Extended Abstract Machine for Prettyprinting Intermediate Computations

January 11, 2017

1 Compilation Scheme

C(i) = INT(i)
C(b) = BOOL(b)
C(a op b) = C(a); C(b); OP(o)
C(a = b) = C(a); C(b); EQ C($\underline{n}$) = ACCESS(n)
C(λa) = CLOSURE(C(a); RETURN)
C(let a in b) = C(a); LET; C(b); ENDLET
C(a b) = C(a); C(b); APPLY
C(if a then b else c) = C(λb); C(λc); C(a); BRANCH

2 For arithmetic only

Machine state before				Machine state after			
Code	Env	Stack	Print	Code	Env	Stack	Print
INT(i); c	e	s	p	c	e	i.s	-
OP(o); c	e	i.i′.s	p	c	e	o(i, i′).s	p

3 Add lets

e.g `let x = 1 in let y = 2 in x + y` compiles to:

4 Full machine

Machine state before				Machine state after			
Code	Env	Stack	Uncompile	Code	Env	Stack	Print
INT(i); c	e	s	p	c	e	i.s	-
BOOL(b); c	e	s	u	c	e	b.s	-
OP(o); c	e	i.i′.s	p	c	e	o(i, i′).s	p
ACCESS(n); c	e	s	p	c	e	e(n).s	-
LET; c	e	v.s	p	c	v.e	s	-
ENDLET; c	v.e	s	p	c	e	s	-
CLOSURE(c′); c	e	s	p	c	e	c′[e].s	-
APPLY; c	e	v.c′[e′].s	p	c′	v.e′	c.e.s	-
RETURN; c	e	v.c′.e′.s	p	c′	e′	v.s	-