CSE 341:
Programming Languages

Spring 2005

Lecture 21 — Exceptions & Continuations

Exceptions in Scheme

Recall exceptions in Java, ML: Transfer control to nearest *dynamically scoped* exception handler (i.e., nearest on "call stack").

Transfer control: Forget what you're doing. Result of entire program is now result of the handle (catch) in the "call stack" that existed when the handler was reached.

Scheme has a *more powerful* concept that can be a little less convenient for exceptions:

- You explicitly indicate what "handler" (*continuation*) to transfer control to.

- You do the transfer via a function application (that does not have function-application semantics)

- The continuation does not even have to be on the "call stack" when it's transferred to!

Continuations for exceptions

Plan:

- Using continuations for exceptions (More details later, time permitting)

Syntax:

- (let/cc k $e1$) : in $e1$, bind k to "current continuation" (basically, the point immediately after the let/cc) then eval $e1$

- (k $e2$): "invoke" continuation bound to k, passing value $e2$, in lieu of the value of $e1$ (now aborted)

Exception idiom:

- Instead of handler, use let/cc

- Pass an appropriate function that invokes k to any function that needs to "raise"