Vocabulary of the trustful JVM$_\mathcal{C}$

Instructions:

$$
\begin{aligned}
\mathit{Instr} = {} & \ldots \\
& \mid \mathit{GetStatic}(\mathit{Type}, \mathit{Class}/\mathit{Field}\,) \\
& \mid \mathit{PutStatic}(\mathit{Type}, \mathit{Class}/\mathit{Field}\,) \\
& \mid \mathit{InvokeStatic}(\mathit{Type}, \mathit{Class}/\mathit{MSig}) \\
& \mid \mathit{Return}(\mathit{MoveType})
\end{aligned}
$$

Universes:

$$
\begin{aligned}
\mathit{MoveType} = {} & \ldots \\
& \mid \texttt{void}
\end{aligned}
$$

Static function:

$$\mathit{cEnv} \colon \mathit{Class} \to \mathit{ClassFile}$$

Class Files

$$\begin{aligned}
\mathit{ClassFile} = \mathit{CFile}(\mathit{classNm} &: \mathit{Class},\\
\mathit{isInterface} &: \mathit{Bool},\\
\mathit{modifiers} &: \mathit{Powerset}(\mathit{Modifier}),\\
\mathit{super} &: \mathit{Class},\\
\mathit{implements} &: \mathit{Powerset}(\mathit{Class}),\\
\mathit{fields} &: \mathit{FieldTab},\\
\mathit{methods} &: \mathit{MethTab})
\end{aligned}$$

Field and Method tables

$$\begin{aligned}\mathit{FieldTab} &= \mathit{Map}(\mathit{Field}, \mathit{FDec})\\ \mathit{MethTab} &= \mathit{Map}(\mathit{MSig}, \mathit{MDec})\end{aligned}$$

$$\begin{aligned}\mathit{FDec} = \mathit{FDec}(&\mathit{modifiers} : \mathit{Powerset}(\mathit{Modifier}),\\ &\mathit{type} : \mathit{Type})\end{aligned}$$

$$\begin{aligned}\mathit{MDec} = \mathit{MDec}(&\mathit{modifiers} : \mathit{Powerset}(\mathit{Modifier}),\\ &\mathit{returnType} : \mathit{Type},\\ &\mathit{code} : \mathit{Instr}^*,\\ &\mathit{excs} : \mathit{Exc}^*,\\ &(\mathit{maxOpd}, \mathit{maxReg}) : (\mathit{Nat}, \mathit{Nat}))\end{aligned}$$

Dynamic state of the $\mathrm{JVM}_{\mathcal{C}}$

Universes:

$$
\begin{aligned}
\mathit{Frame} &= (\mathit{Pc}, \mathit{Map}(\mathit{RegNo}, \mathit{Word}), \mathit{Word}^*, \mathit{Class}/\mathit{MSig}) \\
\mathit{Switch} &= \mathit{Noswitch} \\
&\mid\ \mathit{Call}(\mathit{Class}/\mathit{MSig}, \mathit{Args}) \\
&\mid\ \mathit{Result}(\mathit{Val}) \\
&\mid\ \mathit{InitClass}(\mathit{Class}) \\
\mathit{Args} &= \mathit{Word}^* \\
\mathit{Val} &= \mathit{Word}^*
\end{aligned}
$$

Dynamic functions:

$$
\begin{aligned}
\mathit{meth} &: \mathit{Class}/\mathit{MSig} \\
\mathit{stack} &: \mathit{Frame}^* \\
\mathit{switch} &: \mathit{Switch}
\end{aligned}
$$

Initial state:

$$
\begin{aligned}
\mathit{meth} &= (\mathit{Main}/\texttt{main()}) \\
\mathit{stack} &= [\,] \\
\mathit{switch} &= \mathit{Noswitch}
\end{aligned}
$$

The trustful JVM$_{\mathcal{C}}$

$$trustfulVM_C = trustfulScheme_C(execVM_C, switchVM_C)$$

```
$trustfulScheme_C(execVM, switchVM) =$
  **if** $switch = Noswitch$ **then**
    $execVM(code(pc))$
  **else**
    $switchVM$
```

Trustful execution of JVM$_{\mathcal{C}}$ instructions

```
$execVM_C(instr) =$
  $execVM_I(instr)$
  **case** $instr$ **of**
    $GetStatic(\_, c/f) \rightarrow$ **if** $initialized(c)$ **then**
                                   $opd := opd \cdot globals(c/f)$
                                   $pc \;\;:= pc + 1$
                               **else** $switch := InitClass(c)$
    $PutStatic(\_, c/f) \rightarrow$ **if** $initialized(c)$ **then**
                                   **let** $(opd', ws) = split(opd, size(c/f))$
                                   $globals(c/f) := ws$
                                   $opd := opd'$
                                   $pc \;\;:= pc + 1$
                               **else** $switch := InitClass(c)$
```

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

```
$execVM_C(instr) =$
  **case** $instr$ **of**
    $InvokeStatic(\_, c/m) \rightarrow$ **if** $initialized(c)$ **then**
                                   **let** $(opd', ws) = split(opd, argSize(c/m$
                                   $opd \quad := opd'$
                                   $switch := Call(c/m, ws)$
                                 **else** $switch := InitClass(c)$
    $Return(t) \rightarrow$ **let** $(opd', ws) = split(opd, size(t))$
                                 $switch := Result(ws)$
```

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

```
$switchVM_C =$
  $\textbf{case}\ switch\ \textbf{of}$
    $Call(meth, args) \rightarrow \textbf{if}\ \neg isAbstract(meth)\ \textbf{then}$
                                   $pushFrame(meth, args)$
                                   $switch := Noswitch$
    $Result(res) \rightarrow \textbf{if}\ implicitCall(meth)\ \textbf{then}\ popFrame(0, [\,])$
                           $\textbf{else}\ popFrame(1, res)$
                         $switch := Noswitch$
```

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

```
$switchVM_C =$
  **case** $switch$ **of**
    $InitClass(c) \rightarrow$ **if** $classState(c) = Linked$ **then**
                               $classState(c) := Initialized$
                               **forall** $f \in staticFields(c)$
                                 $globals(c/f) := default(type(c/f))$
                               $pushFrame(c/\texttt{<clinit>}())$
                               **if** $c = \texttt{Object} \lor initialized(super(c))$ **then**
                                 $switch := Noswitch$
                               **else**
                                 $switch := InitClass(super(c))$
```

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

```
$pushFrame(newMeth, args) =$
    $stack := stack \cdot [(pc, reg, opd, meth)]$
    $meth := newMeth$
    $pc := 0$
    $opd := [\,]$
    $reg := makeRegs(args)$

$popFrame(offset, result) =$
    $\textbf{let } (stack', [(pc', reg', opd', meth')]) = split(stack, 1)$
    $pc := pc' + offset$
    $reg := reg'$
    $opd := opd' \cdot result$
    $meth := meth'$
    $stack := stack'$
```

Compilation of Java$_{\mathcal{C}}$ expressions/Statements

$$\begin{aligned}
\mathcal{E}(c.f) &= \mathit{GetStatic}(\mathcal{T}(c/f), c/f)\\
\mathcal{E}(c.f = \mathit{exp}) &= \mathcal{E}(\mathit{exp}) \cdot\\
&\quad \mathit{Dupx}(0, \mathit{size}(\mathcal{T}(\mathit{exp}))) \cdot\\
&\quad \mathit{PutStatic}(\mathcal{T}(c/f), c/f)\\
\mathcal{E}(c.m(\mathit{exps})) &= \mathcal{E}(\mathit{exps}) \cdot\\
&\quad \mathit{InvokeStatic}(\mathcal{T}(c/m), c/m)
\end{aligned}$$

$$\mathcal{E}((\mathit{exp}_1, \ldots, \mathit{exp}_n)) = \mathcal{E}(\mathit{exp}_1) \cdot \ldots \cdot \mathcal{E}(\mathit{exp}_n)$$

$$\begin{aligned}
\mathcal{S}(\texttt{static}\ \mathit{stm}) &= \mathcal{S}(\mathit{stm})\\
\mathcal{S}(\texttt{return};) &= \mathit{Return}(\texttt{void})\\
\mathcal{S}(\texttt{return}\ \mathit{exp};) &= \mathcal{E}(\mathit{exp}) \cdot \mathit{Return}(\mathcal{T}(\mathit{exp}))
\end{aligned}$$