Ghica 4

(semaphores)

$f : com \to nat \to com \vdash$ new s

$$\text{new } x \; f(\text{grab } s;\ x{+}{+};\ \text{rel } s)\ (\text{new } y\ \text{grab } s\ y := !x\ \text{rel } s\ !y) \;=\; \text{new } s\ \text{new } x\ f(\text{grab } s;\ x{-}{-};\ \text{rel } s)\ (\text{new } y\ \text{grab } s\ y := -!x\ \text{rel } x;\ !y).$$

(rel ↓ release semaphore)

$f : com \to nat \to com, x, s \vdash \ldots : com$

parallel unknown interleaving ↘ (replicated by copy-cat strategy)

2	2
g_s	g_s
OK_s	OK_s
r_x	r_x
n_x	m_x
$w(n+1)_x$	rel_s
OK_x	OK_s
O_x	$-m$

← one grab will succeed but not both

2, 0, 2, 0, ⋮ } 0 or more times

$$(Q \cdot 0)^* \cdot (Q' \cdot OK') \cdot (Q \cdot 1)^* \cdots$$

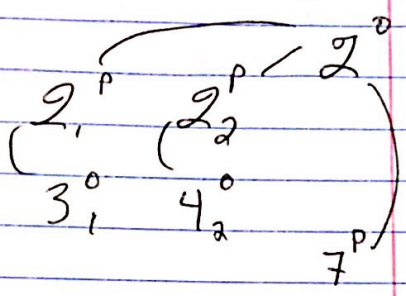
soundness is relatively straightforward

definability (no garbage in semantics)
play $p \to \exists t$ s.t. $[\![t]\!] = strat(p)$

Scanned with CamScanner

$P_{nat_1 \to nat_2 \to nat} \ni 2^Q_1 2^Q_1 3^P_1 2^Q_2 4^P_2 7 \in [\![\lambda x,y.\, x+y]\!]$ when compared w/ 3 4

1) consider only the j.s.



any $Q \to Q$ ⇝ function call (thunk) call-by-name

$Q\ Q\ Q$ ⇝ consider them ∥ parallel

PA ⇝ constants

$\lambda x_1 \lambda x_2 . (x_1 \parallel x_2); 7$

$\in \lambda x_1 \lambda x_2$ new y in $(y := x_1 \parallel y := x_2); 7$

here we have loads of other plays

if anything other than $3, 4$ in threads $_1$, we dont have a response (like div by Θ)

test (x) = if x then skip else $g(s); g(s)$

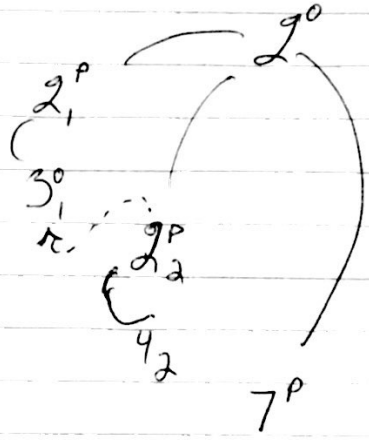
$\lambda x_1 \lambda x_2$. new $\overline{y}$ ($y_1, y_2 \dots y_n$) in $(y_1 := x_1;\ \text{test}(y_1 = 3) \parallel y_2 := x_2;\ \text{test}(y_2 = 4))$; 7

still, can have any interleaving

we can sync P moves on O moves

want sync 2^P_2 after 3^O_1

Scanned with CamScanner



$\lambda x_1 \lambda x_2$. new s in new y in

$$\left(\begin{array}{l} y_1 := x_1; \\ test(y_1 = 3); \\ grab(s); \end{array} \;\Big\|\; \begin{array}{l} release(s); \\ y_2 := x_2; \\ test(y_2 = 4) \end{array} \right); 7$$

have procedure (definability)

any play → term is
least strat invel that play

Applications of GS

Semantic Model (GS)

suitable for application

- precise
- elementary (not simple though)
  simple combinatorial operations
  formalized as automata
  by implementing automata that they represent

Verification – equality in semantics is not decidable
tricks/techniques – restrict language to get decidable equivalence

Compilation (for example by restricting
to not nesting functions
$\cdots f(\cdots f \cdots) -$
get decidability)

Scanned with CamScanner

another way is to approximate the model

CEGAR
counter example guided abstract refinement

effective in practice

Compilation

automata - can be implemented in a variety of ways

- circuits
- distributed
- heterogeneous architectures
- FFI

how do you say FFI does not break language?

answer: look at patterns of call / return

Game semantics

Scanned with CamScanner