Abstract Interpretation with Frama-C

EJCP 2021

Virgile Prevosto
virgile.prevosto@cea.fr

Université Paris-Saclay, CEA, List

June 15th, 2021

Introduction

- Software is more and more pervasive in embedded systems...
- ...and keeps getting larger
- Tests and code review too costly beyond a certain size and coverage criterion
- Need for correct tools
  - ✔ Detect all potential issues
  - ✘ May issue spurious warnings
  - ✘ Impossible for an automated tool to warn for all real issues and only for them (Rice theorem)

Introduction

- Software is more and more pervasive in embedded systems...
- ...and keeps getting larger
- Tests and code review too costly beyond a certain size and coverage criterion
- Need for correct tools
  - ✔ Detect all potential issues
  - ✘ May issue spurious warnings
  - ✘ Impossible for an automated tool to warn for all real issues and only for them (Rice theorem)

Summary

Context

Abstract Domains

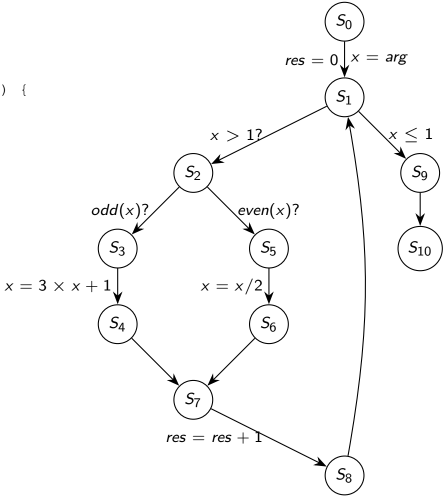
Termination

Improving precision

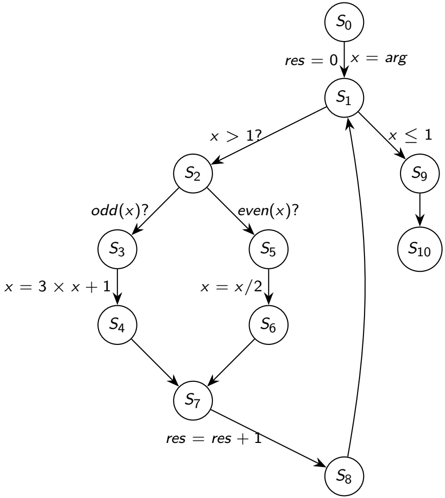
Setting Analysis Context

Control-Flow Graph

```
mpz_ptr syracuse(mpz_t res, const mpz_t arg) {
  mpz_t x;
  mpz_init_set_ui(res,0UL);
  mpz_init_set(x,arg);
  while (mpz_cmp_ui(x,1UL)>0) {
    mpz_out_str(stdout,10,x);
    putchar('\n');
    if (mpz_odd_p(x)) {
      mpz_mul_ui(x,x,3UL);
      mpz_add_ui(x,x,1UL);
    } else {
      mpz_cdiv_q_ui(x,x,2UL);
    }
    mpz_add_ui(res,res,1UL);
  }
  mpz_clear(x);
  return res;
}
```

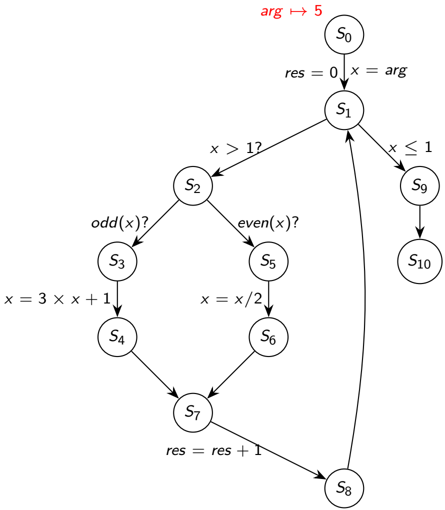


Trace Semantics



- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

Trace Semantics

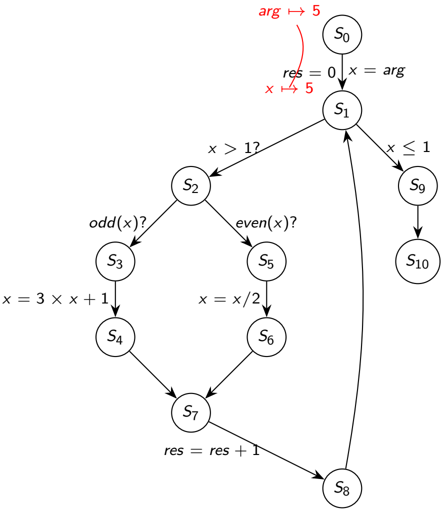


s_0

$arg \mapsto 5$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

Trace Semantics

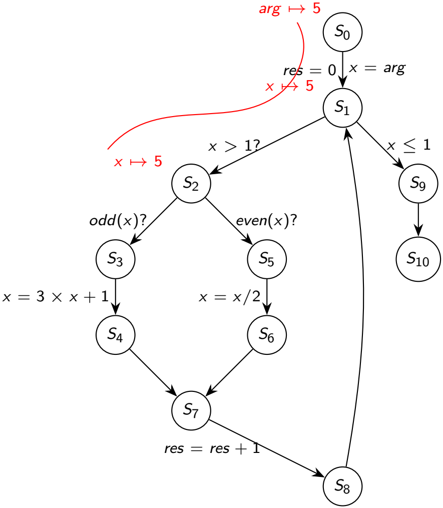


$$\overset{s_0}{arg \mapsto 5} \Longrightarrow \overset{s_1}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

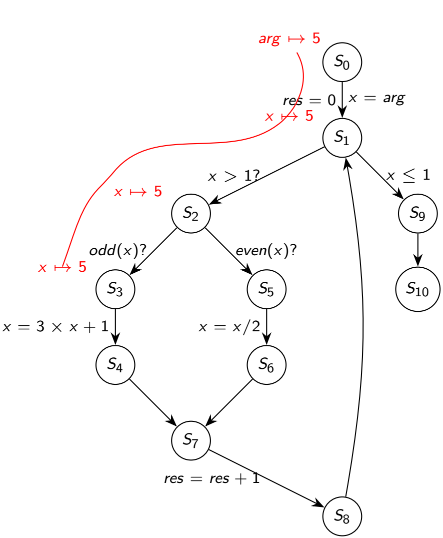
Trace Semantics

$$\overset{s_0}{arg \mapsto 5} \Longrightarrow \overset{s_1}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}} \Longrightarrow \overset{s_2}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}}$$



- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

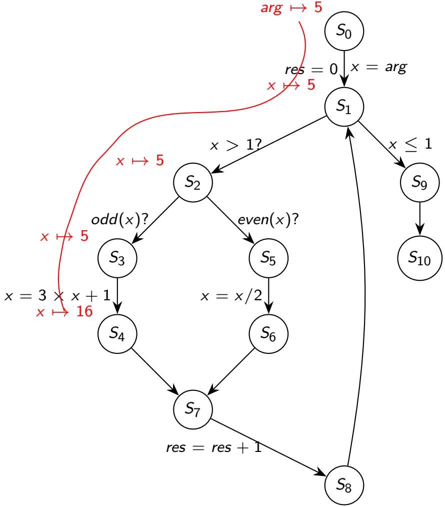
Trace Semantics



$$\overset{s_0}{arg \mapsto 5} \Longrightarrow \overset{s_1}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}} \Longrightarrow \overset{s_2}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}} \Longrightarrow \overset{s_3}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

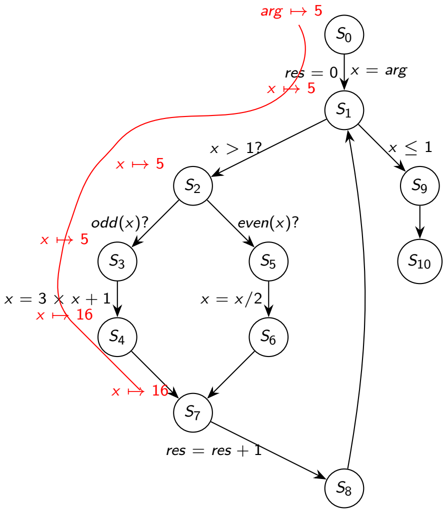
Trace Semantics



$$\begin{tikzcd}
\begin{array}{c} s_0 \\ arg \mapsto 5 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[d, Rightarrow] \\
 & & & \begin{array}{c} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{array}
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

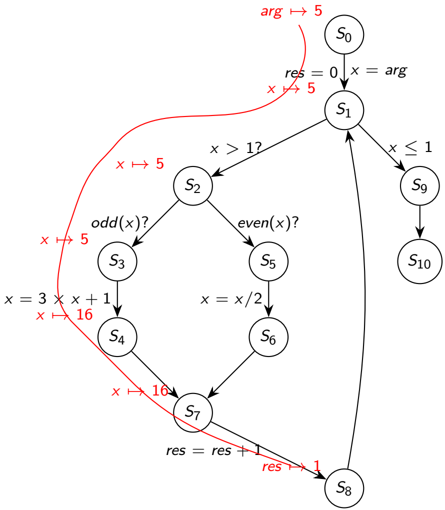
Trace Semantics



$$\begin{tikzcd}
\begin{matrix} s_0 \\ arg \mapsto 5 \\ \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{matrix} \arrow[d, Rightarrow] \\
& & \begin{matrix} s_7 \\ x \mapsto 16 \\ res \mapsto 0 \end{matrix} & \begin{matrix} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{matrix} \arrow[l, Rightarrow]
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

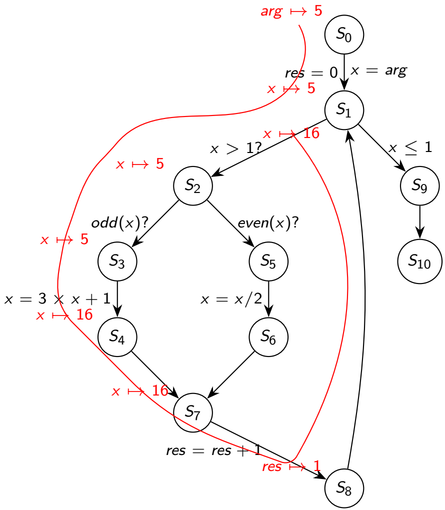
Trace Semantics



$$\begin{tikzcd}
\begin{array}{c} s_0 \\ arg \mapsto 5 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[d, Rightarrow] \\
& \begin{array}{c} s_8 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} & \begin{array}{c} s_7 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow]
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

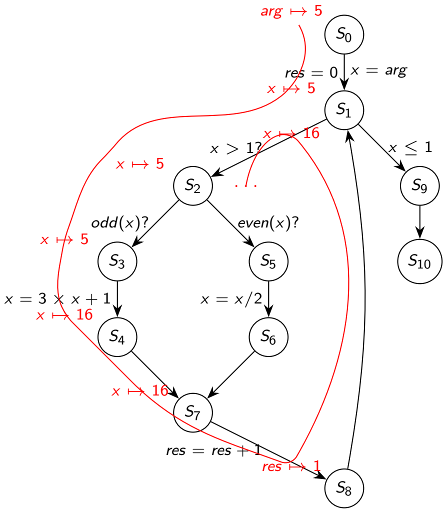
Trace Semantics



$$\begin{tikzcd}
\overset{s_0}{arg \mapsto 5} \arrow[r, Rightarrow] & \overset{s_1}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}} \arrow[r, Rightarrow] & \overset{s_2}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}} \arrow[r, Rightarrow] & \overset{s_3}{\begin{array}{c} x \mapsto 5 \\ res \mapsto 0 \end{array}} \arrow[d, Rightarrow] \\
\overset{s_1}{\begin{array}{c} x \mapsto 16 \\ res \mapsto 1 \end{array}} & \overset{s_8}{\begin{array}{c} x \mapsto 16 \\ res \mapsto 1 \end{array}} \arrow[l, Rightarrow] & \overset{s_7}{\begin{array}{c} x \mapsto 16 \\ res \mapsto 0 \end{array}} \arrow[l, Rightarrow] & \overset{s_4}{\begin{array}{c} x \mapsto 16 \\ res \mapsto 0 \end{array}} \arrow[l, Rightarrow]
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

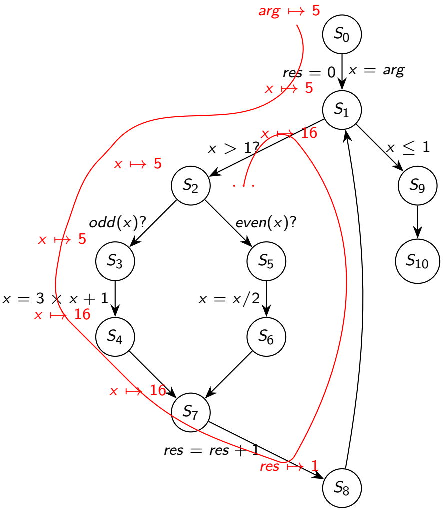
Trace Semantics



$$\begin{tikzcd}
\begin{array}{c} s_0 \\ arg \mapsto 5 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[d, Rightarrow] \\
\begin{array}{c} s_1 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[d, Rightarrow] & \begin{array}{c} s_8 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_7 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] \\
\begin{array}{c} s_2 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} & & &
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

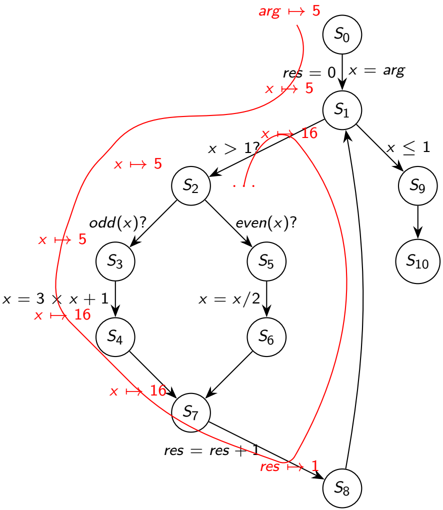
Trace Semantics

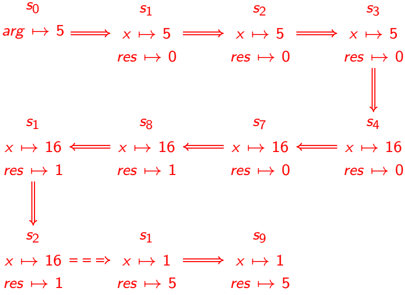


$$\begin{tikzcd}
\begin{array}{c} s_0 \\ arg \mapsto 5 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[d, Rightarrow] \\
\begin{array}{c} s_1 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[d, Rightarrow] & \begin{array}{c} s_8 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_7 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] \\
\begin{array}{c} s_2 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[r, Rightarrow, dashed] & \begin{array}{c} s_1 \\ x \mapsto 1 \\ res \mapsto 5 \end{array} & &
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

Trace Semantics




s_0: $arg \mapsto 5$ $\Longrightarrow$ s_1: $x \mapsto 5$, $res \mapsto 0$ $\Longrightarrow$ s_2: $x \mapsto 5$, $res \mapsto 0$ $\Longrightarrow$ s_3: $x \mapsto 5$, $res \mapsto 0$

$\Downarrow$

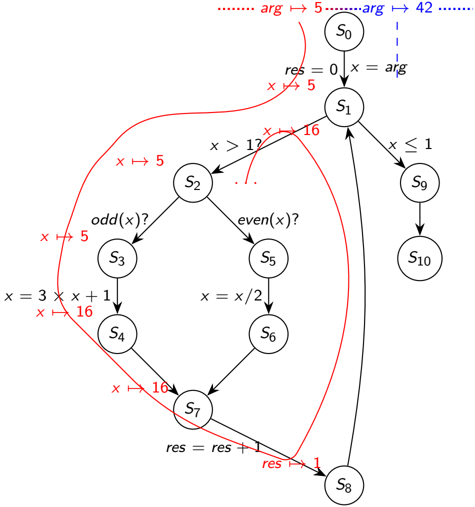
s_4: $x \mapsto 16$, $res \mapsto 0$ $\Longleftarrow$ … s_7: $x \mapsto 16$, $res \mapsto 0$ $\Longleftarrow$ … s_8: $x \mapsto 16$, $res \mapsto 1$ $\Longleftarrow$ … s_1: $x \mapsto 16$, $res \mapsto 1$

$\Downarrow$

s_2: $x \mapsto 16$, $res \mapsto 1$ $= = \Rightarrow$ s_1: $x \mapsto 1$, $res \mapsto 5$ $\Longrightarrow$ s_9: $x \mapsto 1$, $res \mapsto 5$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

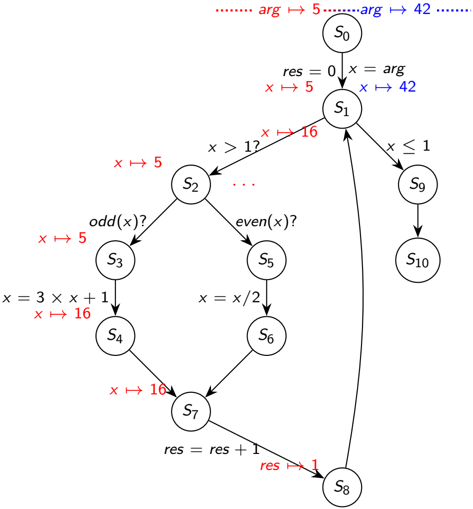
Trace Semantics



$$\begin{tikzcd}
\begin{matrix} s_0 \\ arg \mapsto 5 \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{matrix} \arrow[d, Rightarrow] \\
\begin{matrix} s_1 \\ x \mapsto 16 \\ res \mapsto 1 \end{matrix} \arrow[d, Rightarrow] & \begin{matrix} s_8 \\ x \mapsto 16 \\ res \mapsto 1 \end{matrix} \arrow[l, Rightarrow] & \begin{matrix} s_7 \\ x \mapsto 16 \\ res \mapsto 0 \end{matrix} \arrow[l, Rightarrow] & \begin{matrix} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{matrix} \arrow[l, Rightarrow] \\
\begin{matrix} s_2 \\ x \mapsto 16 \\ res \mapsto 1 \end{matrix} \arrow[r, Rightarrow, dashed] & \begin{matrix} s_1 \\ x \mapsto 1 \\ res \mapsto 5 \end{matrix} \arrow[r, Rightarrow] & \begin{matrix} s_9 \\ x \mapsto 1 \\ res \mapsto 5 \end{matrix} &
\end{tikzcd}$$

- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

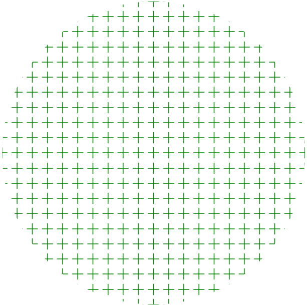
Trace Semantics



$$\begin{tikzcd}
\begin{array}{c} s_0 \\ arg \mapsto 5 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_1 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_2 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_3 \\ x \mapsto 5 \\ res \mapsto 0 \end{array} \arrow[d, Rightarrow] \\
\begin{array}{c} s_1 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[d, Rightarrow] & \begin{array}{c} s_8 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_7 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] & \begin{array}{c} s_4 \\ x \mapsto 16 \\ res \mapsto 0 \end{array} \arrow[l, Rightarrow] \\
\begin{array}{c} s_2 \\ x \mapsto 16 \\ res \mapsto 1 \end{array} \arrow[r, Rightarrow, dashed] & \begin{array}{c} s_1 \\ x \mapsto 1 \\ res \mapsto 5 \end{array} \arrow[r, Rightarrow] & \begin{array}{c} s_9 \\ x \mapsto 1 \\ res \mapsto 5 \end{array} &
\end{tikzcd}$$

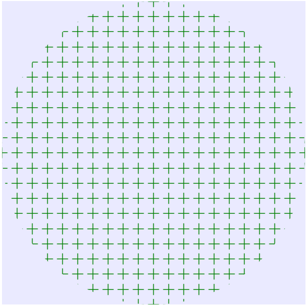
- Initial state on start node
- Transfer functions across edges
- Collecting semantics: recall all states associated to each program point

Static Analysis framework



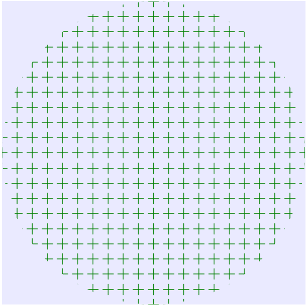
- Replace set of states ...
- ... by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Static Analysis framework



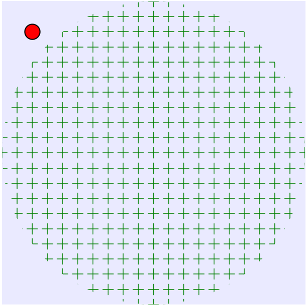
- Replace set of states …
- … by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Static Analysis framework



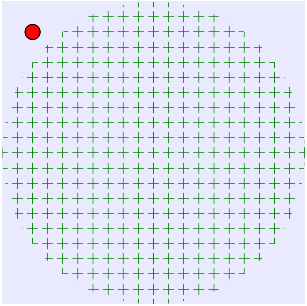
- Replace set of states ...
- ... by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Static Analysis framework



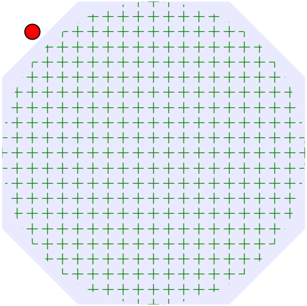
- Replace set of states ...
- ... by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Static Analysis framework



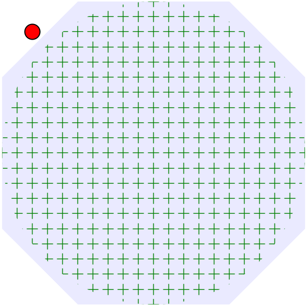
- Replace set of states ...
- ... by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Static Analysis framework



- Replace set of states ...
- ... by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Static Analysis framework



- Replace set of states ...
- ... by one element in an abstract lattice
- Correctness: dot not miss any possible concrete state
- Over-approximation and false alarms
- Termination: ensure analysis always terminates
- Trade-off between precision and computation time
- Abstract interpretation: systematic way to build correct and terminating analyses (Galois connexions and widening)

Abstract interpretation in practice

A few tools

- Polyspace Verifier: check absence of runtime errors (C/C++/Ada) https://fr.mathworks.com/products/polyspace.html
- ASTRÉE: absence of runtime errors without false alarm in SCADE-generated code https://www.absint.com/astree/index.htm
- Verasco: certified (in Coq) analyzer http://compcert.inria.fr/verasco/
- aiT/StackAnalyzer: WCET and stack size (assembly code) https://www.absint.com/ait/
- FLUCTUAT: accuracy of floating-point computations and origin of rounding errors https://www.lix.polytechnique.fr/~putot/fluctuat.html
- Frama-C: platform for analyzing C code, including through abstract interpretation https://frama-c.com
- MoPSA: modular platform for abstract interpretation https://gitlab.com/mopsa/mopsa-analyzer

Frama-C at a glance

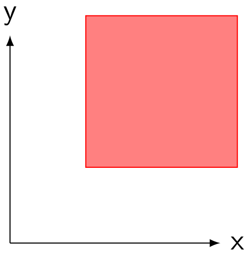
- A Framework for modular analysis of C code.
- https://frama-c.com/
- Developed at CEA Tech List and Inria
- Released under LGPL license (v23.0-rc1 Vanadium in June 2021)
- Kernel based on CIL (Necula et al. – Berkeley).
- ACSL annotation language.
- Extensible platform
  - Collaboration of analyses over same code
  - Inter plug-in communication through ACSL formulas.
  - Adding specialized plug-in is easy

Relational and Non-relational Lattices

Non-relational domain

- Considers each variable independently
- ✓ Simpler and less costly
- ✗ lose properties over 2+ variables

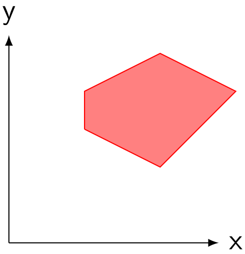
Example: intervals



Relational domain

- Considers several variables at once
- ✓ More precise
- ✗ More complex and costly

Example: Polyhedra



Frama-C: Integer Arithmetic

Corresponding Abstract Domain

$$\begin{array}{ll} & \text{small set of integers (by default, cardinal} \leq 8) \\ \uplus & \text{integer interval} \end{array}$$

Examples

- $\{0; 40; \}$ = 0 or 40
- $[0..40]$ = an integer between 0 and 40 (inclusive)
- $[-..-]$ = any integer (within the bound of the corresponding integral type)

Integers in Eva Quiz

Question

if `x` is in the interval `[-10 .. 10]` before the execution of statement

```
if (x==0) { y = 14; }
else { y = x<0 ? 13 : x + 2; }
```

What is the value associated to `y` after the statement?

Answers

- ▸ a `[-8 .. 14]`
- ▸ b `[2 .. 13]`
- ▸ c `[2 .. 14]`
- ▸ d `[3 .. 14]`

◂ Back to presentation ▸ See solution ⏩ Skip quiz

Reduced product

- Combining abstract domains
- reduce abstract value from one domain using information from the other

✘ In practice, not as simple and generic as it looks

✘ Combining transfer function is complex

Reduced product

Question

We have information from two domains:
Intervals:

- $x \in [0;20]$
- $y \in [5;10]$

Octagons:
$0 \leq x - y \leq 20$

What can be said about x and y?

Answers

- a $x \in [0;20]$, $y \in [5;10]$; $0 \leq x - y \leq 20$
- b $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 15$
- c $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 10$
- d $x \in [5;20]$, $y \in [0;20]$, $0 \leq x - y \leq 20$

◂ Back to presentation ▸ See solution ⏩ Skip quiz

Frama-C: Memory Address

Base Address

Global variable
$\uplus$ Formal parameter of main function
$\uplus$ literal string constant
$\uplus$ NULL
$\uplus$ . . .

Addresses

- Base address + Offset (integer)
- Integer representation: product of interval and modulo (aka alignment) information
- Each base has a maximal valid offset
- Abstract Values are sets of addresses

Content of a memory location

Abstract Domain

valid left value

$$\begin{array}{rl} & \text{adress} \\ \times & \text{initialized?} \\ \times & \textit{not dangling pointer}\text{?} \end{array}$$

Example

```
   int x,y;
   if (e) x = 2;
L: if (e) y = x + 1;
```

- At *L*, we know that *x* equals 2 iff it has been initialized
- Depending on the complexity of *e*, we know that *y* equals 3 if *x* equals 2

next

Memory in Eva Quiz

Question

if `a` is an array of size `3`, initialized to `0`, and `c` in `[0 .. 2]` what would be the content of `a` after executing the following statement:

```
if (c) { a[c] = c; } else a[1] =3;
```

Answers

▸ a `a[0] IN {0}, a[1] IN {0,1,3}, a[2] IN {0,2}`

▸ b `a[i] IN {0,1,2,3}` for all indices

▸ c

`a[0] IN {0}, a[1] IN {0,1,2,3}, a[2] IN {0,1,2}`

▸ d `a[0] IN {0}, a[1] IN {1,3}, a[2] IN {2}`

◂ Back to presentation ▸ See solution ▸▸ Skip quiz

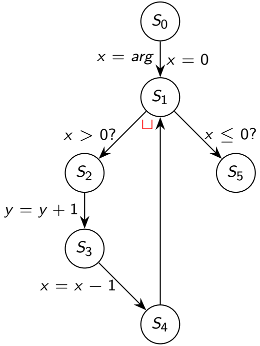
Adding other domains

- New domains can provide additional information:
  - equalities between values
  - values of symbolic locations
  - gauges, affine relation wrt number of loop steps
- Possible to add new domains
- Inter-domain communication done through queries:

```
val extract_expr :
(exp -> value evaluated) ->
state -> exp -> (value * origin) evaluated

val extract_lval :
  (exp -> value evaluated) ->
  state -> lval -> typ -> location ->
  (value * origin) evaluated
```

Widening

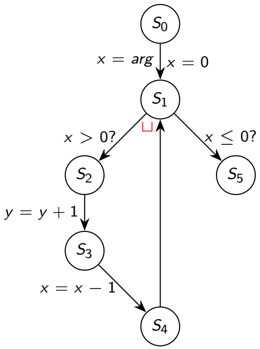


	S_1 (before)		S_4		S_1 (after)
$y \in$	$[0;0]$	$\sqcup$	$[1;1]$	$=$	$[0;1]$

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator ∇:

correctness	$x \sqcup y \sqsubseteq x \nabla y$
termination	no infinitely growing sequence $x_0 \nabla x_1 \nabla \dots \nabla x_n \dots$

Widening



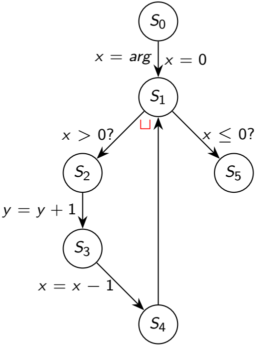
	S_1 (before)		S_4		S_1 (after)
$y \in$	$[0; 1]$	$\sqcup$	$[1; 2]$	$=$	$[0; 2]$

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator ∇:

correctness $x \sqcup y \sqsubseteq x \nabla y$

termination no infinitely growing sequence $x_0 \nabla x_1 \nabla \ldots \nabla x_n \ldots$

Widening



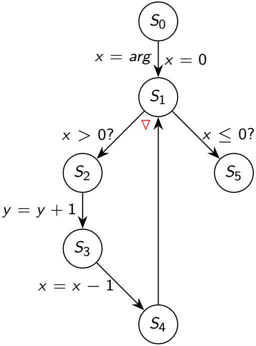
$$\begin{array}{cccccc} & S_1\text{ (before)} & & S_4 & & S_1\text{ (after)}\\ y \in & [0;2] & \sqcup & [1;3] & = & [0;3]\end{array}$$

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator ∇:

correctness $x \sqcup y \sqsubseteq x \nabla y$

termination no infinitely growing sequence $x_0 \nabla x_1 \nabla \ldots \nabla x_n \ldots$

Widening

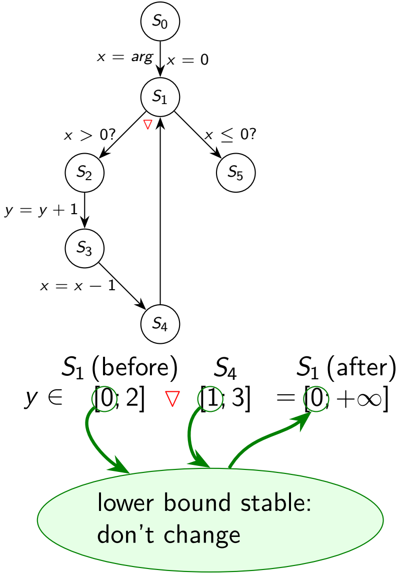


$$\begin{array}{lcccc} & S_1 \text{ (before)} & & S_4 & & S_1 \text{ (after)} \\ y \in & [0;2] & {\color{red}\triangledown} & [1;3] & = & [0;+\infty] \end{array}$$

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator $\triangledown$:

correctness	$x \sqcup y \sqsubseteq x \triangledown y$
termination	no infinitely growing sequence $x_0 \triangledown x_1 \triangledown \dots \triangledown x_n \dots$

Widening



S_1 (before) $\quad S_4 \quad S_1$ (after)

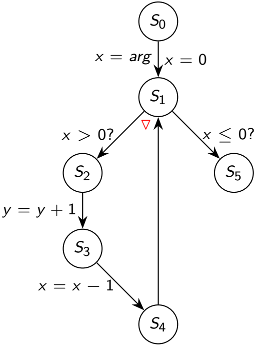
$y \in [0; 2] \;\triangledown\; [1; 3] = [0; +\infty]$

lower bound stable: don't change

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator $\triangledown$:

correctness	$x \sqcup y \sqsubseteq x \triangledown y$
termination	no infinitely growing sequence $x_0 \triangledown x_1 \triangledown \ldots \triangledown x_n \ldots$

Widening



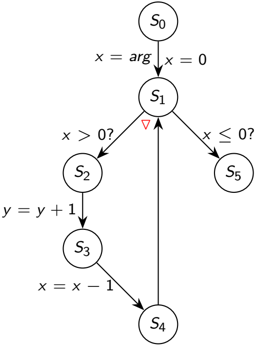
$$\begin{array}{lccccc} & S_1\text{ (before)} & & S_4 & & S_1\text{ (after)} \\ y \in & [0;2] & \triangledown & [1;3] & = & [0;+\infty] \end{array}$$

upper bound grows:
widen interval

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator ∇:

correctness	$x \sqcup y \sqsubseteq x \nabla y$
termination	no infinitely growing sequence $x_0 \nabla x_1 \nabla \dots \nabla x_n \dots$

Widening



$$y \in \overset{S_1\text{ (before)}}{[0;+\infty]} \triangledown \overset{S_4}{[1;+\infty]} = \overset{S_1\text{ (after)}}{[0;+\infty]}$$

check fixpoint is reached

- for loop nodes, state grows slowly at each step
- convergence could require infinite time
- replace $\sqcup$ with widening operator $\triangledown$:

correctness	$x \sqcup y \sqsubseteq x \triangledown y$
termination	no infinitely growing sequence $x_0 \triangledown x_1 \triangledown \ldots \triangledown x_n \ldots$

Loops in Eva

Widening

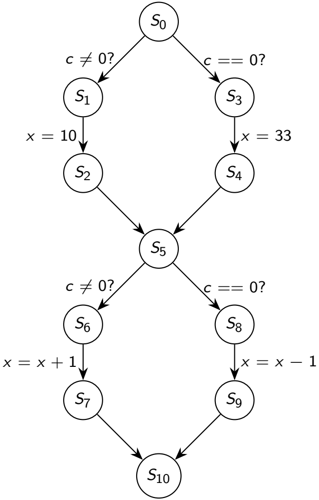
- Controlling widening: `-eva-widening-delay`, `-eva-widening-period`
- Controlling widening bounds:

```
widen_hints v, 10, 100, 1000;
```

Loop unrolling

- `-eva-auto-loop-unroll n`
- `-eva-min-loop-unroll  n`
- or with annotations on specific loops

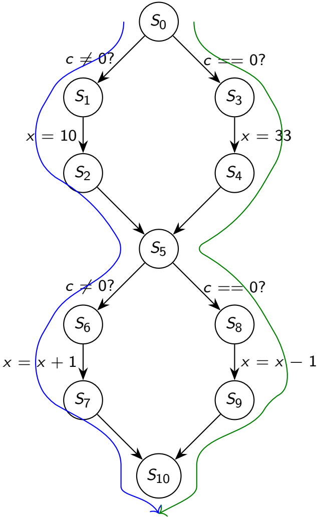
Trace Partitioning



	S_5	S_6	S_8	S_{10}
c	$\mathbb{Z}$	$\mathbb{Z}$	0	$\mathbb{Z}$
x	[10; 33]	[10; 33]	[10; 33]	[9; 34]

- ▶ Consider several abstract traces separately…
- ▶ …At least for some time
- ✔ More precise than collecting semantics
- ✘ Finding appropriate partition is difficult

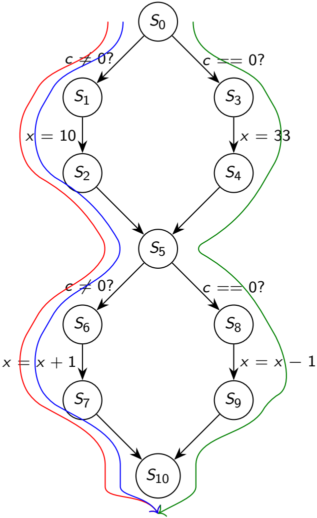
Trace Partitioning



	S_5	S_6	S_8	S_{10}
c	0	$\bot$	0	0
x	33	$\bot$	33	32
c	$\mathbb{Z}$	$\mathbb{Z}$	0	$\mathbb{Z}$
x	10	10	10	$[9; 11]$

- ▶ Consider several abstract traces separately...
- ▶ ...At least for some time
- ✔ More precise than collecting semantics
- ✘ Finding appropriate partition is difficult

Trace Partitioning



	S_5	S_6	S_8	S_{10}
c	0	$\perp$	0	0
x	33	$\perp$	33	32
c	$[1; +\infty]$	$[1; +\infty]$	$\perp$	$[1; +\infty]$
x	10	10	$\perp$	11
c	$[-\infty; -1]$	$[-\infty; -1]$	$\perp$	$[-\infty; -1]$
x	10	10	$\perp$	11

- ▶ Consider several abstract traces separately...
- ▶ ...At least for some time
- ✔ More precise than collecting semantics
- ✘ Finding appropriate partition is difficult

Separating states in Eva

Slevel options

- option `-eva-slevel`: allows Eva to explore *n* separated paths before joining them
- option `-eva-slevel-function`: same as previous, but for a particular function
- Can also use ACSL annotations to partition the state

ACSL Annotations

use a disjunction (that covers all possible cases) to split the state, and appropriate `-eva-slevel`

```
/*@ assert A || B || C; */
```

Trace Partioning in Eva

Explicit split

```
/*@ split expr; */
...
/*@ merge expr; */
```

`expr` must evaluate to a small set of values (e.g. be a boolean expression)

Entry Point

- Which part of the code should be analyzed?
- `-main f` starts the analysis at function `f`
- `-lib-entry` indicates that the the initial global context is not 0-initialized
- `-eva-context-width`, `-eva-context-depth`
- Use of a driver function with some builtins to provide non-determinism:

```
void f_wrapper() {
  setup_analysis_context();
  f(arg_1, arg_2);
}
```

External Library Functions

Provide an “implementation” for Eva

- Assumed to match the real implementation
- Write stub directly in C (aimed at ease of analysis, not performance)
- Provide an ACSL specification
- `-eva-use-spec f`
- Use an Eva built-in (`-eva-builtin`)
- `-eva-builtins-list`

CERT ARR30-C bad code sample

```
static int *table = NULL;
static size_t size = 0;

int insert_in_table(size_t pos, int value) {
  if (size < pos) {
    int *tmp;
    size = pos + 1;
    tmp = (int *)realloc(
       table, sizeof(*table) * size);
    if (tmp == NULL) {
      return -1;   /* Failure */
    }
    table = tmp;
  }
  table[pos] = value;
  return 0;
```

Analyzing real code

- D. Delmas and J. Souyris: ASTRÉE: from Research to Industry, SAS 2007
- TrustInSoft startup (created 2013): `https://trust-in-soft.com/`
- A. Ourghanlian: Evaluation of static analysis tools used to assess software important to nuclear power plant safety. In Nuclear Engineering and Technology, vol 47 issue 2, 2015.
- A. Ebalard &al.: Journey to a RTE-free X.509 parser

  `https://www.sstic.org/2019/presentation/journey-to-a-rte-free-x509-parser/`
- Open-Source Case Studies:

  `https://github.com/Frama-C/open-source-case-studies`
- A. Maroneze: Analysis of the Chrony NTP server.

  `https://frama-c.com/2018/06/19/Analyzing-Chrony-with-Frama-C-Eva.html`

General

- Correnson &al. Frama-C User Manual (v22 - Titanium). November 2020
- Kirchner &al. Frama-C, a Software Analysis Perspective, vol 37 of Formal Aspects of Computing, March 2015.

Eva

- Cuoq &al. Frama-C's value analysis plug-in. November 2020
- Blazy &al. Structuring Abstract Interpreters through State and Value Abstractions. VMCAI, January 2017

Static Analysis

Course

- Patrick Cousot, MIT 2005 `http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/`

Books

- Hanne Nielson, Flemming Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer 1999
- Neil Jones and Flemming Nielson, *Abstract Interpretation: a Semantics-Based Tool for Program Analysis*. In *Handbook of Logic in Computer Science, vol. 4*, Oxford University Press 1994

Static Analysis (cont'd)

Founding Articles

- Patrick and Radhia Cousot, *Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints*. PoPL'77
- Patrick Cousot and Nicolas Halbwachs, *Automatic Discovery of Linear Restraints Among Variables of a Program*. PoPL'78
- Patrick and Radhia Cousot, *Systematic Design of Program Analysis Frameworks*. PoPL'79
- `http://www.di.ens.fr/~cousot/COUSOTpapers.shtml`

Solutions to Quizzes

Integers in Eva Quiz

Question

if `x` is in the interval `[-10 .. 10]` before the execution of statement

```
if (x==0) { y = 14; }
else { y = x<0 ? 13 : x + 2; }
```

What is the value associated to `y` after the statement?

Answers

- ▸ a `[-8 .. 14]` ✘
- ▸ b `[2 .. 13]`
- ▸ c `[2 .. 14]`
- ▸ d `[3 .. 14]`

◂ Back to presentation ▸ See solution

Integers in Eva Quiz

Question

if `x` is in the interval `[-10 .. 10]` before the execution of statement

```
if (x==0) { y = 14; }
else { y = x<0 ? 13 : x + 2; }
```

What is the value associated to `y` after the statement?

Answers

- ▸ a `[-8 .. 14]`
- ▸ b `[2 .. 13]` ✘
- ▸ c `[2 .. 14]`
- ▸ d `[3 .. 14]`

◂ Back to presentation ▸ See solution

Integers in Eva Quiz

Question

if x is in the interval [-10 .. 10] before the execution of statement

```
if (x==0) { y = 14; }
else { y = x<0 ? 13 : x + 2; }
```

What is the value associated to y after the statement?

Answers

- ▸ a [-8 .. 14]
- ▸ b [2 .. 13]
- ▸ c [2 .. 14] ✔
- ▸ d [3 .. 14]

◂ Back to presentation ▸ See solution

Integers in Eva Quiz

Question

if `x` is in the interval `[-10 .. 10]` before the execution of statement

```
if (x==0) { y = 14; }
else { y = x<0 ? 13 : x + 2; }
```

What is the value associated to `y` after the statement?

Answers

- ▸ a `[-8 .. 14]`
- ▸ b `[2 .. 13]`
- ▸ c `[2 .. 14]`
- ▸ d `[3 .. 14]` ✗

◂ Back to presentation ▸ See solution

Reduced product

Question

We have information from two domains:

Intervals:

- $x \in [0;20]$
- $y \in [5;10]$

Octagons:

$0 \leq x - y \leq 20$

What can be said about x and y?

Answers

- a $x \in [0;20]$, $y \in [5;10]$; $0 \leq x - y \leq 20$ ✘
- b $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 15$
- c $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 10$
- d $x \in [5;20]$, $y \in [0;20]$, $0 \leq x - y \leq 20$

◂ Back to presentation ▸ See solution

Reduced product

Question

We have information from two domains:

Intervals:

- $x \in [0;20]$
- $y \in [5;10]$

Octagons:

$0 \leq x - y \leq 20$

What can be said about x and y?

Answers

- a $x \in [0;20]$, $y \in [5;10]$; $0 \leq x - y \leq 20$
- b $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 15$ ✓
- c $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 10$
- d $x \in [5;20]$, $y \in [0;20]$, $0 \leq x - y \leq 20$

◂ Back to presentation

▸ See solution

Reduced product

Question

We have information from two domains:
Intervals:

- $x \in [0;20]$
- $y \in [5;10]$

Octagons:
$0 \leq x - y \leq 20$

What can be said about x and y?

Answers

▸ a $x \in [0;20]$, $y \in [5;10]$; $0 \leq x - y \leq 20$

▸ b $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 15$

▸ c $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 10$ ✗

▸ d $x \in [5;20]$, $y \in [0;20]$, $0 \leq x - y \leq 20$

◂ Back to presentation ▸ See solution

Reduced product

Question

We have information from two domains:
Intervals:

- $x \in [0;20]$
- $y \in [5;10]$

Octagons:
$0 \leq x - y \leq 20$

What can be said about x and y?

Answers

▸ a $x \in [0;20]$, $y \in [5;10]$; $0 \leq x - y \leq 20$

▸ b $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 15$

▸ c $x \in [5;20]$, $y \in [5;10]$, $0 \leq x - y \leq 10$

▸ d $x \in [5;20]$, $y \in [0;20]$, $0 \leq x - y \leq 20$ ✘

◂ Back to presentation

▸ See solution

Memory in Eva Quiz

Question

if `a` is an array of size `3`, initialized to `0`, and `c` in `[0 .. 2]` what would be the content of `a` after executing the following statement:

```
if (c) { a[c] = c; } else a[1] =3;
```

Answers

▸ a `a[0] IN {0}, a[1] IN {0,1,3}, a[2] IN {0,2}`

✘

▸ b `a[i] IN {0,1,2,3}` for all indices

▸ c

`a[0] IN {0}, a[1] IN {0,1,2,3}, a[2] IN {0,1,2}`

▸ d `a[0] IN {0}, a[1] IN {1,3}, a[2] IN {2}`

◂ Back to presentation

▸ See solution

Memory in Eva Quiz

Question

if `a` is an array of size `3`, initialized to `0`, and `c` in `[0 .. 2]` what would be the content of `a` after executing the following statement:

```
if (c) { a[c] = c; } else a[1] =3;
```

Answers

- ▸ a `a[0] IN {0}, a[1] IN {0,1,3}, a[2] IN {0,2}`
- ▸ b `a[i] IN {0,1,2,3}` for all indices ✘
- ▸ c

  `a[0] IN {0}, a[1] IN {0,1,2,3}, a[2] IN {0,1,2}`
- ▸ d `a[0] IN {0}, a[1] IN {1,3}, a[2] IN {2}`

◂ Back to presentation

▸ See solution

Memory in Eva Quiz

Question

if `a` is an array of size `3`, initialized to `0`, and `c` in `[0 .. 2]` what would be the content of `a` after executing the following statement:

```
if (c) { a[c] = c; } else a[1] =3;
```

Answers

- ▸ a `a[0] IN {0}, a[1] IN {0,1,3}, a[2] IN {0,2}`
- ▸ b `a[i] IN {0,1,2,3}` for all indices
- ▸ c `a[0] IN {0}, a[1] IN {0,1,2,3}, a[2] IN {0,1,2}` ✔
- ▸ d `a[0] IN {0}, a[1] IN {1,3}, a[2] IN {2}`

◂ Back to presentation ▸ See solution

Memory in Eva Quiz

Question

if `a` is an array of size `3`, initialized to `0`, and `c` in `[0 .. 2]` what would be the content of `a` after executing the following statement:

```
if (c) { a[c] = c; } else a[1] =3;
```

Answers

- ▸ a `a[0] IN {0}, a[1] IN {0,1,3}, a[2] IN {0,2}`
- ▸ b `a[i] IN {0,1,2,3}` for all indices
- ▸ c

  `a[0] IN {0}, a[1] IN {0,1,2,3}, a[2] IN {0,1,2}`
- ▸ d `a[0] IN {0}, a[1] IN {1,3}, a[2] IN {2}` ✗

◂ Back to presentation

▸ See solution