CS653: Functional Programming
2017-18 *II*nd Semester

G Machine

Amey Karkare
karkare@cse.iitk.ac.in

http://www.cse.iitk.ac.in/~karkare/cs653
Department of CSE, IIT Kanpur

Agenda

- G Machine (continued)

E is a letrec-expression

$$\begin{aligned}\mathbb{C}[\![\ \text{letrec}\ \mathcal{D}s\ \text{in}\ E_b\ ]\!]\ \rho\ d = {} & \mathbb{C}\mathit{letrec}[\![\ \mathcal{D}s\ ]\!]\ \rho'\ d' \\ & \mathbb{C}[\![\ E_b\ ]\!]\ \rho'\ d' \\ & \text{slide}\ (d' - d)\end{aligned}$$

where

$$\begin{aligned}
\mathcal{D}s &\equiv \begin{bmatrix} x_1 = E_1 \\ \dots \\ x_n = E_n \end{bmatrix} \\
(\rho', d') &= Xr[\![\ \mathcal{D}s\ ]\!]\rho\ d \\
Xr[\![\ \mathcal{D}s\ ]\!]\ \rho\ d &= (\rho[x_1 = d+1, \dots, x_n = d+n], d+n) \\
\mathbb{C}\mathit{letrec}[\![\ \mathcal{D}s\ ]\!]\ \rho\ d &= \text{alloc}\ n \\
& \quad \mathbb{C}[\![\ E_1\ ]\!]\ \rho\ d; \text{update}\ n \\
& \quad \dots \\
& \quad \mathbb{C}[\![\ E_n\ ]\!]\ \rho\ d; \text{update}\ 1
\end{aligned}$$

E is a letrec-expression

- $Xr[\![\ ]\!]$ creates an environment in which $x_1 \ldots x_n$ refer to the n locations above the top of the stack and lengthens the current context by n
- $\mathbb{C}letrec[\![\ ]\!]$ actually allocates n locations and creates pointers pointing to them from the top of stack.
- Then it constructs the graphs corresponding to E_1 to E_n, and overwrites the allocated locations with these graphs.
- Finally E_b is evaluated in the letrec environment and slid into position

Code for Built-in Functions

E is `cons`

$\mathbb{C}[\![\, cons \,]\!] \, \rho \, d$ = pushglobal $cons

- $cons code is

```
cons
update 1
unwind / return
```

- `cons` is the g-machine instruction that pops the top two elements of the stack and makes a cons out of them.
- The root of the redex is updated
- Since unwind will find : which is in WHNF, we can use a return instead.

E is $+$

$\mathbb{C}[\![\, + \,]\!]\ \rho\ d$ = pushglobal $+

- $+ code is

```
push 1
eval
push 1
eval
add
update 3
pop 2
unwind / return
```

E is `head`

$\mathbb{C}[\![\ \textit{head}\ ]\!]\ \rho\ d$ = pushglobal $head

- $head code is

```
eval      -- expose the cons cell
head      -- pick the head
eval      -- evaluate the head before updation
update 1 -- else there will be duplicate
          -- evaluation (see SPJ - section 12.4)
unwind    -- a return is not correct here (why?)
```

E is UNPACK_SUM_1_2

$\mathbb{C}[\![$ UNPACK_SUM_1_2 $]\!]\ \rho\ d$ = pushglobal $UNPACK_SUM_1_2

- $UNPACK_SUM_1_2 code is

```
                -- remember that $UNPACK_SUM_...  is called with
                -- a function f and a value v as arguments
push 1          -- push the value
eval            -- evaluate
testcons        -- test.1.2 actually, tests whether the WHNF
                -- matches the first alternative
jfail L         -- jump on fail to L
push 1          -- push the value once again
SEL_1^2         -- push the second component
push 2          -- push the value yet again
SEL_1^1         -- push the first component
push 2          -- push the function
mkap
mkap
update 3
pop 2
unwind
L: pushfail     -- push the value fail, the previous fail has
                -- been consumed by jfail and the surrounding
                -- FATBAR needs to see a fail.
update 3
pop 2
return
```

E is `if`

$\mathbb{C}[\![\ \mathit{if}\ ]\!]\ \rho\ d$ = pushglobal $if

- $if code is

```
push 0
eval
jfalse L1
push 1
jump L2
L1:  push 2
L2:  update 4
pop 3
unwind
```

E is [] (FATBAR)

$\mathbb{C}[\![\ [] \ ]\!]\ \rho\ d$ = pushglobal $FATBAR

- $FATBAR code is

```
push 0
eval
jfail L1
push 0
jump L2
L1:  push 1
L2:  update 3
pop 2
unwind
```

A large example

Supercombinators:

```
$xxs f x xs = f x :  $map f xs
$map f l    = if l == [] then []
            else unpack.1.2 ($xxs f) l
                 [] error
```

$xxs code

```
globstart $xxs, 3:
push 2              -- xs
push 1              -- f
pushglobal $map
mkap
mkap                -- ($map f x)
push 2              -- x
push 2              -- f
mkap                -- (f x)
pushglobal PACK_SUM_1_2
mkap
mkap                -- f x :  $map f xs
update 4
pop 3
unwind
```

$map code

```
globstart $map, 2
push error
push 2                      -- l
push 2                      -- f
pushglobal $xxs
mkap                        -- ($xxs f)
pushglobal $UNPACK_SUM_1_2
mkap
mkap
pushglobal $FATBAR
mkap
mkap               -- $UNPACK_SUM_1_2 ($xxs f) l
pushglobal $PACK_SUM_2_0  -- Nil
pushglobal $PACK_SUM_2_0
                            -- continued on next slide
```

$map code (continued)

```
push 4
pushglobal $==
mkap
mkap   -- l == []
pushglobal $if
mkap
mkap
mkap   -- if ...  [] error
update 3
pop 2
unwind
```

$\boxed{\text{That's all, folks!}}$