CS571

- Notes 18
- Compiled denotations and Program Properties

“Compiling” a program

- Instead of expressing the denotation as a result (from Nat) we can express it as a function of \( n \in \text{Nat} \)
- \( P[Z := 1; \text{if } A = 0 \text{ then diverge}; Z := 3] \)
  = \( \lambda n. \text{let } s = \text{update } [A] n \text{ newstore in} \)
  \( \text{let } s' = E \text{ in} \)
  \( \text{let } s'' = \text{update } Z s' \text{ in} \)
  \( \text{access } [Z] s'' \)
  Where E = \( C[Z := 3]; (\text{if } A = 0 \text{ then diverge}); (C[Z := 1]; s) \)
- i.e.
  \( ((\lambda s.3).((\lambda s'.(\text{access } [A] s') \text{ equals zero} \rightarrow (\lambda s'',\text{update } [Z] s'') s''))) \)

A more readable version

- Convert strict lambdas to lets (same property of strictness)
- \( P[Z := 1; \text{if } A = 0 \text{ then diverge}; Z := 3] \)
  = \( \lambda n. \text{let } s = \text{update } [A] n \text{ newstore in} \)
  \( \text{let } s_1 = \)
  \( \text{let } s_2 = \)
  \( \text{let } s_3 = \)
  \( \text{update } [Z] s_1 \text{ one } s_1 \text{ in} \)
  \( \text{(access } [A] s_2 \text{ equals zero} \rightarrow (\lambda s_3) s_3 \text{ in} \)
  \( \text{update } [Z] \text{ with } s_3 \text{ in} \)
  \( \text{access } [Z] s_3 \)
Simplify the code (optimize)

- Since `newstore` is proper, we can apply the update for `A`, and the first update for `Z`.
- Simplify to:
  \[
  \lambda n. \text{let } s' = \text{let } s_3 = (\text{access}[A] s_2) \text{ equals zero } \rightarrow \text{update}[A] s_2 \text{ in access}[Z] \text{ three } s_2 \text{ in access}[Z] s'
  \]
  - Where \( s_2 = \text{update}[Z] \text{ one (update}[A] n \text{ newstore}) \)
  - We can use (by extensionality of functions):
    let \( s = (e_1 \rightarrow \bot \rightarrow e_2) \) in \( e_3 \) is the same as \( e_1 \rightarrow \bot \square [e_2] e_3 \) to simplify further

Final simplification

- \( \lambda n. \text{let } s' = (n \text{ equals zero } \rightarrow \bot \square \text{ update}[Z] \text{ three } s_2 \) in access}[Z] s'\)
- We can apply the same simplification again:
  \( \lambda n. n \text{ equals zero } \rightarrow \bot \square \text{ access}[Z] \text{ (update}[Z] \text{ three } s_2) \)
- Finally:
  \( \lambda n. n \text{ equals zero } \rightarrow \bot \square \text{ three} \)
- Here all identifiers have been optimized away; stores have been used where they are proper (not bottom); the final result is very intuitive as to the core meaning of the program

Proving program properties

- If two denotations are equal, then the two programs with those denotations are equivalent
- Since denotations can be functions, we need to show that two functions are the same
- Use extensionality principle:
  - If \( f \ x = g \ x \) for all \( x \), then \( f \) and \( g \) are the same function
Program equivalence example

Prove:

\[ \text{X := 0; Y := X + 1 and} \]
\[ \text{Y := 1; X := 0 are equivalent} \]

We can do this if:

\[ \text{P} [ \text{X := 0; Y := X + 1} ] \text{ and} \]
\[ \text{P} [ \text{Y := 1; X := 0} ] \text{ are the same function} \]

Program equivalence example

\[ \text{P}[\text{X := 0; Y := X + 1}] \]
Relies on
\[ \text{C}[\text{X := 0; Y := X + 1}][\text{X} = \text{zero}] \]
\[ \text{= C}[\text{X := 0; Y := X + 1}][\text{X} = \text{zero}] \]
\[ \text{= update}[\text{Y := one}][\text{X} = \text{zero}] \]
\[ \text{= [Y := one][X := zero s]} \]

\[ \text{P}[\text{Y := 1; X := 0}] \]
Relies on
\[ \text{C}[\text{Y := 1; X := 0}][\text{X} = \text{zero}] \]
\[ \text{= C}[\text{Y := 1; X := 0}][\text{X} = \text{zero}] \]
\[ \text{= update}[\text{X := 0}][\text{Y} = \text{one}] \]
\[ \text{= [X := 0][Y := one s]} \]

These two are different functions, but are extensionally equivalent because they produce the same result for \( [X] \) (zero) and for \( [Y] \) (one) and for any other argument \( [I] \) (s(I))