Programming Languages (CS 328) **Lecture on 2000-02-03**
Computer Science and Engineering **Soumen Chakrabarti**
Indian Institute of Technology Bombay

The denotational definitions for abstraction and application in FLK are as follows:

$$\left(\mathcal{E}\left[\texttt{(call } E_1\ E_2\texttt{)}\right]\right) = \lambda u.\left(\left(\mathcal{E}\left[E_1\right]u\right)\left(\mathcal{E}\left[E_2\right]u\right)\right) \tag{1}$$

$$\left(\mathcal{E}\left[\texttt{(nlambda I } E\texttt{)}\right]\right) = \lambda u.\lambda e.\left(\mathcal{E}\left[E\right]\begin{array}{|c|}\hline \texttt{I} \to e \\ \hline u \\ \hline\end{array}\right) \tag{2}$$

The environment is shown "stacked up," and looking it up involves searching it from top to bottom for an identifier.

Note that the procedure is closed with the environment u at the point that the procedure is *declared*, not the point(s) where it is *invoked*.

Using the above denotation, we wish to find:

$$\left(\mathcal{E}\left[\begin{array}{l}\texttt{(call (nlambda x (call (nlambda f (call f 2))} \\ \qquad\qquad\qquad\qquad\qquad\texttt{(nlambda y (+ x y))} \\ \qquad\qquad\qquad\quad\texttt{)} \\ \qquad\texttt{)} \\ \qquad\texttt{3)}\end{array}\right]\boxed{\texttt{nullenv}}\right) \tag{3}$$

$$= \left(\left(\mathcal{E}\left[\begin{array}{l}\texttt{(nlambda x (call (nlambda f (call f 2))} \\ \qquad\qquad\qquad\texttt{(nlambda y (+ x y))))}\end{array}\right]\boxed{\texttt{nullenv}}\right)\left(\mathcal{E}\left[\texttt{3}\right]\boxed{\texttt{nullenv}}\right)\right) \tag{4}$$

$$= \left(\lambda e.\left(\mathcal{E}\left[\begin{array}{l}\texttt{(call (nlambda f (call f 2))} \\ \qquad\texttt{(nlambda y (+ x y)))}\end{array}\right]\begin{array}{|c|}\hline \texttt{x} \to e \\ \hline \texttt{nullenv} \\ \hline\end{array}\right)\bar{3}\right) \tag{5}$$

$$= \left(\mathcal{E}\left[\begin{array}{l}\texttt{(call (nlambda f (call f 2))} \\ \qquad\texttt{(nlambda y (+ x y)))}\end{array}\right]\begin{array}{|c|}\hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right) \tag{6}$$

$$= \left(\left(\mathcal{E}\left[\texttt{(nlambda f (call f 2))}\right]\begin{array}{|c|}\hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right)\left(\mathcal{E}\left[\texttt{(nlambda y (+ x y))}\right]\begin{array}{|c|}\hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right)\right) \tag{7}$$

$$= \left(\lambda e_1.\left(\mathcal{E}\left[\texttt{(call f 2)}\right]\begin{array}{|c|}\hline \texttt{f} \to e_1 \\ \hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right)\quad \lambda e_2.\left(\mathcal{E}\left[\texttt{(+ x y)}\right]\begin{array}{|c|}\hline \texttt{y} \to e_2 \\ \hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right)\right) \tag{8}$$

$$= \left(\mathcal{E}\left[\texttt{(call f 2)}\right]\begin{array}{|l|}\hline \texttt{f} \to \lambda e_2.\left(\mathcal{E}\left[\texttt{(+ x y)}\right]\begin{array}{|c|}\hline \texttt{y} \to e_2 \\ \hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right) \\ \hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right) \tag{9}$$

$$\cdots = \left(\lambda e_2.\left(\mathcal{E}\left[\texttt{(+ x y)}\right]\begin{array}{|c|}\hline \texttt{y} \to e_2 \\ \hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right)\bar{2}\right) \tag{10}$$

$$= \left(\mathcal{E}\left[\texttt{(+ x y)}\right]\begin{array}{|c|}\hline \texttt{y} \to \bar{2} \\ \hline \texttt{x} \to \bar{3} \\ \hline \texttt{nullenv} \\ \hline\end{array}\right) \tag{11}$$

$$\cdots = \bar{5}. \tag{12}$$

Notice how the outer environment can hold bindings from names (such as `f`) to closures which in turn have environments in them. Under certain assumptions one can optimize away some of the copies of the environment indicated by the above lines.