Method Call

Method call with actual parameters $arg_1, \ldots, arg_n$

$$\{arg_1 := t_1, \ldots, arg_n := t_n,\ o := t_o\}\langle o.m(arg_1, \ldots, arg_n);\rangle\phi$$

Where method declaration is `void` $m(T_1\ p_1, \ldots, T_n\ p_n)$

Method Call

Method call with actual parameters $arg_1, \ldots, arg_n$

$$\{arg_1 := t_1, \ldots, arg_n := t_n,\ o := t_o\}\langle o.m(arg_1, \ldots, arg_n);\rangle\phi$$

Where method declaration is $\texttt{void}\ m(T_1\ p_1, \ldots, T_n\ p_n)$

What the rule method-call does:

- **(type conformance of arg_i to T_i guaranteed by JAVA compiler)**
- **for each formal parameter p_i of m**
  declare & initialize new local variable '$T_i\ p_i = arg_i$;'
- **look up implementation class C of m**
  split proof, if implementation not statically computable
- **make concrete call $o.C::m(p_1, \ldots, p_n)$**

Method Body Expand

After processing code that binds actual to formal parameters (symbolic execution of '$T_i\ p_i = arg_i$;')

$$\textbf{METHOD-BODY-EXPAND}\ \frac{\Gamma \ \texttt{==>}\ \langle \pi\ \texttt{method-frame(}C(o)\texttt{)\{}\ b\ \texttt{\}}\ \omega \rangle \phi, \Delta}{\Gamma \ \texttt{==>}\ \langle \pi\ o.C::m(p_1, \ldots, p_n);\ \omega \rangle \phi, \Delta}$$

Method Body Expand

After processing code that binds actual to formal parameters (symbolic execution of '$T_i\ p_i = arg_i;$')

$$\textbf{METHOD-BODY-EXPAND}\ \frac{\Gamma\ \texttt{==>}\ \langle \pi\ \texttt{method-frame(}C(o)\texttt{)\{}\ b\ \texttt{\}}\ \omega \rangle \phi, \Delta}{\Gamma\ \texttt{==>}\ \langle \pi\ o.C::m(p_1, \ldots, p_n);\ \omega \rangle \phi, \Delta}$$

Symbolic Execution

Method Body Expand

After processing code that binds actual to formal parameters (symbolic execution of '$T_i\ p_i = arg_i;$')

$$\textsc{method-body-expand}\ \frac{\Gamma \texttt{ ==> } \langle \pi\ \texttt{method-frame(}C(o)\texttt{)\{}\ b\ \texttt{\}}\ \omega \rangle \phi, \Delta}{\Gamma \texttt{ ==> } \langle \pi\ o.C::m(p_1, \ldots, p_n);\ \omega \rangle \phi, \Delta}$$

Symbolic Execution

Only static information available, proof splitting

Method Body Expand

After processing code that binds actual to formal parameters (symbolic execution of '$T_i\ p_i = arg_i;$')

$$\textsc{method-body-expand}\ \frac{\Gamma \texttt{ ==> } \langle \pi\ \texttt{method-frame(}C(o)\texttt{)\{}\ b\ \texttt{\}}\ \omega \rangle \phi, \Delta}{\Gamma \texttt{ ==> } \langle \pi\ o.C::m(p_1, \ldots, p_n);\ \omega \rangle \phi, \Delta}$$

Symbolic Execution

Only static information available, proof splitting

Runtime infrastructure required in calculus

Method Body Expand

After processing code that binds actual to formal parameters (symbolic execution of '$T_i\ p_i = arg_i;$')

$$\textsc{method-body-expand}\ \frac{\Gamma \texttt{ ==> } \langle \pi\ \texttt{method-frame(}C(o)\texttt{)\{ }b\texttt{ \}}\ \omega \rangle \phi, \Delta}{\Gamma \texttt{ ==> } \langle \pi\ o.C::m(p_1, \ldots, p_n);\ \omega \rangle \phi, \Delta}$$

Symbolic Execution

Only static information available, proof splitting

Runtime infrastructure required in calculus

Demo

`javaDL/method2.key`