DINING PHILOSOPHERS IN ANST ACTOR LANGUAGE

\[
\text{bad} = \\text{rec}(\lambda b. \lambda b. \lambda r. \lambda s. \lambda e. \lambda c. \lambda a. \lambda m.
\text{if } (\text{eq?} (m, \text{self}),
\text{if } (\text{eq?} (\text{acks}, \emptyset),
\text{become} (b(2, r, \text{self}, \text{acks} + 1)),
\text{seq} (\text{send}(2, \text{mkrelease} (\text{self})),
\text{send}(r, \text{mkrelease} (\text{self})),
\text{become} (b (2, r, \text{self}, \emptyset)),
\text{send}(l, \text{mkpickup} (\text{self})),
\text{send}(r, \text{mkpickup} (\text{self}))),
\text{become} (b(4, r, \text{self}, \text{acks})))
\]
Dining Philosophers in AMST (2)

\texttt{sync = rec (λb. λh. λw. λm.}

\texttt{if (pickup?(m),}

\texttt{if (eq?(h, nil),}

\texttt{seg(send (phil(m), phil(n)),}

\texttt{become (b(phil(m), nil))),}

\texttt{become (b(h, phil(m))))},

\texttt{if (release?(m),}

\texttt{if (eq? (w, nil),}

\texttt{become (b(nil, nil))),}

\texttt{seg (send (w, w),}

\texttt{become (b(w, nil)))),}

\texttt{become (b(h, w)))))}
Using the definitions to set up a 2-phil2s dining table:

`detactr { f1 := Bfork(nil, nil),
             f2 := Bfork(nil, nil),
             P1 := Bphil(f1, f2, P, Ø),
             P2 := Bphil(f2, f1, P, Ø) } e`

where `e` is defined as:

```
e = seq(send(f1, mkpickup(P1)),
             send(f2, mkpickup(P1)),
             send(f1, mkpickup(P2)),
             send(f2, mkpickup(P2)))
```
DINING PHILOSOPHERS IN AMST (9)

Auxiliary definitions:

\[ mk\text{pickup} = \lambda p. \text{pr}(\text{"pickup"}, p) \]
\[ mk\text{release} = \lambda p. \text{pr}(\text{"release"}, p) \]
\[ \text{pickup?} = \lambda m. \text{if} (\text{ispr?(m)}, \]
\[ \quad \text{eg?} (\text{1st(m)}, \text{"pickup"}, \]
\[ \quad \text{nil}) \]
\[ \text{release?} = \lambda m. \text{if} (\text{ispr?(m)}, \]
\[ \quad \text{eg?} (\text{1st(m)}, \text{"release"}, \]
\[ \quad \text{nil}) \]
\[ \text{phil} = \lambda m. \text{if} (\text{pickup?}(m), \]
\[ \quad \text{2nd}(m), \]
\[ \quad \text{nil}) \]
Reference Cell in Actor Language

\[ B_{cell} = \text{rec}(\lambda b. \lambda c. \lambda m. \]
\[ \quad \text{if } (\text{get?}(m), \]
\[ \quad \quad \text{seg } (\text{become}(b(c)), \]
\[ \quad \quad \quad \text{send}(\text{cust}(m), c)) \]
\[ \quad \text{if } (\text{set?}(m), \]
\[ \quad \quad \text{become}(b[\text{contents}(m)]) \]
\[ \quad \quad \text{become}(b(c(())))) \]

Using the cell:
\[ \text{letactor } \{ a := B_{cell}[θ] \} e \]
\[ e = \text{seg}(\text{send}(a, \text{mkset}(3)), \]
\[ \text{send}(a, \text{mkset}(5)), \]
\[ \text{send}(a, \text{mkget}(e))) \]