Review: Meaning of a Method Call

We can formalize the notion of a (void) method call, using the concept of a stack frame, as follows:

\[ M : \text{Call} \times \text{Method} \times \text{State} \rightarrow \text{State} \]

\[ M(c, f, \sigma) = \text{remove frame}(f \text{params}, f \text{locals}, \]
\[ M(f, \text{body}, \text{ByValue}(f \text{params}, c \text{args}, \]
\[ \text{add frame}(f \text{params}, f \text{locals}, \sigma))) \]

- Remove stack frame after evaluating method (i.e., once it returns)
- Assign method parameters by value
- Add stack frame for the method being called

Friday, 12 Apr. 19

Review: Method Return

- By the time \( A() \) is complete, it can return control back to \( \text{main()} \):

\[ M(A(z), \sigma_4) = \text{remove frame}([z], [1]), \sigma_5 \]

\[ \sigma_5 = \{(x, 0), (y, 1), (w, 4), (1, 5)\} \times \mu_T \times 6 \]

where: \( \mu_T = \{(0, 0), (1, 2), (2, 1), (3, \text{undef}), (4, 1), (5, 2), (6, \text{unused}), \ldots, (n, \text{unused})\} \)

- The call to \( \text{remove frame} \) will \( \text{deallocate} \) memory set aside for \( A() \) and restore local state for \( \text{main()} \):

\[ \sigma_6 = \text{remove frame}([u], [1]), \sigma_7 \]

\[ \sigma_7 = \text{deallocate}(x, i, \sigma_6) \]

\[ = \{(x, 0), (y, 1), (z, 2), (c, 0)\} \times \mu_S \times 4 \]

where: \( \mu_S = \{(0, 0), (1, 2), (2, 1), (3, \text{undef}), (4, \text{unused}), \ldots, (n, \text{unused})\} \)

Wednesday, 17 Apr. 19

Review: Updating the State

- After line 15 of \( \text{main()} \), the program state includes allocated space for the global variables and the locals of \( \text{main()} \) itself:

\[ \sigma_4 = \{(x, 0), (y, 1), (z, 2), (c, 3)\} \times \mu_4 \times 4 \]

where: \( \mu_4 = \{(0, 0), (1, 0), (2, 1), (3, \text{undef}), (4, \text{unused}), \ldots, (n, \text{unused})\} \)

- At line 16, call to void method \( A() \) produces transition:

\[ M(A(z), \sigma_4) = \text{remove frame}([z], [3]), \]
\[ M(f, \text{body}, \text{ByValue}(f \text{params}, f(z) \text{args}, \text{add frame}(f \text{locals}, f(z) \text{args}, \sigma_4)))) \]

\[ = \text{remove frame}([u], [1]), \]
\[ M(f, \text{body}, \text{ByValue}([u], [2], \text{add frame}([w], [1]), \sigma_4))) \]

Friday, 12 Apr. 19

Programming Languages (CS 421/521)
Calling a Non-void method

- The call to the non-void method is largely the same as that for a void method:

  \[ M : \text{Call} \times \text{Method} \times \text{State} \rightarrow \text{State} \]

  \[ M(c, f, \sigma) = \sigma(f, \text{remove frame}(f, \text{params}, \text{f.locals}), \]

  \[ M(f, \text{allocate}(f, \text{ByValue}(f, \text{params}, c, \text{args}), \]

  \[ \text{add frame}(f, \text{f.locals}, \sigma)))))) \]

After execution, assign the value returned by the method to that variable

Extra allocate to create return variable, with same name as the method

Next, a step that we did not do in the void case—allocating the return variable:

\[ M(c, f, \sigma) = \sigma(f, \text{remove frame}(f, \text{params}, \text{f.locals}), \]

\[ M(f, \text{allocate}(f, \text{ByValue}(f, \text{params}, \text{B(args)}, \]

\[ \text{add frame}(f, \text{f.locals}, \sigma)))))) \]

At that point since \( B() \) has no parameters, the call to \( \text{ByValue} \) changes nothing:

\[ \sigma_f = \text{ByValue}([\{\}], \{y, \sigma_a\}) = \sigma_f \]

Calling a Non-void method

We allocate space for the stack-frame of \( B() \):

\[ \sigma_f = \text{add frame}([\{\}], \{y, \sigma_a\}) = \]

\[ \text{allocate}(y, \sigma_f, \text{main.locals}) = \]

\[ \{(x, 0), (y, 1), (y, 4)\} \times \mu_y \times 5 \]

where: \( \mu_y = \{
0, 1, 2, 3, \text{undef}, 4, \text{undef}, 5, \text{undefined}, \ldots, n, \text{undefined}\} \)

This new variable, \( B \), is placed on the stack as usual:

\[ \sigma_{14} = \text{allocate}(B, \sigma_{10}) = \]

\[ \{(x, 0), (y, 1), (y, 4), (B, 5)\} \times \mu_{14} \times 6 \]

where: \( \mu_{14} = \{
0, 1, 2, 3, \text{undef}, 4, \text{undef}, 5, \text{undefined}, 6, \text{undefined}, \ldots, n, \text{undefined}\} \)
Method Execution

We can now evaluate the body of method B() itself:

\[ M(B(), B, \sigma_0) = \sigma(B, \text{remove frame}([], \text{locals})) \]

\[ \sigma_{11} = \{(x, 0), (y, 1), (y, 4), (B, 5)\} \times \mu_{12} \times 6 \]

where: \( \mu_{12} = \{(0, 0), (1, 2), (2, 1), (3, \text{undef}), (4, \text{undef}), (5, \text{undef}), (6, \text{unused}), \ldots, (n, \text{unused})\} \)

Again, this will do arithmetic and assignments as usual:

\[ \sigma_{12} = \{(x, 0), (y, 1), (y, 4), (B, 5)\} \times \mu_{13} \times 6 \]

where: \( \mu_{13} = \{(0, 0), (1, 2), (2, 1), (3, \text{undef}), (4, 3), (5, 3), (6, \text{unused}), \ldots, (n, \text{unused})\} \)

Method Return, I

Since method B() contains a return statement it is also evaluated

In the context of any method f a return has meaning:

\[ M : \text{Return} \times \text{State} \rightarrow \text{State} \]

\[ M(r, \sigma) = \sigma \sqcup \{(f.id, M(r.result))\} \]

That is, we assign the value of the expression following return to the designated method variable:

\[ \sigma_{13} = \sigma_{12} \sqcup \{(f_1(B), 3)\} \]

= \( \sigma_{12} \sqcup \{(5, 3)\} \)

= \( \{(x, 0), (y, 1), (y, 4), (B, 5)\} \times \mu_{13} \times 6 \)

where: \( \mu_{13} = \{(0, 0), (1, 2), (2, 1), (3, \text{undef}), (4, 3), (5, 3), (6, \text{unused}), \ldots, (n, \text{unused})\} \)

Method Return, II

Method B() then returns that value, along with control, back to line 17 of main():

\[ M(B(), B, \sigma_0) = \sigma(B, \text{remove frame}([], \text{locals})) \]

\[ \sigma_{11} = \{(x, 0), (y, 1), (y, 4), (B, 5)\} \times \mu_{13} \times 6 \]

where: \( \mu_{13} = \{(0, 0), (1, 2), (2, 1), (3, \text{undef}), (4, 3), (5, 3), (6, \text{unused}), \ldots, (n, \text{unused})\} \)

This deallocates memory for B(), restores local environment of main() and leads to the final state by assigning the return value to variable c:

\[ \sigma_{12} = \sigma(B, \text{remove frame}([], \text{locals})) \]

\[ = \text{deallocate}(y, \sigma_{11} \cup \text{main_locals}) \sqcup \{(c, 3)\} \]

\[ = \text{deallocate}(y, \sigma_{11} \cup \text{main_locals}) \sqcup \{(3, 3)\} \]

\[ = \{(x, 0), (y, 1), (x2, 2), (c, 3)\} \times \mu_{14} \times 4 \]

where: \( \mu_{14} = \{(0, 0), (1, 2), (2, 1), (3, 3), (4, \text{unused}), \ldots, (n, \text{unused})\} \)

This Week

**Topic:** Semantics of non-void methods; imperative programming languages

**Meet:** usual schedule Wednesday/Friday

**Homework 04:** due Friday, 26 April, 5:00 PM

**Office Hours:** Wing 210

- Tuesday: 3:00 PM – 4:00 PM
- Wednesday: 9:00 AM – 10:30 AM
- Thursday: 2:00 PM – 3:00 PM
- Friday: 9:00 AM – 10:30 AM