Exceptions and Continuations

Kyle Diodati, Bryce Di Geronimo, Newton Blair

11/23/19

Part I
Control

Control is How we decide what is to be activated next

- i.e. Jumps in assembly language
- loops in C (for, while) allow us to jump to the beginning of a block

Example

```
fun prod [] = 1
|    prod(x::xs) = x * (pred xs)
```

```
xs
acc = 1
for (int i = 0; i ¡ xs.len; i++) {
        if (xs[i] == 0) then break;
        acc *= xs[i];
}
```

Part II
Exceptions

Exceptions are used to handle exceptional cases where a function may not perform as expected

Example

```
fun prode [] = 1
|    prode(0::xs) = raise ProdZero
|    prode(x::xs) = x*(prode xs)

prod[1,2,0,3,4] ->
prode[1,2,0,3,4] ->
1 * (prode[2,0,3,4]) ->
1 * (2 * (prode[0,3,4]) ->
1 * (2 * (raise ProdZero)) ->
1 * raise ProdZero ->          //this step refers to popping off stack
raise ProdZero ->          //again popping off stack
0 ->return

Control 1
Exception Exit Code of int
M handle Exit Code 1 ->E1
|          ExitCode 2 ->E2
```

-Exceptions are special in that they work up the call stack dynamically

-Exception handling follows dynamic link

env |- e:B

raise $\rightarrow \frac{l \text{ is an exception w/ argument of type } B}{env \mid- e \text{ raise } l \text{ e:} A}$

$handler : \frac{env \mid- A \qquad env, x{:}B \mid- t_1 {:}\ A}{env \mid- e \text{ handle } l x => t{:}\ A}$

Part III
Continuations

Continuations represent work that needs to be done after computing calculations
- Continuations require higher order functions

Example

fun prod[]k = k
| prod(0::xs) k = k0
| prod k(x::xs)k = prod k xs(fn v $\rightarrow k(v * x)$)

prodk[1,2,0,3,4](fx x $\rightarrow x$)
$\rightarrow prodk[2,0,3,4](fnv1 \rightarrow fnx \rightarrow x(v1 * 1)$
$prodk[0,3,4](fnv2 \rightarrow (fnv1 \rightarrow (fnx \rightarrow v1 * 1))(v2 * 2))$
$\rightarrow (fnv2 \rightarrow (fnv1 \rightarrow (fnx \rightarrow x)(v1 * 1)(v2 * 2))0$
$\rightarrow (fnv1 \rightarrow (fnx \rightarrow x)(v1 * 1))(0 * 2)$
$\rightarrow (fnx \rightarrow x)(0 * 1)$
$\rightarrow 0$

fun interp(VAR x, env) = env x
| interp(Fun(x, e), env) = CLOD(env, x, e)
| interp(APP(e1, e2)env) =
let v1 = interp(e1, env)
v2 = interp(e2, env)
in
case v1 of
CLOS(env', x, e3) $\rightarrow interp(e3, updateenv'xv2)$
$\Rightarrow RaiseError"typeError"$
end

Convert Above Into Continuation:
interpk(term, env) $\rightarrow (result \rightarrow' r) \rightarrow' r$
$fun\ interpk(varx, env)k = k(envx) : r$

$$
\begin{array}{l}
\mid interpk(FUN(x,E),env)K = K(CLOS(env,x,E)) : r \\
\mid interpk(APP(E_1,E_2),env)k = \\
\quad interpk(E_1,env)(fn\, V_1 => \\
\qquad interpk(E_2,env)(fn\, V_2 => \\
\qquad\quad case\, v_1\, of \\
\qquad\qquad CLOS(env',x,E_3) => interpk(E_3, env'[\frac{v_2}{x}])k \\
\qquad\qquad \_ => raise\, error \\
\qquad\qquad) : r \\
\qquad\quad) : r
\end{array}
$$