IMP

This is the symbolic semantics of IMP enriched with reachability logic.

This semantics receives as input programs generated from reachability formulas. Additionally, the semantics contains the circularities also generated from reachability formulas.

MODULE IMP-SYNTAX

SYNTAX $AExp ::= Int$
$\quad\mid Id$
$\quad\mid AExp$ `/` $AExp$ [strict]
$\quad\mid AExp$ `+` $AExp$ [strict]
$\quad\mid AExp$ `-` $AExp$ [strict]
$\quad\mid (AExp)$ [bracket]

SYNTAX $BExp ::= Bool$
$\quad\mid AExp \leq AExp$ [seqstrict]
$\quad\mid$ `!` $BExp$ [strict]
$\quad\mid BExp$ `&&` $BExp$ [strict(1)]
$\quad\mid (BExp)$ [bracket]

SYNTAX $Block ::= \{\}$
$\quad\mid \{Stmt\}$
$\quad\mid Id$ `:` $Block$

SYNTAX $Stmt ::= Block$
$\quad\mid Id$ `=` $AExp$ `;` [strict(2)]
$\quad\mid$ `if` $(BExp)Block$ `else` $Block$ [strict(1)]
$\quad\mid$ `while` $(BExp)Block$
$\quad\mid Stmt\ \ Stmt$
$\quad\mid Id$ `:` $Stmt$

SYNTAX $Pgm ::=$ `int` Ids `;` $Stmt$

SYNTAX $Ids ::= List\{Id, “,”\}$

SYNTAX $Pgm ::=$ `#ps` (Bag)

K tool issues

SYNTAX $Int ::=$ `#symInt` (Id) [onlyLabel, klabel(#symInt)]

SYNTAX $Int ::= (Int)$ [bracket]

SYNTAX $X ::=$ `symInt` [dummySymInt]

END MODULE

MODULE IMP

SYNTAX $KResult ::= Int$
$\quad\mid Bool$

The configuration of IMP is enriched with cells <frozen> and <goal>. Cell <ruleConstraints> will store some labels which are meant to block the first application of the circularity rule. The cell <goal> contains the current goal. Whenever multiple rules can be applied to the same configuration, the current goal will be splitted into multiple goals.

CONFIGURATION:

$$\left\langle \langle \texttt{\#ps}\ (\$PGM{:}Bag) \rangle_{\mathsf{k}}\ \langle \bullet_{Map} \rangle_{\mathsf{state}}\ \langle \mathsf{true} \rangle_{\mathsf{condition}} \right\rangle_{\mathsf{T?}}\ \langle \mathsf{false} \rangle_{\mathsf{success}}$$

The initial semantics of IMP.

RULE $\left\langle \dfrac{X{:}Id}{I} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \cdots X \mapsto I \cdots \right\rangle_{\mathsf{state}}$

RULE $\dfrac{I1{:}Int + I2{:}Int}{I1 +_{Int} I2}$

RULE $\dfrac{I1{:}Int - I2{:}Int}{I1 -_{Int} I2}$

RULE $\dfrac{I1{:}Int \leq I2{:}Int}{I1 \leq_{Int} I2}$

RULE $\dfrac{!\ T{:}Bool}{\neg_{Bool} T}$

RULE $\dfrac{\{\}}{\bullet_{K}}$ [structural]

RULE $\dfrac{\{S\}}{S}$ [structural]

RULE $\left\langle \dfrac{X = I{:}Int\ ;}{\bullet_{K}} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \cdots X \mapsto \dfrac{\_}{I} \cdots \right\rangle_{\mathsf{state}}$

RULE $\dfrac{S1\ \ S2}{S1 \curvearrowright S2}$ [structural]

RULE $\dfrac{\texttt{while}\ (B)S}{\texttt{if}\ (B)\{S\ \ \texttt{while}\ (B)S\}\ \texttt{else}\ \{\}}$ [transition]

RULE $\left\langle \texttt{int}\ \dfrac{X{:}Id\texttt{,}\ Xs{:}Ids}{Xs}\ \texttt{;}\ \_ \right\rangle_{\mathsf{k}}\ \left\langle \rho{:}Map\ \ \dfrac{\bullet_{Map}}{X \mapsto 0} \right\rangle_{\mathsf{state}}$ requires $\neg_{Bool}(X\ \texttt{in}\ \texttt{keys}\ (\rho))$

RULE $\dfrac{\texttt{int}\ \bullet_{Ids}\ \texttt{;}\ S}{S}$ [structural]

Symbolic semantics - the transformed rules

RULE $\left\langle \dfrac{I1{:}Int\ /\ I2{:}Int}{I1 \div_{Int} I2} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \dfrac{Phi{:}Bool}{Phi \wedge_{Bool} I2 =/=_{Int} 0} \right\rangle_{\mathsf{condition}}$ requires `checkSat` $(Phi \wedge_{Bool} (I2 =/=_{Int} 0)) \neq_{K}$ "unsat" [transition]

RULE $\left\langle \dfrac{B1{:}Bool\ \texttt{\&\&}\ B2{:}BExp}{B2} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \dfrac{Phi{:}Bool}{Phi \wedge_{Bool} B1\ \texttt{==Bool true}} \right\rangle_{\mathsf{condition}}$ requires `checkSat` $(Phi \wedge_{Bool} B1) \neq_{K}$ "unsat" [transition]

RULE $\left\langle \dfrac{B1{:}Bool\ \texttt{\&\&}\ B2{:}Bool}{\mathsf{false}} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \dfrac{Phi{:}Bool}{Phi \wedge_{Bool} \neg_{Bool} B1} \right\rangle_{\mathsf{condition}}$ requires `checkSat` $(Phi \wedge_{Bool} \neg_{Bool} B1) \neq_{K}$ "unsat" [transition]

RULE $\left\langle \dfrac{\texttt{if}\ (B{:}Bool)S\ \texttt{else}\ \_}{S} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \dfrac{Phi{:}Bool}{Phi \wedge_{Bool} B} \right\rangle_{\mathsf{condition}}$ requires `checkSat` $(Phi \wedge_{Bool} B) \neq_{K}$ "unsat" [transition, computational]

RULE $\left\langle \dfrac{\texttt{if}\ (B{:}Bool)\_\ \texttt{else}\ S}{S} \cdots \right\rangle_{\mathsf{k}}\ \left\langle \dfrac{Phi{:}Bool}{Phi \wedge_{Bool} \neg_{Bool} B} \right\rangle_{\mathsf{condition}}$ requires `checkSat` $(Phi \wedge_{Bool} \neg_{Bool} B) \neq_{K}$ "unsat" [transition, computational]

These rules must be generated from reachability formulas given as input and added to the semantics at runtime. Since we don't have this posibility now, we added them manually

For each reachability formula given as input we have two corresponding generated rules: - one corresponding to circularity deduction rule - one checking if the final configuration implies the righ-hand side of the formula (corresponding to consequece deduction rule).

SYNTAX $Id ::= Token\{\text{"}a\text{"}\}$
$\quad\mid Token\{\text{"}b\text{"}\}$
$\quad\mid Token\{\text{"}x\text{"}\}$
$\quad\mid Token\{\text{"}y\text{"}\}$
$\quad\mid Token\{\text{"}ll0\text{"}\}$
$\quad\mid Token\{\text{"}ll1\text{"}\}$
$\quad\mid Token\{\text{"}ll2\text{"}\}$

SYNTAX $Pgm ::=$ `check0`
$\quad\mid$ `check1`
$\quad\mid$ `check2`

SYNTAX $Bag ::=$ `success`

RULE
$$\dfrac{\langle (\mathsf{ll0} : (\mathsf{a = 0\ ;\ \ b = x\ ;\ \ (ll1\ :\ while\ (y \leq b)\{b = b\ \text{-}\ y\ ;\ \ a = a + 1\ ;\})))} \curvearrowright K{:}K \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A{:}Int\ \ \mathsf{b} \mapsto B{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi{:}Bool \rangle_{\mathsf{condition}}}{\langle K \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A'{:}Int\ \ \mathsf{b} \mapsto B'{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi \wedge_{Bool} 0 \leq_{Int} X \wedge_{Bool} 0 <_{Int} Y \rangle_{\mathsf{condition}}}$$
requires `fresh` $(B'{:}Int) \wedge_{Bool}$ `fresh` $(A'{:}Int) \wedge_{Bool}$ `checkSat` $(Psi \wedge_{Bool} \neg_{Bool}(0 \leq_{Int} X \wedge$
[transition]

RULE
$$\dfrac{\left\langle \langle \mathsf{check0} \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A'{:}Int\ \ \mathsf{b} \mapsto B'{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi{:}Bool \rangle_{\mathsf{condition}} \right\rangle_{\mathsf{T}}}{\bullet_{Bag}}\ \left\langle \dfrac{\_}{\mathsf{true}} \right\rangle_{\mathsf{success}}$$
requires `checkSat` $(Psi \wedge_{Bool} \neg_{Bool}((X ==_{Int} A' *_{Int} Y +_{Int} B' \wedge_{Bool} B' \geq_{Int} 0 \wedge_{Bool} B' <_{Int} Y))) =_{K}$ "unsat" [transition]

RULE
$$\dfrac{\langle (\mathsf{ll1\ :\ while\ (y \leq b)(\{b = b\ \text{-}\ y\ ;\ \ a = a + 1\ ;\}))} \curvearrowright K{:}K \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A{:}Int\ \ \mathsf{b} \mapsto B{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi{:}Bool \rangle_{\mathsf{condition}}}{\langle K \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A'{:}Int\ \ \mathsf{b} \mapsto B'{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi \wedge_{Bool} X ==_{Int} A' *_{Int} Y +_{Int} B' \wedge_{Bool} B' \geq_{Int} 0 \wedge_{Bool} B' <_{Int} Y \rangle_{\mathsf{condition}}}$$
requires `fresh` $(B'{:}Int) \wedge_{Bool}$ `fresh` $(A'{:}Int) \wedge_{Bool}$ `checkSat` $(Psi \wedge_{Bool} \neg_{Bool}(X ==_{Int} A *_{Int} Y +_{Int} B$
[transition]

RULE
$$\dfrac{\left\langle \langle \mathsf{check1} \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A'{:}Int\ \ \mathsf{b} \mapsto B'{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi{:}Bool \rangle_{\mathsf{condition}} \right\rangle_{\mathsf{T}}}{\bullet_{Bag}}\ \left\langle \dfrac{\_}{\mathsf{true}} \right\rangle_{\mathsf{success}}$$
requires `checkSat` $(Psi \wedge_{Bool} \neg_{Bool}(X ==_{Int} A' *_{Int} Y +_{Int} B' \wedge_{Bool} B' \geq_{Int} 0 \wedge_{Bool} B' <_{Int} Y)) =_{K}$ "unsat" [transition]

RULE
$$\dfrac{\langle (\mathsf{ll2\ :\ \{b = b\ \text{-}\ y\ ;\ \ a = a + 1\ ;\})} \curvearrowright K \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A{:}Int\ \ \mathsf{b} \mapsto B{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi{:}Bool \rangle_{\mathsf{condition}}}{\langle K \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A'{:}Int\ \ \mathsf{b} \mapsto B'{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi \wedge_{Bool} (X ==_{Int} A' *_{Int} Y +_{Int} B' \wedge_{Bool} B' \geq_{Int} 0) \rangle_{\mathsf{condition}}}$$
requires `fresh` $(B'{:}Int) \wedge_{Bool}$ `fresh` $(A'{:}Int) \wedge_{Bool}$ `checkSat` $(Psi \wedge_{Bool} \neg_{Bool}(X ==_{Int} A *_{Int} Y +_{Int} B \wedge_{Bool} B \geq_{Int} 0$
[transition]

RULE
$$\dfrac{\left\langle \langle \mathsf{check2} \rangle_{\mathsf{k}}\ \langle \mathsf{a} \mapsto A'{:}Int\ \ \mathsf{b} \mapsto B'{:}Int\ \ \mathsf{x} \mapsto X{:}Int\ \ \mathsf{y} \mapsto Y{:}Int \rangle_{\mathsf{state}}\ \langle Psi{:}Bool \rangle_{\mathsf{condition}} \right\rangle_{\mathsf{T}}}{\bullet_{Bag}}\ \left\langle \dfrac{\_}{\mathsf{true}} \right\rangle_{\mathsf{success}}$$
requires `checkSat` $(Psi \wedge_{Bool} \neg_{Bool}(X ==_{Int} A' *_{Int} Y +_{Int} B' \wedge_{Bool} B' \geq_{Int} 0)) =_{K}$ "unsat" [transition]

Utils

RULE $\dfrac{\left(\langle \texttt{\#ps}\ (B{:}Bag) \rangle_{\mathsf{k}}\ \langle \_ \rangle_{\mathsf{state}}\ \langle \_ \rangle_{\mathsf{condition}} \right)}{B}$ [structural]

END MODULE