1/3 Sun Aug 26 2007

~~Technically, any non capturing~~

~~But for recursion, any non capturing~~

Any non-capturing clause can be rewritten as a capturing clause + α-conversion:

$$[\![\{ p \to v \}]\!] = \text{let } t_1 = \text{self},\ t_2 = \text{super} \text{ in } [p \to [\![v]\!]\sigma_{\text{self},\text{super}}]$$

∴ ignore non-capturing clauses in the core calculus. Also ignore tuples + other complexity of pattern-matching

core λ: $M, N = \lambda x . M \mid M N \mid x$

core TNG:

$M, N = [P \to M] \mid M N \mid M \oplus N \mid V \mid x$
$P = V \mid x$
$V = .\text{atom}$

~~Evaluation contexts~~

$\mathcal{E}\sigma : M \to V.$ Aha.

exps $M, N = [P \to M] \mid M N \mid M \oplus N \mid L \mid x$
pats $P = L \mid x$
lits $L = .\text{atom}$

values $V, W =$
$[\sigma, P \to M] \mid L \mid V \oplus W \mid \emptyset$

2/3

$\forall V.\ V \equiv V \oplus \emptyset$

~~$\mathcal{E}\Gamma : M \to V.$ eval.~~

~~$\mathcal{E}\Gamma\ [P \to M] = [\Gamma \vdash P \to M]$~~
~~$\mathcal{E}\Gamma\ M\,N = \text{send}(\mathcal{E}\Gamma M, \mathcal{E}\Gamma N)$~~
~~$\mathcal{E}\Gamma\ M \oplus N = \mathcal{E}\Gamma M \oplus \mathcal{E}\Gamma N$~~
~~$\mathcal{E}\Gamma\ L = L$~~
~~$\mathcal{E}\Gamma\ x = V$ if $x \mapsto V$ in Γ~~
~~$= \bot$ otherwise~~

~~$\Gamma = \emptyset \mid x \mapsto V, \Gamma$~~

~~$\text{send}(L \oplus V, W) = \bot$~~
~~send (~~

Forgot self & super!

$M, N = [P \to M] \mid M\,N \mid M \oplus N \mid L \mid x \mid \text{self} \mid \text{super}$
$P = L \mid x$
$L = .\text{atom}.$

$V, W = [\Gamma \vdash P \to M] \mid L \mid \emptyset \mid V \oplus W$

$\Gamma = \emptyset \mid x \mapsto V, \Gamma$ $\qquad \forall v{:}V.\ v \oplus \emptyset \equiv v$

$\overbrace{\mathcal{E}\Gamma_{s_1, s_2}}^{e}\ [P \to M] = \mu x.[\Gamma, \vdash P \to M]$
$M\,N = \text{send}(e M, e N)$
$M \oplus N = e M \oplus e N$
$L = L$
$x = \Gamma x$ orelse $\bot$
$\text{self} = s_1$
$\text{super} = s_2$

3/3.

~~$\text{send} : (V, V) \to V$~~

~~$\text{send}\ (\emptyset \oplus V, \_) \to \bot$~~

~~$\text{send}\ (L \oplus V, \_) \to \bot$~~

~~$\text{send}\ ([\Gamma, s_1, s_2$ [illegible]~~

~~$\text{send}\ ([\Gamma, \vdash L \to M] \oplus V, L) = \mathcal{E}\,\Gamma\, s_1\, s_2\, M$~~

~~$\text{send}\ ([\Gamma, s_1, s_2 \vdash$~~

$\text{send}\ (V_R, V_M) = \text{search}\ V_R\ V_R\ V_M$

$\text{search}\ V_R\ \emptyset \oplus \emptyset\ V_M = \bot$

$\text{search}\ V_R\ L \oplus V_S\ V_M = \text{search}\ V_R\ V_S\ V_M$

$\star\ \text{search}\ V_R\ [\Gamma \vdash L \to M] \oplus V_S\ L = \mathcal{E}\,\Gamma\ V_R\ V_S\ M$

$\star\ \text{search}\ V_R\ [\Gamma \vdash x \to M] \oplus V_S\ V_M = \mathcal{E}\,(x \mapsto V_M, \Gamma)\ V_R\ V_S\ M$

$\text{search}\ V_R\ \_ \oplus V_S\ V_M = \text{search}\ V_R\ V_S\ V_M$

Now, tuples.

$\text{search}\ V_R\ [\Gamma \vdash P \to M]\ V_M = \mathcal{E}\,\Gamma_2\ V_R\ V_S\ M$

where $\Gamma_2 = \Gamma_1 \mathbin{++} \Gamma$

if $\text{match}(P, V_M) = \Gamma_1$

$\emptyset \mathbin{++} \Gamma = \Gamma\ ;\ x \mapsto V, \Gamma_1 \mathbin{++} \Gamma_2 = x \mapsto V, (\Gamma_1 \mathbin{++} \Gamma_2)$

$\text{match}\ (L, L) = \emptyset$

$\text{match}\ (x, V) = x \mapsto V, \emptyset$

$\text{match}\ ((P_1, P_2), (V_1, V_2)) = \Gamma_1 \mathbin{++} \Gamma_2$ if $\text{match}(P_1, V_1) = \Gamma_1$ and $\text{match}(P_2, V_2) = \Gamma_2$

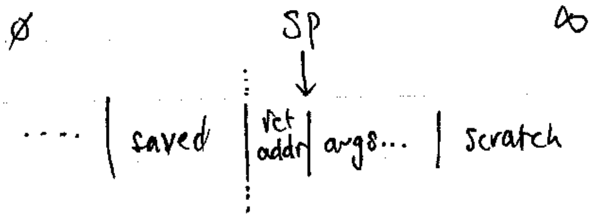
Without tuples, not powerful enough to make simultaneous bindings, ∴ capturing self & super simultaneously not possible — WRONG. Currying + renaming works ok:

[msg = [xself = [xsuper = ...]] self super msg]

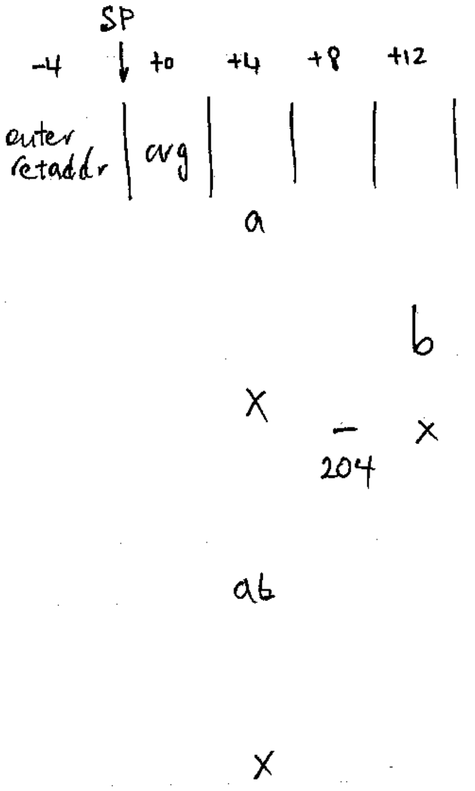
exp := [p → exp ; ⋯]   CLOSE
  | exp exp   SEND
  | exp + exp   EXTEND
  | .atom   LIT
  | var   REF

return regs?	callee save?		
✓		ax	accumulator
	✓	bx	
		cx	argc?
✓		dx	
	✓	si	
	✓	di	
	✓	bp	
	N/A	sp	

(a b) c
a (b c)

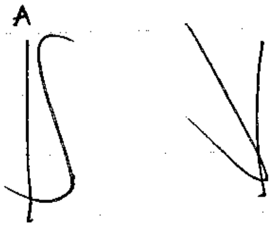


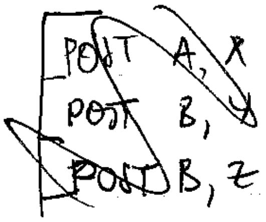
compilation
⟦(a b) c⟧ ⇒

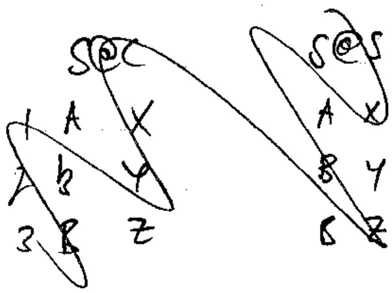


```
0: ⟦a⟧ → result in eax
100: STORE  SP+4        (or +0, arg is dead here
101: ⟦b⟧
200: STORE  SP+12
201: LOAD   SP+4
202: CALL   SP+8
203: liveness vector ?
204: STORE  SP+4
205: ⟦c⟧
300: STORE  SP+0        (tail call)
301: LOAD   SP+4
302: JUMP
```

ERROR code, action, args
DECLARE q, kind, args
DESTROY q
CLONE q, acks, targetq, timeout
PUT q, id, ctype, bbb
MISSING q, acks
INDEX q, acks
LIST q
DELETE q, acks
? REJECT q, acks, reason







A@S
13 S@S/1 — X

S@C

ID	ORIG	TGT	BODY
1	-	A@S	X
2	-	B@S	Y
3	-	B@S	Z

S@S

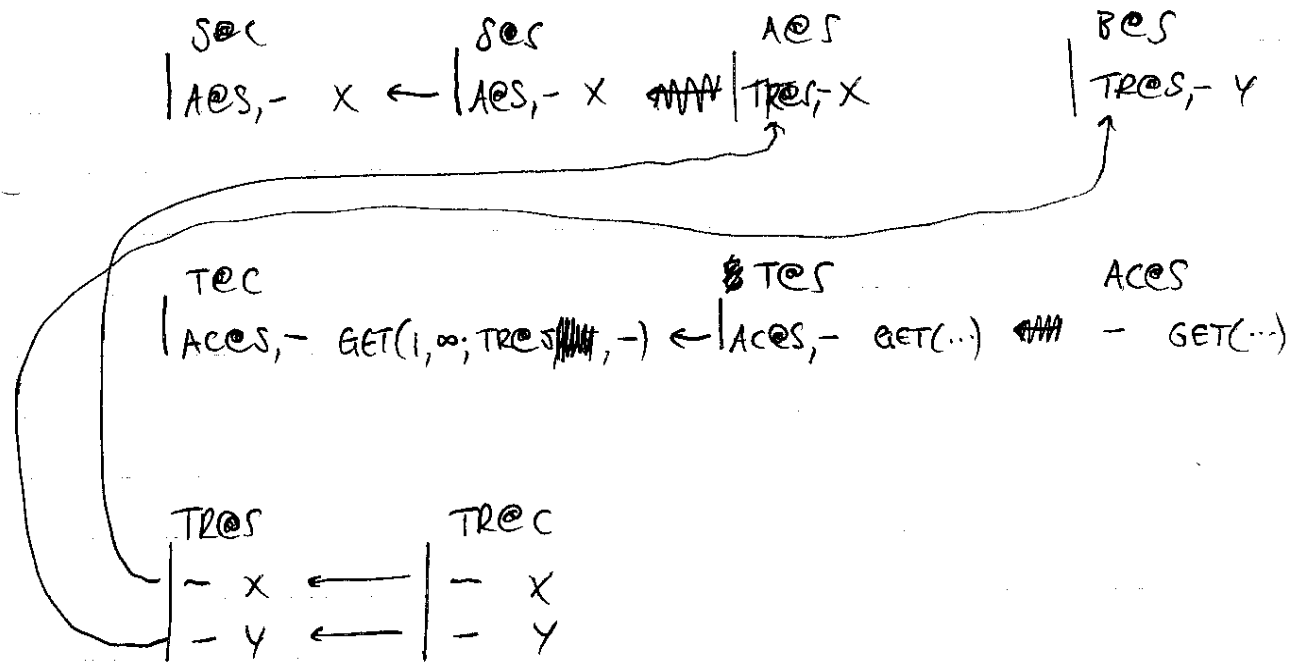
1	S@C/1	A@S	X
2	S@C/2	B@S	Y
3	S@C/3	B@S	Z

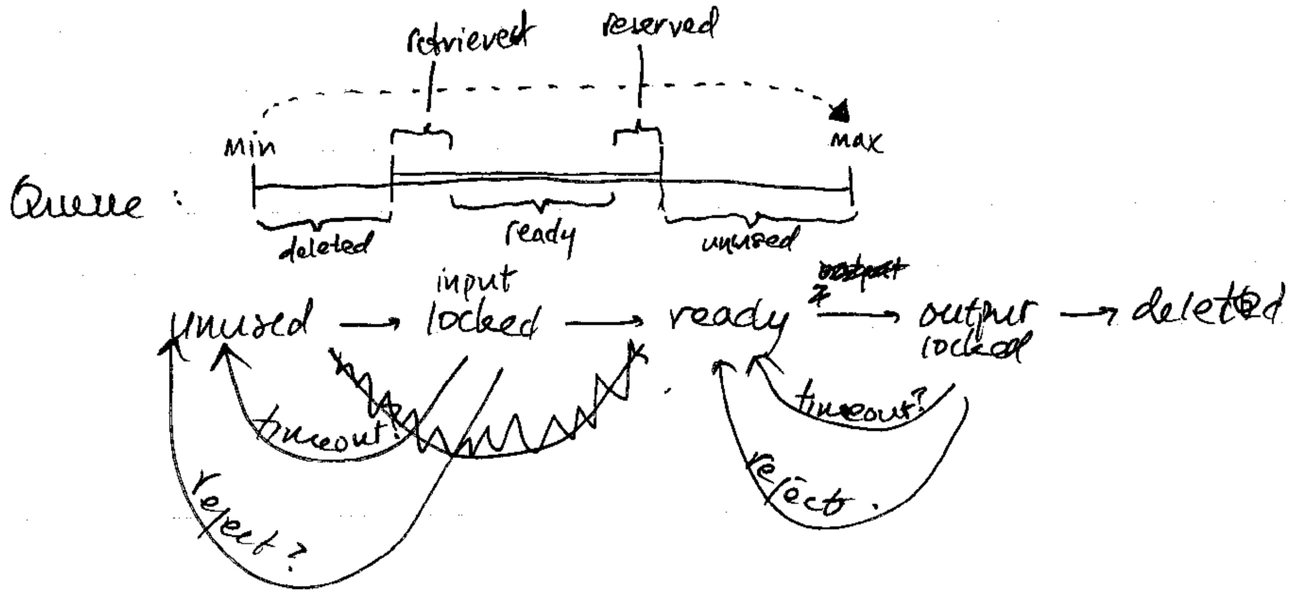
S@C				S@S				A@S		TR@S	
1	–	A@S	X	1	S@C/1	A@S	X	13	S@S/1		X

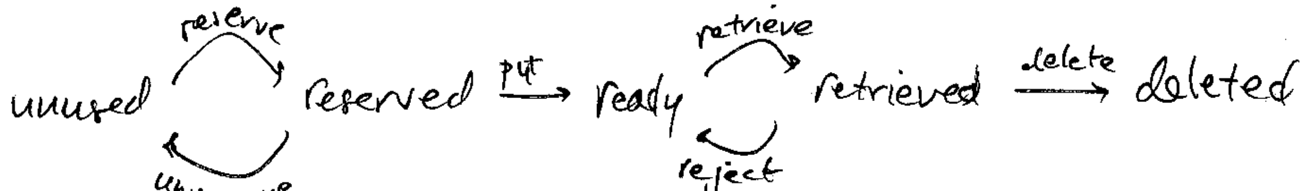
PUT S@C/1 → S@S/1 A@S X
IDX ← S@S {1}

nextslot (A@S) = 13 .
PUT S@S/1 → A@S/13 – X

PUT T@C/1 → T@S/1 AC@S GET(1, ∞, TR@S/15, –)
PUT A@S/13 → TR@S/15 – X


S@C
A@S,- X
S@S
A@S,- X
A@S
TR@S,- X
B@S
TR@S,- Y
T@C
AC@S,- GET(1,∞;TR@S,-)
T@S
AC@S,- GET(…)
AC@S
- GET(…)
TR@S
- X
- Y
TR@C
- X
- Y


Queue :
retrieved
reserved
min
max
deleted
ready
unused
unused
input locked
ready
output locked
deleted
timeout?
reject?
timeout?
reject.


unused
reserve
reserved
put
ready
retrieve
retrieved
delete
deleted
reject