Abstract interpretation with numeric intervals

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

Christian Micheletti
October 22, 2023

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Outline

The Language

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.

Arithmetic Expressions (1)

$$AExp ::= n \mid x \mid -e \mid (e)$$
$$\mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1/e_2$$
$$\mid x++ \mid ++x \mid x-- \mid --x$$

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

$$\mathcal{A}[\![n]\!]\varphi = (n_\mathbb{Z}, \varphi)$$
$$\mathcal{A}[\![x]\!]\varphi = (\varphi(x), \varphi)$$
$$\mathcal{A}[\![(e)]\!]\varphi = \mathcal{A}[\![e]\!]\varphi$$
$$\mathcal{A}[\![-e]\!]\varphi = \begin{cases} (-a, \varphi') & \mathcal{A}[\![e]\!]\varphi = (a, \varphi') \\ \uparrow & (\mathcal{A}[\![e]\!]\varphi) \uparrow \end{cases}$$

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

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

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

Arithmetic Expressions (3)

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

$$\mathcal{A}[\![x\text{++}]\!]\varphi = (\varphi(x), \varphi[x \mapsto x + 1])$$
$$\mathcal{A}[\![\text{++}x]\!]\varphi = let\ \varphi' = \varphi[x \mapsto x + 1]$$
$$in\ (\varphi'(x), \varphi')$$
$$\mathcal{A}[\![x\text{--}]\!]\varphi = (\varphi(x), \varphi[x \mapsto x - 1])$$
$$\mathcal{A}[\![\text{--}x]\!]\varphi = let\ \varphi' = \varphi[x \mapsto x - 1]$$
$$in\ (\varphi'(x), \varphi')$$

$$BExp ::= \texttt{true} \mid \texttt{false} \mid (b) \mid b_1 \texttt{ and } b_2 \mid b_1 \texttt{ or } b_2$$
$$\mid e_1 \texttt{ = } e_2 \mid e_1 \texttt{ != } e_2 \mid e_1 \texttt{ < } e_2 \mid e_1 \texttt{ >= } e_2$$

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

$$\mathcal{B}[\![\texttt{true}]\!]\varphi = (\textbf{tt}, \varphi)$$
$$\mathcal{B}[\![\texttt{false}]\!]\varphi = (\textbf{ff}, \varphi)$$
$$\mathcal{B}[\![(b)]\!]\varphi = \mathcal{B}[\![b]\!]\varphi$$

Operators between booleans short circuits results:

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

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

Comparison operations propagate updates in the state:

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

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

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

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

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

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

Negation is expressed by syntactic sugar:

$$\texttt{not true} \overset{\text{def}}{=} \texttt{false}$$

$$\texttt{not false} \overset{\text{def}}{=} \texttt{true}$$

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

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

$$\texttt{not } e_1 \texttt{ = } e_2 \overset{\text{def}}{=} e_1 \texttt{ != } e_2$$

$$\texttt{not } e_1 \texttt{ < } e_2 \overset{\text{def}}{=} e_1 \texttt{ >= } e_2$$

$$\texttt{not } e_1 \texttt{ != } e_2 \overset{\text{def}}{=} e_1 \texttt{ = } e_2$$

$$\texttt{not } e_1 \texttt{ >= } e_2 \overset{\text{def}}{=} e_1 \texttt{ < } e_2$$

Also other arithmetic comparisons are expressed with syntactic sugar:

$$e_1 \; > \; e_2 \; \overset{\text{def}}{=} \; e_2 \; < \; e_1$$

$$e_1 \; <= \; e_2 \; \overset{\text{def}}{=} \; e_2 \; >= \; e_1$$

$$\textbf{While} ::= x := e \mid \texttt{skip} \mid \{S\} \mid S_1 \ ; \ S_2$$
$$\mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S$$

$\mathcal{S}_{ds} : \textit{While} \rightarrow \textit{State} \hookrightarrow \textit{State}$

$$\mathcal{S}_{ds}[\![x \ := \ e]\!]\varphi = \begin{cases} \varphi'[x \mapsto a] & \mathcal{A}[\![e]\!]\varphi = (a, \varphi') \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mathcal{S}_{ds}[\![\texttt{skip}]\!]\varphi = \varphi$$

$$\mathcal{S}_{ds}[\![\{S\}]\!]\varphi = \mathcal{S}_{ds}[\![S]\!]\varphi$$

$\mathcal{S}_{ds} : \textit{While} \rightarrow \textit{State} \hookrightarrow \textit{State}$

$$\mathcal{S}_{ds}[\![S_1 \; ; \; S_2]\!]\varphi = (\mathcal{S}_{ds}[\![S_2]\!] \circ \mathcal{S}_{ds}[\![S_1]\!])\varphi$$

$$\mathcal{S}_{ds}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!]\varphi = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{ds}[\![S_1]\!], \mathcal{S}_{ds}[\![S_2]\!])$$

$$\mathcal{S}_{ds}[\![\texttt{while } b \texttt{ do } S]\!]\varphi = \text{FIX}(\lambda g.cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{ds}[\![S]\!], id))$$

Where

$$cond(pred, g_1, g_2) = \begin{cases} g_1(\varphi') & pred(\varphi) = (\textbf{tt}, \varphi') \\ g_2(\varphi') & pred(\varphi) = (\textbf{ff}, \varphi') \\ \uparrow & \text{otherwise} \end{cases}$$

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

$$E = (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor x_2 \lor x_4)$$

Example block

Proin tincidunt, neque at tincidunt mollis