation left-unification

Example



- unstable ranking functions yield $\top_O$



Theorem (Soundness)

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

Examples

Example

```
int : x1, x2
while 1(x1 > 0 ∧ x2 > 0) do
    if 2( ? ) then
        3x1 := x1 − 1
        4x2 := ?
    else
        5x2 := x2 − 1
od6
```

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

Example

```
int : x1, x2
while 1(x1 ≠ 0 ∧ x2 > 0) do
   if 2(x1 > 0) then
      if 3( ? ) then
         4x1 := x1 − 1
         5x2 := ?
      else
         6x2 := x2 − 1
   else /* x1 < 0 */
      if 7( ? ) then
         8x1 := x1 + 1
      else
         9x2 := x2 − 1
         10x1 := ?
od11
```

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

Example

```
int : x1, x2
while 1(x1 ≠ 0 ∧ x2 > 0) do
    if 2(x1 > 0) then
        if 3( ? ) then
            4x1 := x1 − 1
            5x2 := ?
        else
            6x2 := x2 − 1
    else /* x1 < 0 */
        if 7( ? ) then
            8x1 := x1 + 1
        else
            9x2 := x2 − 1
            10x1 := ?
```

the coefficients and their **order** are **automatically 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}

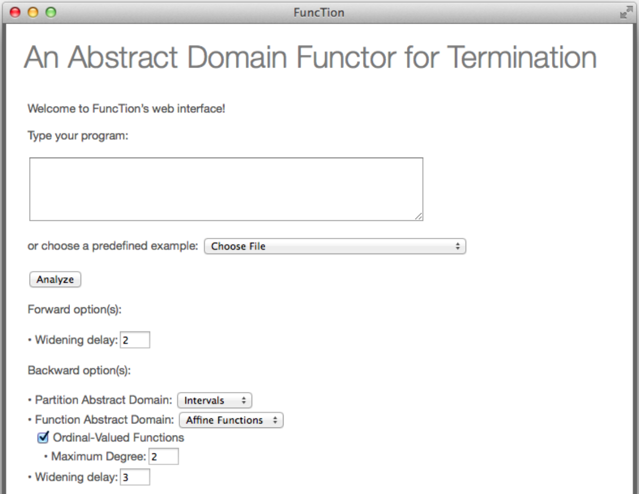
$$f_1(x_1, x_2, N) = \begin{cases} 1 & x_1 < 0 \\ \omega \cdot (x_1 + 1) + 6x_1 + 7 & x_1 \geq 0 \end{cases}$$

the loop terminates in a **finite number of iterations**

Implementation

`http://www.di.ens.fr/~urban/FuncTion.html`

- written in OCaml



Experiments

Benchmarks: 38 programs (collected from publications on termination)

- 25 always terminating programs
- 13 conditionally terminating programs
- 9 simple loops
- 7 nested loops
- 13 non-deterministic programs

Results: proved 30 out of 38 programs

- proved 8 out of 9 simple loops
- proved 4 out of 7 nested loops
  - ordinals required for 2 out of 4
- proved 10 out of 13 non-deterministic programs
  - ordinals required for 5 out of 10

SV-COMP 2014

SV-COMP 2014 – 3rd International Competition on Software Verification

sv-comp.sosy-lab.org/2014/index.php

ETAPS
EUROPEAN JOINT CONFERENCES ON THEORY & PRACTICE OF SOFTWARE

TACAS 2014

Competition on Software Verification (SV-COMP)

TACAS'14
April, 2014
Grenoble, France

- About SV-COMP
- Important Dates
- Competition Jury
- Definitions and Rules
- Submission
- Verification Tasks
- Demonstration Section
- Participants
- Results
- Previous Results
- Previous SV-COMP

3rd Intl. Competition on Software Verification held at TACAS 2014 in Grenoble, France.

The results of the 2013 competition are available in the competition report.

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.

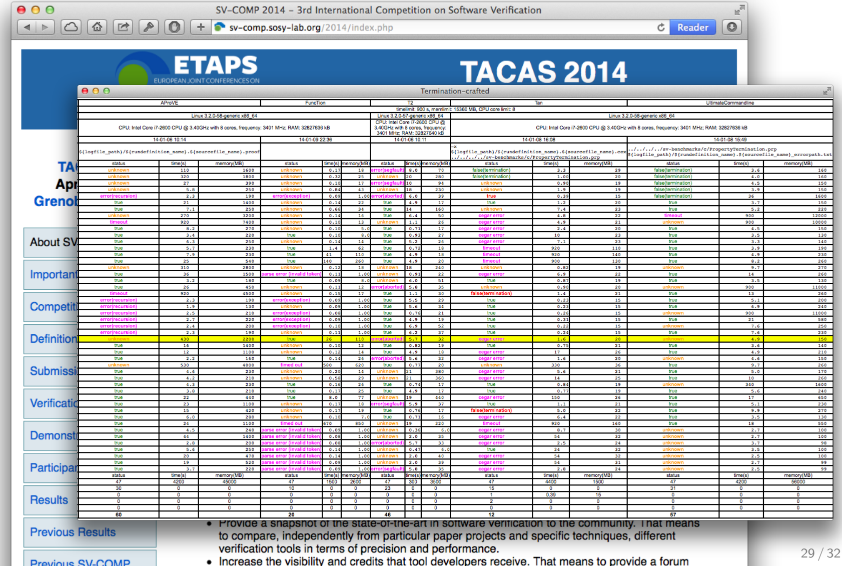
There are several new and powerful software-verification tools around, but they are very difficult to compare. The reason is that so far no widely distributed benchmark suite of verification tasks was available and most concepts are only validated in research prototypes. This competition has changed 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 to compare, independently from particular paper projects and specific techniques, different verification tools in terms of precision and performance.
- Increase the visibility and credits that tool developers receive. That means to provide a forum

SV-COMP 2014



AProVE			FuncTion			T2			Tan			UltimateCommandline		
timelimit: 900 s, memlimit: 15360 MB, CPU core limit: 8														
Linux 3.2.0-58-generic x86_64						Linux 3.2.0-57-generic x86_64			Linux 3.2.0-58-generic x86_64					
CPU: Intel Core i7-2600 CPU @ 3.40GHz with 8 cores, frequency: 3401 MHz; RAM: 32827636 kB						CPU: Intel Core i7-2600 CPU @ 3.40GHz with 8 cores, frequency: 3401 MHz; RAM: 32827640 kB			CPU: Intel Core i7-2600 CPU @ 3.40GHz with 8 cores, frequency: 3401 MHz; RAM: 32827636 kB					
14-01-06 10:14			14-01-09 22:36			14-01-06 10:11			14-01-08 16:08			14-01-08 15:49		
${logfile_path}/${rundefinition_name}.${sourcefile_name}.proof									-x ${logfile_path}/${rundefinition_name}.${sourcefile_name}.cex ../../../../sv-benchmarks/c/PropertyTermination.prp			../../../../sv-benchmarks/c/PropertyTermination.prp ${logfile_path}/${rundefinition_name}.${sourcefile_name}_errorpath.txt		
status	time(s)	memory(MB)	status	time(s)	memory(MB)	status	time(s)	memory(MB)	status	time(s)	memory(MB)	status	time(s)	memory(MB)
unknown	110	1600	unknown	0.17	18	error(segfault)	8.0	70	false(termination)	3.3	29	false(termination)	3.6	160
unknown	320	1800	unknown	0.32	25	unknown	20	280	false(termination)	1.00	20	false(termination)	4.0	160
unknown	27	390	unknown	0.10	17	error(segfault)	10	94	unknown	0.90	19	false(termination)	4.5	150
unknown	5.8	250	unknown	0.84	43	unknown	18	230	unknown	1.9	19	false(termination)	3.9	150
error(recursion)	2.3	190	error(exception)	0.09	1.00	error(aborted)	6.0	39	true	0.39	15	false(termination)	36	1600
true	21	1400	unknown	0.14	22	true	4.9	17	true	1.2	20	true	3.7	150
true	7.1	250	unknown	0.66	34	true	14	160	unknown	7.4	23	true	5.2	220
unknown	270	3200	unknown	0.14	16	true	6.4	50	cegar error	4.8	22	timeout	900	12000
timeout	920	7400	unknown	0.10	13	unknown	1.1	26	cegar error	4.9	21	unknown	900	10000
true	8.2	270	unknown	0.10	5.0	true	0.71	17	cegar error	2.4	20	true	4.5	150
true	3.4	220	true	0.10	8.0	true	0.93	27	cegar error	10	23	true	3.5	130
true	6.3	250	unknown	0.14	14	true	5.2	26	cegar error	7.1	23	true	3.3	140
true	5.7	230	true	1.4	62	true	0.72	18	timeout	920	110	true	3.9	190
true	7.9	230	true	41	110	true	4.9	18	timeout	920	140	true	4.9	230
true	25	540	true	140	260	true	4.9	20	timeout	900	130	true	8.2	260
unknown	310	2800	unknown	0.12	18	unknown	18	240	unknown	0.82	19	unknown	9.7	270
true	36	1500	parse error (invalid token)	0.11	1.00	unknown	0.91	22	cegar error	6.9	22	true	14	260
true	3.2	180	true	0.09	8.0	unknown	6.0	51	true	0.87	19	true	3.5	130
true	26	450	unknown	0.11	12	error(aborted)	5.8	35	unknown	0.90	20	unknown	900	11000
timeout	920	4500	unknown	0.15	17	true	1.1	30	false(termination)	1.4	21	true	13	260
error(recursion)	2.3	190	error(exception)	0.09	1.00	true	5.5	29	true	0.23	15	true	5.1	200
error(recursion)	1.9	130	unknown	0.09	1.00	true	5.6	34	true	0.23	15	true	6.9	240
error(recursion)	2.5	210	error(exception)	0.08	1.00	true	0.76	21	true	0.26	15	unknown	900	11000
error(recursion)	2.7	220	error(exception)	0.09	1.00	true	4.9	19	true	0.31	15	true	21	580
error(recursion)	2.4	200	error(exception)	0.10	1.00	true	6.9	52	true	0.22	15	unknown	7.6	250
error(recursion)	2.3	190	unknown	0.11	1.00	true	6.2	37	true	0.24	15	true	7.6	230
unknown	430	2200	true	26	110	error(aborted)	5.7	32	cegar error	1.6	20	unknown	4.9	150
true	16	1400	unknown	0.10	12	true	0.82	19	true	0.75	21	true	3.6	140
true	12	1100	unknown	0.12	14	true	4.9	18	cegar error	17	26	true	4.9	210
true	2.2	160	true	0.14	26	error(aborted)	5.6	32	cegar error	1.6	20	unknown	4.6	150
unknown	530	4000	timed out	580	620	true	0.77	20	unknown	330	36	true	9.7	260
true	4.6	230	unknown	0.20	14	unknown	21	380	cegar error	5.6	21	true	5.0	170
true	4.2	210	unknown	0.58	29	unknown	21	360	cegar error	14	25	true	10	260
true	4.3	230	true	0.16	26	true	0.74	17	true	0.84	19	unknown	340	1600
true	3.8	210	true	0.17	25	true	4.9	17	true	0.77	19	true	5.6	240
true	22	440	true	8.0	77	unknown	19	440	cegar error	150	26	true	17	650
true	23	1100	unknown	0.17	18	error(segfault)	5.9	37	true	1.1	21	true	5.1	230
true	15	420	unknown	0.17	19	true	0.76	17	false(termination)	5.0	22	true	9.9	270
true	6.0	280	unknown	0.10	7.0	true	0.71	16	cegar error	6.4	22	true	3.5	130
true	24	1100	timed out	670	850	unknown	19	220	timeout	920	160	true	18	550
true	4.5	240	parse error (invalid token)	0.09	1.00	unknown	0.36	6.0	cegar error	8.7	30	unknown	2.7	100
true	44	1400	parse error (invalid token)	0.08	1.00	unknown	2.0	35	cegar error	54	32	unknown	2.7	100
true	2.8	200	parse error (invalid token)	0.08	1.00	error(aborted)	5.7	33	cegar error	2.5	24	unknown	3.7	98
true	5.6	250	parse error (invalid token)	0.14	1.00	unknown	0.47	6.0	true	24	32	unknown	3.5	100
true	20	470	parse error (invalid token)	0.14	1.00	unknown	2.0	40	cegar error	54	32	unknown	2.5	100
true	19	520	parse error (invalid token)	0.09	1.00	unknown	2.0	39	cegar error	54	31	unknown	2.7	99
true	3.7	220	parse error (invalid token)	0.09	1.00	error(segfault)	5.8	35	cegar error	2.8	24	unknown	2.5	99
status	time(s)	memory(MB)	status	time(s)	memory(MB)	status	time(s)	memory(MB)	status	time(s)	memory(MB)	status	time(s)	memory(MB)
47	4200	45000	47	1500	2600	47	300	3500	47	4400	1500	47	4200	56000
30	0	0	10	0	0	23	0	0	15	0	0	31	0	0
0	0	0	0	0	0	0	0	0	1	0.39	15	0	0	0
0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60			20			46			12			57		

- 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 verification tools in terms of precision and performance.
- Increase the visibility and credits that tool developers receive. That means to provide a forum

Related Work

- Ordinal-Valued and Lexicographic Ranking Functions

$$\omega^k \cdot \underbrace{f_k}_{\in\mathbb{N}} + \ldots + \omega^2 \cdot \underbrace{f_2}_{\in\mathbb{N}} + \omega \cdot \underbrace{f_1}_{\in\mathbb{N}} + \underbrace{f_0}_{\in\mathbb{N}} \in \mathbb{O}$$

$$\iff (f_k, \ldots, f_2, f_1, f_0) \in \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{k}$$

  - Lee & Jones & Ben-Amram - The Size-Change Principle for Program Termination (POPL 2001)
  - Alias & Darte & Feautrier & Gonnord - Multi-Dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs (SAS 2010)
  - Cook & See & Zuleger - Ramsey vs. Lexicographic Termination Proving (TACAS 2013)

- **Ordinal-Valued + Piecewise-Defined** Ranking Functions $\longrightarrow$ **Conditional Termination**

Related Work

- Ordinal-Valued and Lexicographic Ranking Functions

$$\omega^k \cdot \underbrace{f_k}_{\in\mathbb{N}} + \ldots + \omega^2 \cdot \underbrace{f_2}_{\in\mathbb{N}} + \omega \cdot \underbrace{f_1}_{\in\mathbb{N}} + \underbrace{f_0}_{\in\mathbb{N}} \in \mathbb{O}$$

$$\Longleftrightarrow (f_k\ ,\ \ldots\ ,\ f_2\ ,\ f_1\ ,\ f_0) \in \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{k}$$

  - Lee & Jones & Ben-Amram - The Size-Change Principle for Program Termination (POPL 2001)
  - Alias & Darte & Feautrier & Gonnord - Multi-Dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs (SAS 2010)
  - Cook & See & Zuleger - Ramsey vs. Lexicographic Termination Proving (TACAS 2013)

- **Ordinal-Valued + Piecewise-Defined** Ranking Functions
  $\longrightarrow$ **Conditional Termination**

Related Work

- Ordinal-Valued and Lexicographic Ranking Functions

$$\omega^k \cdot \underbrace{f_k}_{\in \mathbb{N}} + \ldots + \omega^2 \cdot \underbrace{f_2}_{\in \mathbb{N}} + \omega \cdot \underbrace{f_1}_{\in \mathbb{N}} + \underbrace{f_0}_{\in \mathbb{N}} \in \mathbb{O}$$

$$\Longleftrightarrow (f_k , \ldots , f_2 , f_1 , f_0) \in \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{k}$$

  - Lee & Jones & Ben-Amram - The Size-Change Principle for Program Termination (POPL 2001)
  - Alias & Darte & Feautrier & Gonnord - Multi-Dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs (SAS 2010)
  - Cook & See & Zuleger - Ramsey vs. Lexicographic Termination Proving (TACAS 2013)

- **Ordinal-Valued + Piecewise-Defined** Ranking Functions

  $\longrightarrow$ Conditional Termination

Related Work

- Ordinal-Valued and Lexicographic Ranking Functions

$$\omega^k \cdot \underbrace{f_k}_{\in\mathbb{N}} + \ldots + \omega^2 \cdot \underbrace{f_2}_{\in\mathbb{N}} + \omega \cdot \underbrace{f_1}_{\in\mathbb{N}} + \underbrace{f_0}_{\in\mathbb{N}} \in \mathbb{O}$$

$$\Longleftrightarrow (f_k , \ldots , f_2 , f_1 , f_0) \in \underbrace{\mathbb{N} \times \ldots \times \mathbb{N}}_{k}$$

  - Lee & Jones & Ben-Amram - The Size-Change Principle for Program Termination (POPL 2001)
  - Alias & Darte & Feautrier & Gonnord - Multi-Dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs (SAS 2010)
  - Cook & See & Zuleger - Ramsey vs. Lexicographic Termination Proving (TACAS 2013)

- **Ordinal-Valued + Piecewise-Defined** Ranking Functions
  $\longrightarrow$ **Conditional Termination**

Conclusions

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

Future Work

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

Conclusions

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

Future Work

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

Thank You!