How to handle the loop?

- The goal is to provide a static object as the semantics of a loop
- Recursion gives an operational semantics (good) but is not static (bad)

\[
\lambda s. B[s] \rightarrow C[\text{while } B \text{ do } C][C[s]] s \Downarrow s
\]

Unwanted recursion

Solutions to recursive function equations

- This equation has many solutions
  \[
  q = \lambda n. n \text{ equals zero } \rightarrow \text{one } \downarrow q(n \text{ plus one})
  \]

- E.g.
  \[
  q = \lambda n. n \text{ equals zero } \rightarrow \text{one } \downarrow \\
  q(\text{zero}) = \text{one} \\
  q(\text{one}) = q(\text{two}) \\
  q(\text{two}) = q(\text{three}) \\
  \ldots
  \]

- Additional solutions could be
  \[
  q = \lambda n. n \text{ equals zero } \rightarrow \text{one } \downarrow \text{four} \\
  q = \lambda n. n \text{ equals zero } \rightarrow \text{one } \downarrow \text{twoHundred}
  \]

- In general there is an infinite family of solutions
  \[
  q_k = \lambda n. n \text{ equals zero } \rightarrow \text{one } \downarrow k
  \]
The ‘best’ solution?

- Our goals should be
  - Find at least one solution
  - Find the best solution among many
  - Try to ensure the solution follows operational intuition
- The answer is called fixed point semantics

Factorial

- We will find the best solution to the equation:
  \[ \text{fac} = \lambda n. n \text{ equals zero } \rightarrow \text{ one } \]
  \[ \square n \text{ times } (\text{fac}(n \text{ minus one})) \]

- Consider the family of successive unfoldings

Unfolding factorial

- Zero unfoldings is just fac
- One unfolding is
  \[ \lambda n. n \text{ equals zero } \rightarrow \text{ one } \]
  \[ \square n \text{ times } (\text{fac}(n \text{ minus one})) \]

- Two unfoldings is
  \[ \lambda n. n \text{ equals zero } \rightarrow \text{ one } \]
  \[ \square n \text{ times } \left( (n \text{ minus one}) \text{ equals zero } \rightarrow \text{ one } \right) \]
  \[ \square (n \text{ minus one}) \text{ times } (\text{fac}(n \text{ minus one} \text{ minus one})) \]

Th factorial family

- Use a subscript for the number of unfoldings
  \[ \text{fac}_1 = \lambda n. n \text{ equals zero } \rightarrow \text{ one } \]
  \[ \square n \text{ times } (\text{fac}(n \text{ minus one})) \]

  \[ \text{fac}_2 = \lambda n. n \text{ equals zero } \rightarrow \text{ one } \]
  \[ \square n \text{ times } \left( (n \text{ minus one}) \text{ equals zero } \rightarrow \text{ one } \right) \]
  \[ \square (n \text{ minus one}) \text{ times } (\text{fac}(n \text{ minus one} \text{ minus one})) \]

- Etc.
Factorial behavior

- Write out the (graphs) extensions of each unfolding disallowing the recursive call

\[
\text{graph}(\text{fac}_0) = \{\text{zero, one}\} \\
\text{graph}(\text{fac}_1) = \{\text{zero, one}, \text{one, one}\} \\
\text{graph}(\text{fac}_2) = \{\text{zero, one}, \text{one, one}, \text{two, two}\} \\
\text{graph}(\text{fac}_3) = \{\text{zero, one}, \text{one, one}, \text{two, two}, \text{three, six}\} \\
\text{etc.}
\]

Observations on factorial

- General pattern
  \[\text{graph}(\text{fac}_{i+1}) = \{\text{zero, one}, \text{one, one}, \text{two, two}, \ldots [i, i!]\}\]
- By inspection
  \[\text{graph}(\text{fac}_i) \subseteq \text{graph}(\text{fac}_{i+1})\]
- Also
  \[\text{graph}(\text{fac}_i) \subseteq \text{graph}(\text{factorial})\]
- And then
  \[\bigcup_{i=0}^{\infty} \text{graph}(\text{fac}_i) \subseteq \text{graph}(\text{factorial})\]

Reverse reasoning

- If \([a, b]\) is a member of factorial, then it must be a member of some \(\text{fac}_i\).
- This implies
  \[\text{graph(\text{factorial})} \subseteq \bigcup_{i=0}^{\infty} \text{graph(\text{fac}_i)}\]
- Therefore we must have
  \[\text{graph(\text{factorial})} = \bigcup_{i=0}^{\infty} \text{graph(\text{fac}_i)}\]

Zero unfoldings

- Since
  \[\text{graph}(\text{fac}_0) = \emptyset\]
- We will write
  \[\text{fac}_0 = \lambda n. \bot\]