Feb 9
Hoffman
pg.1

Parallelism

Parallelism	vs.	Concurrency
evaluation strategy (a sequential)		managing events that happen at the same time.
deterministic		nondeterministic (you can't control when they happen)

scattering is one here

! no shared data.

Lang PPCF : Binary fork-join

$$\text{exp } e ::= \ldots \mid x \mid z \mid s\,e \mid \mathsf{fix}\{\tau\}(x.e) \mid \mathsf{par}(e_1; e_2; x_1.x_2.e) \mid \ldots$$ also lam, app

concr sntx: par $x_1 = e_1$ and $x_2 = e_2$ in e

Statics ...

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \Gamma, x_1 : \tau_1, x_2 : \tau_2 \vdash e : \tau}{\Gamma \vdash \mathsf{par}(e_1; e_2; x_1.x_2.e) : \tau}$$

Sequential dynamics

$$\frac{e_1 \mapsto e_1'}{\mathsf{par}(e_1; e_2; x_1.x_2.e) \mapsto_{sq} \mathsf{par}(e_1', e_2, x_1.x_2.e)}$$

$$\frac{e_1 \text{ val} \quad e_2 \mapsto_{sq} e_2'}{\mathsf{par}(e_1; e_2; x_1.x_2.e) \mapsto_{sq} \mathsf{par}(e_1; e_2'; x_1.x_2.e)}$$

$$\frac{e_1 \text{ val} \quad e_2 \text{ val}}{\mathsf{par}(e_1; e_2; x_1.x_2.e) \mapsto_{sq} [e_1, e_2 / x_1, x_2]\, e}$$

R Ch 3
= CBV.

try CBN; we lose synchronization points

Parallel dynamics

$$\frac{e_1 \mapsto e_1' \quad e_2 \mapsto e_2'}{\mathsf{par}(e_1; e_2; x_1.x_2.e) \mapsto \mathsf{par}(e_1'; e_2'; x_1.x_2.e)}$$

$$\frac{e_1 \text{ val} \quad e_2 \mapsto e_2'}{\mathsf{par}(e_1; e_2; x_1.x_2.e) \mapsto \mathsf{par}(e_1; e_2'; x_1.x_2.e)}$$

& its symm

$$\frac{e_1 \text{ val} \quad e_2 \text{ val}}{\mathsf{par}(e_1; e_2; x_1.x_2.e) \mapsto [e_1, e_2 / x_1 x_2]\, e}$$

Intro 9
Hoffman
9/12

T $e \overset{*}{\underset{par}{\mapsto}} v$ iff $e \overset{*}{\underset{sq}{\mapsto}} v$

PF induction on # of steps

show $e \overset{n}{\underset{sq}{\mapsto}} e'$ implies $e \overset{*}{\underset{par}{\mapsto}} e'$

intermediate steps into $\Downarrow$

show ① $e \overset{*}{\underset{sq}{\mapsto}} v$ iff $e \Downarrow v$

② $e \overset{*}{\underset{par}{\mapsto}} v$ iff $e \Downarrow v$

Eval dyn rule

$$\frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2 \quad [v_1, v_2 / x_1, x_2] e \Downarrow v}{par(e_1; e_2; x_1 \cdot x_2 \cdot e) \Downarrow v}$$

① $e \Downarrow v$ implies $e \overset{*}{\underset{sq}{\mapsto}} v$

PF by rule induction on $e \Downarrow v$ (a judgment is defined w/ rules)

Case Rule for par

then $e = par(e_1; e_2; x_1 \cdot x_2 \cdot e')$ and $e_1 \Downarrow v_1$ and $e_2 \Downarrow v_2$ and $[v_1, v_2 / x_1, x_2] e \Downarrow v$

IH: $e_1 \overset{n_1}{\underset{sq}{\mapsto}} v_1$, $e_2 \overset{n_2}{\underset{sq}{\mapsto}} v_2$, $[v_1, v_2 / x_1, x_2] e' \overset{n_3}{\underset{sq}{\mapsto}} v$

Show $par(e_1; e_2; x_1 \cdot x_2 \cdot e)$

① by induction on n_1 — use IH

$par(e_1; e_2; x_1 \cdot x_2 \cdot e') \overset{n_1}{\underset{sq}{\mapsto}} par(v_1; e_2; x_1 x_2 \cdot e')$

② by induction on n_2 — use IH $par(v_1; e_2; x_1 \cdot x_2 \cdot e') \overset{n_2}{\underset{sq}{\mapsto}} par(v_1; v_2; x_1 \cdot x_2 \cdot e')$

③ $par(v_1; v_2; x_1 \cdot x_2 \cdot e') \overset{\text{use rule}}{\underset{Sq}{\longmapsto}} [v_1\ v_2 / x_1\ x_2] e'$

② (① backwards)

$e \overset{*}{\underset{sq}{\mapsto}} v$ and v val $e \Downarrow v$

HW **PF** show $e \mapsto e'$ and $e' \Downarrow v$ implies $e \Downarrow v$

proofs for $\mapsto_{par}$ are always identical to those for $\mapsto_{sq}$

Intro 9 Hoffman p03

Cost Semantics

Goal: cost semantics $e \Downarrow^k v$ where k describes both sequential and parallel cost,

S Cost graph

$c ::= 1$ unit cost

0 zero cost

D for costs c_1, c_2 $c_1 \otimes c_2$ is parallel combination

$c_1 \oplus c_2$ is seq. combination

D $\text{Work}_{sq} ::=$

$$\begin{cases} wk(1) = 1 \\ wk(0) = 0 \\ wk(c_1 \otimes c_2) = wk(c_1) + wk(c_2) \\ wk(c_1 \oplus c_2) = wk(c_1) + wk(c_2) \end{cases}$$

$\text{Depth} \equiv \text{Work}_{par} ::=$

$$\begin{cases} dp(1) = 1 \\ dp(0) = 0 \\ dp(c_1 \otimes c_2) = \max(dp(c_1), dp(c_2)) \\ dp(c_1 \oplus c_2) = dp(c_1) + dp(c_2) \end{cases}$$

Eval rules

$$\frac{}{z \Downarrow^0 z} \qquad \frac{e \Downarrow^c v}{se \Downarrow^c sv} \qquad \frac{}{\lambda(x:\tau)e \Downarrow^0 \lambda(x:\tau)e} \qquad \frac{[\text{fix}\{\tau\}(x.e)/x]e \Downarrow^c v}{\text{fix}\{\tau\}(x.e) \Downarrow^{c\oplus 1} v}$$

$$\frac{e_1 \Downarrow^{c_1} v_1 \quad e_2 \Downarrow^{c_2} v_2 \quad [v_2/x]e \Downarrow^{c_3} v}{\text{app}(e_1;e_2) \Downarrow^{\bigoplus_{k\in 1..3} c_k} v}$$

$$\frac{e_1 \Downarrow^{c_1} v_1 \quad e_2 \Downarrow^{c_2} v_2 \quad [v_1/x_1, v_2/x_2]e \Downarrow^{c_3} v}{\text{par}(e_1;e_2;x_1.x_2.e) \Downarrow^{(c_1 \otimes c_2)\oplus c_3 \oplus 1} v}$$

the extra 1 here

T (a) If $e \Downarrow^c v$ then $e \overset{wk(c)}{\underset{sq}{\longmapsto}} v$ and $e \overset{dp(c)}{\underset{par}{\longmapsto}} v$

(b) if $e \overset{w}{\underset{sq}{\longmapsto}} v$ then $e \Downarrow^c v$ for some cost graph c where $wk(c) = w$

(c) if $e \overset{d}{\underset{par}{\longmapsto}} v$ then $e \Downarrow^c v$ for some c where $dp(c) = d$

Hoffman pg. 4

Bounded Implementations

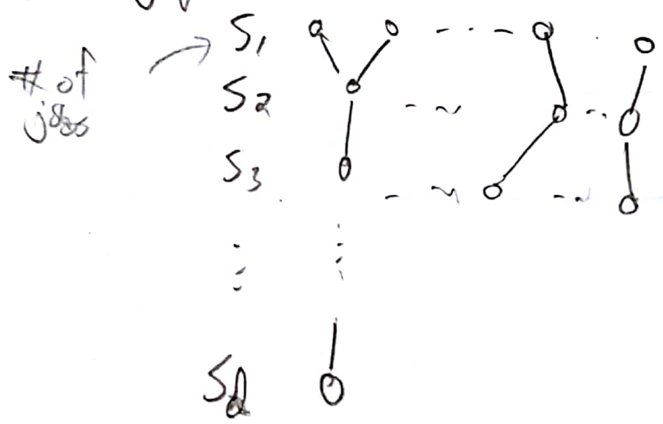
Brent's theorem – the prototypical result

machine model : shared-memory multiprocessor (SMP)

- fixed $p > 0$
- shared memory w constant time access
- constant time synchronization mechanism

T: if $e \Downarrow^{c} v$ where $wk(c) = w$ and $dp(c) = d$ then e can be evaluated on an SMP in time $O(\frac{w}{p} + d)$

cost graphs



$$\sum_{k=1}^{d} \left\lceil \frac{S_k}{p} \right\rceil \le \sum_{k=1}^{d} \left(\frac{S_k}{p} + 1 \right) = \frac{\left(\sum_{k=1}^{d} S_k \right)}{p} + d = \frac{w}{p} + d$$

D: **big O**

$f \in O(g)$ iff $\exists n_0$ and c s.t. $\forall n \geq n_0$ $f(n) \leq c \cdot g(n)$

Research Mtg	Cost Models on λ	or Church Calculus: for other Turing machine

parallel semantics are sequential semantics re

rewrite PCT in Complexity theory

Book: handbook of theoretical CS Peter Van Emde Boas – Chapter 1. machine models & simulations

Complexity team : abstract machine - based cost models

λ Calc : no, language-based.

infinite vs. logarithmic?

RAM

SRAM (succ, pred)

RAM (add, mult)

MRAM (add, sub, mult)

LRAM (log length of words)

RAM-L ( )

log is what you need to express n a register

Parallel machine models

PSPACE

circuit models

TM w alternation

⋮

two parts to a model

the model

- well-defined semantics
- simple
- Close to programming paradigm

part two

Simulation

- Mapping of costs
- good bounds when simulating

"C is assembly for RAM"

lang-based cost model – CM based on a cost semantics instead of a machine

Church's untyped LC is parallelizable

turing machines are not.

CBV λC (w arrays)

Blelloch | par

CBV λC

$e = x \mid e_1\ e_2 \mid \lambda x. e$

$e \Downarrow v$ relation

$\lambda x.e \Downarrow \lambda x.e$ Lam

$$\frac{e_1 \Downarrow \lambda x.e \qquad e_2 \Downarrow v \qquad e[v/x] \Downarrow v'}{e_1\ e_2 \Downarrow v'}\ \text{APP}$$

fully evaluate either of these.

but both have to be finished before substitution

CBV λC is parallel
CBN is not (lazy)

work W sequential work

Span D parallel depth.

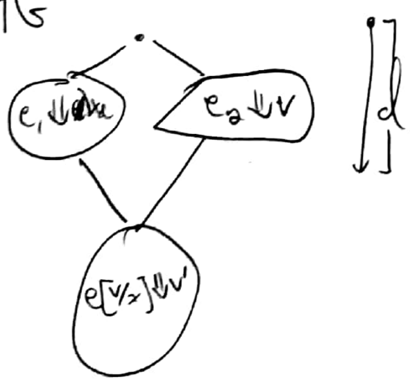
$e \Downarrow v; w, d$

- Span captures dependence depth

$\lambda x.e \Downarrow \lambda x.e; 1, 1$ (Lam)

$$\frac{e_1 \Downarrow \lambda x.e; w_1, d_1 \qquad e_2 \Downarrow v; w_2, d_2 \qquad e[v/x] \Downarrow v'; w_3, d_3}{e_1\ e_2 \Downarrow v'; 1 + \sum_{k=1}^{3} w_k,\ 1 + \max(d_1, d_2) + d_3}\ \text{(APP)}$$

- DAG



work matters more than span.

- let, letrec, datatypes, tuples, case-statements, all implemented w constant overhead.
- integers not simulated in constant overhead, but in log — church numerals in CBV

Simulate (sequential)

CEK machine : a state transition system

$(C, E, K) \Rightarrow (C', E', K')$

different options to simulate.

"control" $C ::= e_1 e_2 \mid \lambda x. e \mid x$

"environment" $E ::= x \to v$

"continuation" $K ::= \text{done} \mid \text{arg}(e, E, K) \mid \text{fun}(e, E, K)$

~~CEK~~

$(e_1 e_2, E, K) \Rightarrow (e_1, E, \text{arg}(e_2, E, K))$

$(x, E, K) \Rightarrow (E(x), E, K)$

lookup

$(v, E, \text{arg}(e, E', K)) \Rightarrow (e, E', \text{fun}(v, E, K))$

insertion might not be constant time.

$(v, E, \text{fun}(\lambda x. e, E', K)) \Rightarrow (e, E' + (x \to v), K)$

we have no hash tables.

log in size of environment.

env size = distinct # of variable names.

PCEK is another state-transition system

$$\langle (c_1, \mathcal{E}_1, k_1), (c_2, \mathcal{E}_2, k_2), \ldots \rangle \Rightarrow \langle (c_1', \mathcal{E}_1', k_1'), (c_2', \mathcal{E}_2', k_2'), \ldots \rangle$$

Simulation bounds

FPCA95 if $e \Downarrow v; w, d$ then v can be calculated from e on a CREW PRAM w p processors in $O\left(\frac{w}{p} + d \log p\right)$

sometimes this is plus.

can't really do better than $\max\left(\frac{w}{p}, d\right)$

if $\frac{w}{p} > d \log p$ then "work dominates"

"$\frac{w}{d}$" is called "the parallelism"

a QSort in λC

the recursive QSort is implicitly parallel

```
fun QSort S =
  if (size S) <= 1 then S
  else
```

QSort on trees is $\log^2$ span

Span on each call is $O(\lg n)$ and there are $O(\lg n)$ calls,

work is $O(n \lg n)$

$$\text{parallelism} = O\left(\frac{n \lg n}{\lg^2 n}\right) = O\left(\frac{n}{\lg n}\right)$$

	work	span
seq	+	+
par	+	max

Recurrence for div & conquer

$$w(n) = 2w\left(\frac{n}{2}\right) + w_{split}(n) + w_{join}(n)$$

$$S(n) = S\left(\frac{n}{2}\right) + S_{split}(n) + S_{join}(n)$$