Vocabulary of the trustful $\text{JVM}_{\mathcal{I}}$

Instructions:

$$
\begin{aligned}
\mathit{Instr} = {} & \mathit{Prim}(\mathit{PrimOp}) \\
\mid {} & \mathit{Load}(\mathit{MoveType}, \mathit{RegNo}) \\
\mid {} & \mathit{Store}(\mathit{MoveType}, \mathit{RegNo}) \\
\mid {} & \mathit{Dupx}(\mathit{Size}, \mathit{Size}) \\
\mid {} & \mathit{Pop}(\mathit{Size}) \\
\mid {} & \mathit{Goto}(\mathit{Offset}) \\
\mid {} & \mathit{Cond}(\mathit{PrimOp}, \mathit{Offset}) \\
\mid {} & \mathit{Halt}
\end{aligned}
$$

Universes:

$$
\begin{aligned}
\mathit{Offset} &= \mathit{Nat} \\
\mathit{Size} &= \mathit{Nat} \\
\mathit{RegNo} &= \mathit{Nat} \\
\mathit{MoveType} &= \texttt{int} \mid \texttt{long} \mid \texttt{float} \mid \texttt{double} \\
\mathit{Code} &= \mathit{Instr}^*
\end{aligned}
$$

Dynamic state of the $\text{JVM}_{\mathcal{I}}$

Universes:

$\mathit{Pc} = \mathit{Nat}$

$\mathit{Word} =$ 32-bit words

Dynamic functions:

$pc : \mathit{Pc}$

$reg : Map(\mathit{RegNo}, \mathit{Word})$

$opd : \mathit{Word}^*$

Initial state:

$pc = 0$

$reg = \emptyset$

$opd = [\,]$

Trustful execution of $\mathrm{JVM}_{\mathcal{I}}$ instructions

$$
\begin{array}{l}
execVM_I(instr) = \\
\quad \textbf{case } instr \textbf{ of} \\
\quad\quad Prim(p) \rightarrow \textbf{let } (opd', ws) = split(opd, argSize(p)) \\
\quad\quad\quad\quad \textbf{if } p \in divMod \Rightarrow sndArgIsNotZero(ws) \textbf{ then} \\
\quad\quad\quad\quad\quad opd := opd' \cdot JVMS(p, ws) \\
\quad\quad\quad\quad\quad pc \;\; := pc + 1 \\
\quad\quad Dupx(s_1, s_2) \rightarrow \textbf{let } (opd', [ws_1, ws_2]) = splits(opd, [s_1, s_2]) \\
\quad\quad\quad\quad opd := opd' \cdot ws_2 \cdot ws_1 \cdot ws_2 \\
\quad\quad\quad\quad pc \;\; := pc + 1 \\
\quad\quad Pop(s) \rightarrow \textbf{let } (opd', ws) = split(opd, s) \\
\quad\quad\quad\quad opd := opd' \\
\quad\quad\quad\quad pc \;\; := pc + 1 \\
\quad\quad Load(t, x) \rightarrow \textbf{if } size(t) = 1 \textbf{ then } opd := opd \cdot [reg(x)] \\
\quad\quad\quad\quad\quad \textbf{else } opd := opd \cdot [reg(x), reg(x+1)] \\
\quad\quad\quad\quad pc := pc + 1
\end{array}
$$

Trustful execution of $\mathrm{JVM}_{\mathcal{I}}$ instructions (continued)

$$
\begin{array}{l}
execVM_I(instr) = \\
\quad \textbf{case } instr \textbf{ of} \\
\quad\quad Store(t,x) \rightarrow \textbf{let } (opd', ws) = split(opd, size(t)) \\
\quad\quad\quad\quad \textbf{if } size(t) = 1 \textbf{ then } reg := reg \oplus \{(x, ws(0))\} \\
\quad\quad\quad\quad\quad \textbf{else } reg := reg \oplus \{(x, ws(0)), (x+1, ws(1))\} \\
\quad\quad\quad\quad opd := opd' \\
\quad\quad\quad\quad pc \;\; := pc + 1 \\
\quad\quad Goto(o) \rightarrow pc := o \\
\quad\quad Cond(p,o) \rightarrow \textbf{let } (opd', ws) = split(opd, argSize(p)) \\
\quad\quad\quad\quad opd := opd' \\
\quad\quad\quad\quad \textbf{if } JVMS(p, ws) \textbf{ then } pc := o \textbf{ else } pc := pc + 1 \\
\quad\quad Halt \rightarrow halt := \texttt{"Halt"}
\end{array}
$$

Compilation of Java$_{\mathcal{I}}$ expressions $\mathcal{E}: Exp \to Code$

$$
\begin{aligned}
\mathcal{E}(lit) &= Prim(lit) \\
\mathcal{E}(loc) &= Load(\mathcal{T}(loc), \overline{loc}) \\
\mathcal{E}(loc = exp) &= \mathcal{E}(exp) \cdot Dupx(0, size(\mathcal{T}(exp))) \cdot \\
&\quad Store(\mathcal{T}(exp), \overline{loc}) \\
\mathcal{E}(!\ exp) &= \mathcal{B}_1(exp, \mathrm{una}_1) \cdot Prim(1) \cdot Goto(\mathrm{una}_2) \cdot \\
&\quad \mathrm{una}_1 \cdot Prim(0) \cdot \mathrm{una}_2 \\
\mathcal{E}(uop\ exp) &= \mathcal{E}(exp) \cdot Prim(uop) \\
\mathcal{E}(exp_1\ bop\ exp_2) &= \mathcal{E}(exp_1) \cdot \mathcal{E}(exp_2) \cdot Prim(bop) \\
\mathcal{E}(exp_0\ ?\ exp_1\ :\ exp_2) &= \mathcal{B}_1(exp_0, \mathrm{if}_1) \cdot \mathcal{E}(exp_2) \cdot Goto(\mathrm{if}_2) \cdot \\
&\quad \mathrm{if}_1 \cdot \mathcal{E}(exp_1) \cdot \mathrm{if}_2
\end{aligned}
$$

Compilation of Java$_{\mathcal{I}}$ expressions for control flow

$$
\begin{aligned}
\mathcal{B}_1(\texttt{true}, lab) &= Goto(lab)\\
\mathcal{B}_1(\texttt{false}, lab) &= \epsilon\\
\mathcal{B}_1(\texttt{!}\ exp, lab) &= \mathcal{B}_0(exp, lab)\\
\mathcal{B}_1(exp_0\,\texttt{?}\,exp_1\ \texttt{:}\ exp_2, lab) &= \mathcal{B}_1(exp_0, \mathrm{if}_1)\cdot\mathcal{B}_1(exp_2, lab)\cdot Goto(\mathrm{if}_2)\cdot\\
&\quad\ \mathrm{if}_1\cdot\mathcal{B}_1(exp_1, lab)\cdot\mathrm{if}_2\\
\mathcal{B}_1(exp, lab) &= \mathcal{E}(exp)\cdot Cond(\texttt{ifne}, lab)
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{B}_0(\texttt{true}, lab) &= \epsilon\\
\mathcal{B}_0(\texttt{false}, lab) &= Goto(lab)\\
\mathcal{B}_0(\texttt{!}\ exp, lab) &= \mathcal{B}_1(exp, lab)\\
\mathcal{B}_0(exp_0\,\texttt{?}\,exp_1\ \texttt{:}\ exp_2, lab) &= \mathcal{B}_1(exp_0, \mathrm{if}_1)\cdot\mathcal{B}_0(exp_2, lab)\cdot Goto(\mathrm{if}_2)\cdot\\
&\quad\ \mathrm{if}_1\cdot\mathcal{B}_0(exp_1, lab)\cdot\mathrm{if}_2\\
\mathcal{B}_0(exp, lab) &= \mathcal{E}(exp)\cdot Cond(\texttt{ifeq}, lab)
\end{aligned}
$$

Compilation of Java$_{\mathcal{I}}$ statements $\mathcal{S}: Stm \to Code$

$$
\begin{aligned}
\mathcal{S}(;) &= \epsilon \\
\mathcal{S}(exp;) &= \mathcal{E}(exp) \cdot Pop(size(\mathcal{T}(exp))) \\
\mathcal{S}(\{stm_1 \ldots stm_n\}) &= \mathcal{S}(stm_1) \cdot \ldots \cdot \mathcal{S}(stm_n) \\
\mathcal{S}(\texttt{if}\ (exp)\ stm_1\ \texttt{else}\ stm_2) &= \mathcal{B}_1(exp, \mathrm{if}_1) \cdot \mathcal{S}(stm_2) \cdot Goto(\mathrm{if}_2) \cdot \\
&\quad\ \mathrm{if}_1 \cdot \mathcal{S}(stm_1) \cdot \mathrm{if}_2 \\
\mathcal{S}(\texttt{while}\ (exp)\ stm) &= Goto(\mathrm{while}_1) \cdot \mathrm{while}_2 \cdot \mathcal{S}(stm) \cdot \\
&\quad\ \mathrm{while}_1 \cdot \mathcal{B}_1(exp, \mathrm{while}_2) \\
\mathcal{S}(lab:\ stm) &= lab_c \cdot \mathcal{S}(stm) \cdot lab_b \\
\mathcal{S}(\texttt{continue}\ lab;) &= Goto(lab_c) \\
\mathcal{S}(\texttt{break}\ lab;) &= Goto(lab_b)
\end{aligned}
$$

Example: Compilation of boolean expressions

Simple compilation:

$$\mathcal{S}(\texttt{if}\ (exp)\ stm_1\ \texttt{else}\ stm_2) =$$
$$\mathcal{E}(exp) \cdot Cond(\texttt{ifne}, \mathrm{if}_1) \cdot \mathcal{S}(stm_2) \cdot Goto(\mathrm{if}_2) \cdot \mathrm{if}_1 \cdot \mathcal{S}(stm_1) \cdot \mathrm{if}_2$$

Example:

```
boolean m(boolean x, boolean y) {
  boolean z;
  if (x && (z = y))
    return z;
  else
    return x;
}
```

Equivalence:

```
x && (z = y)   ≡   x ? z = y : false,
```

Example: Compilation of boolean expressions (continued)

Simple compilation	Compilation with $\mathcal{B}_i$
$Load(\texttt{int}, x)$	$Load(\texttt{int}, x)$
$Cond(\texttt{ifne}, A)$	$Cond(\texttt{ifne}, A)$
$Prim(\texttt{0})$	$Goto(B)$
$Goto(B)$	$A: Load(\texttt{int}, y)$
$A: Load(\texttt{int}, y)$	$Dupx(0, 1)$
$Dupx(0, 1)$	$Store(\texttt{int}, z)$
$Store(\texttt{int}, z)$	$Cond(\texttt{ifne}, L)$
$B: Cond(\texttt{ifne}, L)$	$B: Load(\texttt{int}, x)$
$Load(\texttt{int}, x)$	$Return(\texttt{int})$
$Return(\texttt{int})$	$L: Load(\texttt{int}, z)$
$L: Load(\texttt{int}, z)$	$Return(\texttt{int})$
$Return(\texttt{int})$	

Warning: The simple code is rejected by the bytecode verifer!
(Variable z may not be initialized at label L)