

Abstract interpretation with numeric intervals

Second assignment of the Software Verification
course, A.Y. 2022/2023

Christian Micheletti
October 21, 2023



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- 1** The Language
 - Arithmetic Expressions
 - Boolean Expressions

- 2** Introduction

- 3** First section

The language is a variation of the While language seen in class. It differs on:

- it admits some syntactic sugar (it's not minimal);
- its semantic functions are modified to allow divergence and state changes in both arithmetic and boolean expressions.

$$\begin{aligned} AExp ::= & n \mid x \mid -e \mid (e) \\ & \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1 / e_2 \\ & \mid x++ \mid ++x \mid x-- \mid --x \end{aligned}$$

$$\mathcal{A} : AExp \rightarrow State \hookrightarrow \mathbb{Z} \times State$$

$$\mathcal{A}[[n]]\varphi = (n_{\mathbb{Z}}, \varphi)$$

$$\mathcal{A}[[x]]\varphi = (\varphi(x), \varphi)$$

$$\mathcal{A}[[e]]\varphi = \mathcal{A}[[e]]\varphi$$

$$\mathcal{A}[-e]\varphi = \begin{cases} (-a, \varphi') & \mathcal{A}[[e]]\varphi = (a, \varphi') \\ \uparrow & (\mathcal{A}[[e]]\varphi) \uparrow \end{cases}$$

$\mathcal{A} : AExp \rightarrow State \hookrightarrow \mathbb{Z} \times State$

$$\mathcal{A}[\![e_1/e_2]\!] \varphi = \begin{cases} (a_1 \div a_2, \varphi'') & \mathcal{A}[\![e_1]\!] \varphi = (a_1, \varphi') \\ & \wedge \mathcal{A}[\![e_2]\!] \varphi' = (a_2, \varphi'') \\ & \wedge a_2 \neq 0 \\ \uparrow & \text{otherwise} \end{cases}$$
$$\mathcal{A}[\![e_1 \text{ op } e_2]\!] \varphi = \begin{cases} (a_1 \text{ op } a_2, \varphi'') & \mathcal{A}[\![e_1]\!] \varphi = (a_1, \varphi') \\ & \wedge \mathcal{A}[\![e_2]\!] \varphi' = (a_2, \varphi'') \\ \uparrow & \text{otherwise} \end{cases}$$

$\mathcal{A} : AExp \rightarrow State \hookrightarrow \mathbb{Z} \times State$

$$\mathcal{A}[[x++]]\varphi = (\varphi(x), \varphi[x \mapsto x + 1])$$

$$\begin{aligned} \mathcal{A}[[++x]]\varphi = & \text{let } \varphi' = \varphi[x \mapsto x + 1] \\ & \text{in } (\varphi'(x), \varphi') \end{aligned}$$

$$\mathcal{A}[[x--]]\varphi = (\varphi(x), \varphi[x \mapsto x - 1])$$

$$\begin{aligned} \mathcal{A}[[--x]]\varphi = & \text{let } \varphi' = \varphi[x \mapsto x - 1] \\ & \text{in } (\varphi'(x), \varphi') \end{aligned}$$

$$\begin{aligned} BExp ::= & \text{true} \mid \text{false} \mid (b) \mid b_1 \text{ and } b_2 \mid b_1 \text{ or } b_2 \\ & \mid e_1 = e_2 \mid e_1 \neq e_2 \mid e_1 < e_2 \mid e_1 \geq e_2 \end{aligned}$$

$\mathcal{B} : BExp \rightarrow State \hookrightarrow \mathbb{T} \times State$

$$\mathcal{B}[\text{true}]\varphi = (\mathbf{tt}, \varphi)$$

$$\mathcal{B}[\text{false}]\varphi = (\mathbf{ff}, \varphi)$$

$$\mathcal{B}[(b)]\varphi = \mathcal{B}[b]\varphi$$

Operators between booleans short circuits results:

$$\mathcal{B} : BExp \rightarrow State \hookrightarrow \mathbb{T} \times State$$

$$\mathcal{B}[[b_1 \text{ and } b_2]]\varphi = \begin{cases} (\mathbf{ff}, \varphi') & \mathcal{B}[[b_1]]\varphi = (\mathbf{ff}, \varphi') \\ \mathcal{B}[[b_2]]\varphi' & \mathcal{B}[[b_1]]\varphi = (\mathbf{tt}, \varphi') \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mathcal{B}[[b_1 \text{ or } b_2]]\varphi = \begin{cases} (\mathbf{tt}, \varphi') & \mathcal{B}[[b_1]]\varphi = (\mathbf{tt}, \varphi') \\ \mathcal{B}[[b_2]]\varphi' & \mathcal{B}[[b_1]]\varphi = (\mathbf{ff}, \varphi') \\ \uparrow & \text{otherwise} \end{cases}$$

Comparison operations propagate updates in the state:

$$\mathcal{B} : BExp \rightarrow State \hookrightarrow \mathbb{T} \times State$$

$$\mathcal{B}[\![e_1 = e_2]\!] \varphi = \begin{cases} (a_1 = a_2, \varphi'') & \mathcal{A}[\![e_1]\!] \varphi = (a_1, \varphi') \\ & \wedge \mathcal{A}[\![e_2]\!] \varphi' = (a_2, \varphi'') \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mathcal{B}[\![e_1 < e_2]\!] \varphi = \begin{cases} (a_1 < a_2, \varphi'') & \mathcal{A}[\![e_1]\!] \varphi = (a_1, \varphi') \\ & \wedge \mathcal{A}[\![e_2]\!] \varphi' = (a_2, \varphi'') \\ \uparrow & \text{otherwise} \end{cases}$$

$\mathcal{B} : BExp \rightarrow State \hookrightarrow \mathbb{T} \times State$

$$\mathcal{B}[\![e_1 \neq e_2]\!] \varphi = \begin{cases} (a_1 \neq a_2, \varphi'') & \mathcal{A}[\![e_1]\!] \varphi = (a_1, \varphi') \\ & \wedge \mathcal{A}[\![e_2]\!] \varphi' = (a_2, \varphi'') \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mathcal{B}[\![e_1 \geq e_2]\!] \varphi = \begin{cases} (a_1 \geq a_2, \varphi'') & \mathcal{A}[\![e_1]\!] \varphi = (a_1, \varphi') \\ & \wedge \mathcal{A}[\![e_2]\!] \varphi' = (a_2, \varphi'') \\ \uparrow & \text{otherwise} \end{cases}$$

Negation is expressed by syntactic sugar:

$$\text{not true} \stackrel{\text{def}}{=} \text{false}$$

$$\text{not false} \stackrel{\text{def}}{=} \text{true}$$

$$\text{not } (b_1 \text{ and } b_2) \stackrel{\text{def}}{=} \text{not } b_1 \text{ or not } b_2$$

$$\text{not } (b_1 \text{ or } b_2) \stackrel{\text{def}}{=} \text{not } b_1 \text{ and not } b_2$$

$$\text{not } e_1 = e_2 \stackrel{\text{def}}{=} e_1 \neq e_2$$

$$\text{not } e_1 < e_2 \stackrel{\text{def}}{=} e_1 \geq e_2$$

$$\text{not } e_1 \neq e_2 \stackrel{\text{def}}{=} e_1 = e_2$$

$$\text{not } e_1 \geq e_2 \stackrel{\text{def}}{=} e_1 < e_2$$

Also other arithmetic comparisons are expressed with syntactic sugar:

$$e_1 > e_2 \stackrel{\text{def}}{=} e_2 < e_1$$
$$e_1 <= e_2 \stackrel{\text{def}}{=} e_2 >= e_1$$

Etiam eu interdum ligula

Nunc mi eros, vulputate in ornare a, viverra eget quam

- Morbi **vitae lacus** porta neque tincidunt sodales
- Proin tincidunt, **neque** at tincidunt mollis
- Ut **lacinia sem a nibh** consequat porttitor

Normal block

Fusce luctus venenatis felis quis semper

Alert block

$$E = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

Example block

Proin tincidunt, neque at tincidunt mollis