Ghica 2

strategy examples

$\sigma_* : nat \to nat \to nat$

(arena w/ 3 components)   time flows downward ↓

$$\begin{array}{ccccc}
\sigma_* : & nat' \to & nat'' \to & nat & \\
 & & & 2^O & \\
 & 2^P & & & \text{(output)} \\
 & n^O & & & \\
 & & 2^P & & \\
 & & m^O & & \\
 & & & p^P = m * n &
\end{array}$$

$2 \cdot 2' \cdot n' \cdot 2'' \cdot m'' \cdot p$

$p = m * n$

prime $'$	first nat
double prime $''$	second nat
(no prime)	result nat

make pointers explicit

$$\begin{tikzcd}
2 & 2' \arrow[l, bend right] & n' \arrow[l, bend right] & 2'' \arrow[lll, bend right] & m'' \arrow[l, bend right] & p \arrow[lllll, bend right]
\end{tikzcd}$$

arena interprets type

· every move points at its enabler

· arrows show causal structure

$$\begin{array}{ccc}
nat & \to & nat \\
2 & \to & 2 \\
\top & & \top \\
n & & n
\end{array}$$

turnstiles

most formal defn (for proper definitions)
encode pointers using names

dangling pointer

$$\overset{\downarrow}{2}^{a<b>} \cdot 2^{'b<c>} \cdot n^{'c<d>} \ldots$$

↑ introduce fresh name (like malloc in C)

↑ not used

$$\hat{\sigma}(p^{O}) = m$$

play ending in O move
Say what next P move is

"automata-like"

Example

$\sigma_{\div} : nat \to nat \to nat$

2^{O}

2^{P}

n^{O}

2^{P}

$\emptyset^{O}$

"checkmate"

unsuccessful termination

$\sigma_* : nat \to nat \to nat$

2^O (under the last nat)

2^P

$\emptyset^O$

$\emptyset^P$ (can shortcut)

game-semantic view of <u>state</u>

adding various constants to λ-calc

these constants may be effectful

may be talking to a device

give direct spec in terms of what happens next

<u>Local state</u>

(not straightforward to deal w/ for algebraic effect)

$\underbrace{\text{int } x \text{ in } M}$ $\overset{\text{def.}}{\equiv}$ $\text{new } (\lambda \overset{\text{var}}{x} . \overset{\text{(unit)}}{M}) \to \text{cmd}$ ↖ command

$\text{new} : (\text{var} \to \text{cmd}) \to \text{cmd}$

read^{Q^O} write^{Q^O} ← (opponent question)

$\dashv \quad 2' \dashv 2$

n^{PA} ok^{PA} ← (proponent answer) $\quad a' \quad a$

$p \cdot \overset{o}{wr}(u) \mapsto ok$

$\underbrace{p} \cdot rd \mapsto \overset{?}{\downarrow}$

have to dig deeper in history

2 cases: ① rd or ② wr

① $p \cdot rd \cdot$ [illegible] $\cdot rd \mapsto n$

② $p \cdot wr(n) \cdot ok \cdot rd \mapsto n$

$2 \mapsto 2'$

$p \cdot a' \mapsto a$

$2 \cdot 2' \cdot rd \mapsto$?

reading unassigned value will cause crash

could do

$\mapsto 0$

(uninitialized value)

accomodates either possibility

game semantics is very operational

but say it in a syntax-free way (specification rather than set eqn's)

Control

.... escape x ....

label x in M : (1 + nat)

option nat

$|||$ def

catch ($\lambda x. M$)

$(\text{cmd} \to \text{nat}) \to (1 + \text{nat})$

cmd	nat	1 + nat
		Q
	Q	
	n (could finish here)	
		inr n

another possible play

cmd	nat	1 + nat
		Q^O
	Q^P	
Q^O		
		inl (*)

done
finished execution
abruptly

- alternating O, P, is defining characteristic of <u>sequential</u> programs
- so far only <u>constant</u> behaviors
- to interpret <u>terms</u> need to look at composing strategies

$$\begin{tikzcd}A \arrow[r, "\sigma"] \arrow[rr, bend right, "\sigma;\tau"'] & B \arrow[r, "\tau"] & C\end{tikzcd}$$

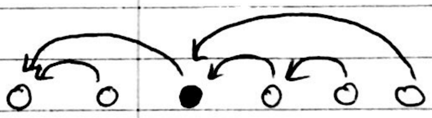
(synchronize on B
hide B)

sync } + hiding
restart }

if τ uses arg several times
need to restart

two operation on traces

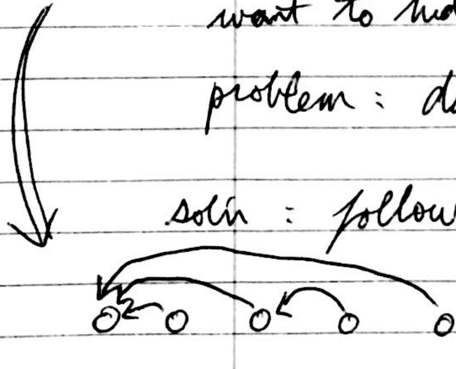
① Hiding



want to hide this

problem: dangling names of ones pointing to it

soln: follow pointer



$p \downharpoonright x$ (p hide x ; p seq moves ; x move)

$= (\rho', \pi)$

new seq. bijection on names $\mathbb{A} \to \mathbb{A}$ pointer reassignment

$\varepsilon \restriction x = \varepsilon$

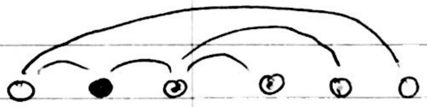
$p \cdot m\, a \langle b \rangle \restriction x = (p' \cdot m\, \pi(a) \langle b \rangle, \pi)$ s.t. $x \not\ni m$

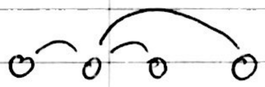
$p \cdot m\, a \langle b \rangle \restriction x = (p', (\pi \mid b \mapsto \pi(a)))$ s.t. $x \ni m$

extend pointer assignment
w/ mapping for b

b reassigned to $\pi(a)$

② Selecting / Threading





transitive closure following pointers

$p \mid X = (p', X')$ $X \leq A$

$\varepsilon \mid X = \varepsilon$

$p \cdot m\, a \langle b \rangle \mid x = (p' \cdot m\, a \langle b \rangle, x \cup \{b\})$ $a \in X$

$p \cdot m\, a \langle b \rangle \mid x = (p', x)$ $a \notin X$

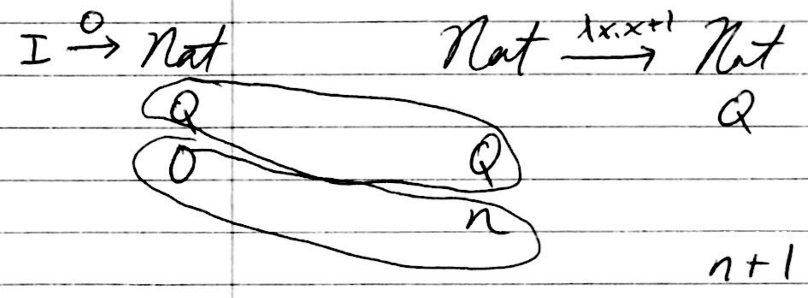
Def synchronization ("interaction")

$\sigma \subseteq J_M \quad , \quad \tau \subseteq J_N$

$$\sigma \underset{M,N}{\|} \tau = \{ p \in {}^{*}J_{M \cup N} \mid p \restriction (M \setminus N) \in \tau \wedge p \restriction (N \setminus M) \in \tau \}$$

$\uparrow$ union (not disjoint union)

Ex



unification of seqs in shared arena

satisfied $\quad n = 0$
$\quad\quad\quad\quad n+1 = 1$

$Q \cdot Q' \cdot 0' \cdot 1$

(difference from CSP - here we extend pointers when hiding)

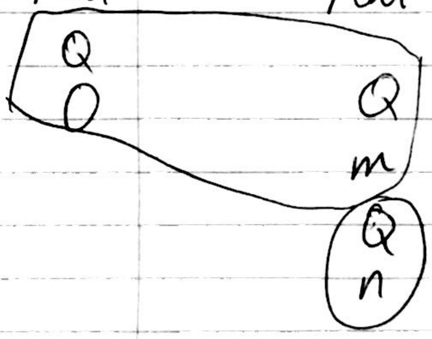
__Def__ ~~Strat~~ Iteration $\sigma \subseteq J_M$ $N \subseteq M$

$$!_N \sigma = \{ \rho \in J_M \mid \forall\ m\ a \langle b \rangle \in \rho \quad m \in N \Rightarrow \rho \upharpoonright \{b\} \in \sigma \}$$

"interleaving σ w/ itself"

– need pointers ~~to~~ see what matches when looking at all possible interleavings

__Ex__ $I \xrightarrow{0} Nat \qquad Nat' \xrightarrow{\lambda x. x+x} Nat$

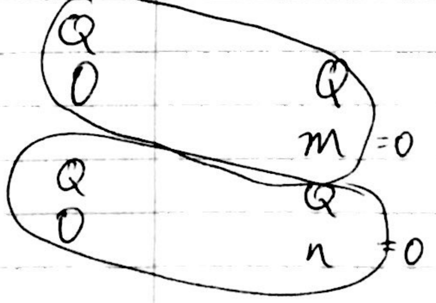


__Def__ Composition $\sigma : A \to B$, $\tau : B \to C$

$$\tau \circ \sigma = \sigma ; \tau = \left(!_{'1_B} \sigma \ \|_{\substack{M_{A \to B} \\ M_{B \to C}}} \ \tau \right) \downarrow M_B$$

Ex.

$I \xrightarrow{0} Nat \qquad Nat \to Nat$



Q: Is composition sensible?

- Is it well-formed $\sigma ; \tau : A \to C$
  - well-justified? Yes
    (use selecting w/ restriction)

Exercises / Lemmas

- $\rho \in P_{A \times B \to C}$ then $\rho \upharpoonright M_A \in P_{B \to C}$
- $\forall\ m a \langle b \rangle \overset{\frown}{\in} \rho\ ,\ n \in I_A$ then $\rho \upharpoonright \{b\} \in P_A$
- $\rho \upharpoonright M_A \upharpoonright M_B = \rho \upharpoonright M_B \upharpoonright M_A$
- $\rho \upharpoonright M_C \upharpoonright \{b\} = \rho \upharpoonright \{b\} \upharpoonright M_C$

(facts of inductive proofs on sequences)

- prefix-closed? ✓
- equivariant ✓ (but careful)

Names in symmetry - book
→ permuting names
α equivalence for names

- $(\sigma;\tau);\nu = \sigma;(\tau;\nu)$ (proof in notes)

  matter of unfolding complicated definitions

- $\sigma \leq \sigma'$ $\quad \nu;\sigma \leq \nu;\sigma'$

  $\sigma;\tau \leq \sigma';\tau$

<u>Identity</u> K

$\forall \sigma : A \to B$

$K_A;\sigma = \sigma$

$\sigma;K_B = \sigma$

communication processes

one is pipe / wire (doesn't do anything)

very simple ; hard to define