Parallellism and Concurrency

Parallellism: Evaluate on multiple processors to speed up computation

Concurrency: Use multiple threads sharing resources (may/may not be parallel)

$$e ::= \dots \mid e \| e$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 \| e_2 : \tau_1 \times \tau_2} \qquad \frac{e_1 \mapsto e_1'}{e_1 \| e_2 \mapsto e_1' \| e_2} \qquad \frac{e_2 \mapsto e_2'}{e_1 \| e_2 \mapsto e_1 \| e_2'}$$

"inter leaving"

$$\frac{e_1 \text{ val} \quad e_2 \text{ val}}{e_1 \| e_2 \mapsto (e_1, e_2)}$$

Why not $\dfrac{e_1 \mapsto e_1' \quad e_2 \mapsto e_2'}{e_1 \| e_2 \mapsto e_1' \| e_2'}$?

$$\left.\begin{aligned} 1+2 \| 3+4 &\mapsto 3 \| 3+4 \mapsto 3 \| 7 \\ &\mapsto 1+2 \| 7 \mapsto 3 \| 7 \end{aligned}\right\} \text{Same answer! (always true for STLC)}$$

$$\frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2}{e_1 \| e_2 \Downarrow (v_1, v_2)}$$ - Can't capture diff. interleavings but that's OK.

"Nested" parallellism

```
fix fib = λn. if n ≤ 1 then n
                else
                  let p = fib (n-1) || fib (n-2)
                  in
                  (fst p) + (snd p)
```

What about IMP?

$s ::= x := e \mid \text{if } e \text{ then } s \text{ else } s \text{ fi} \mid \text{while } e \text{ do } s \text{ od}$
$\mid s; s \mid \text{skip} \mid s \| s$

$$\frac{\langle s_1, \sigma\rangle \mapsto \langle s_1', \sigma'\rangle}{\langle s_1 \| s_2, \sigma\rangle \mapsto \langle s_1' \| s_2, \sigma'\rangle} \qquad \frac{\langle s_2, \sigma\rangle \mapsto \langle s_2', \sigma'\rangle}{\langle s_1 \| s_2, \sigma\rangle \mapsto \langle s_1 \| s_2', \sigma'\rangle}$$

$$\frac{}{\langle \text{skip} \| \text{skip}, \sigma\rangle \mapsto \langle \text{skip}, \sigma\rangle}$$

$$\langle x := x+1 \| x := x*2, \{x=1\}\rangle \left.\begin{array}{l} \mapsto^* \langle \text{skip} \| x := x*2, \{x=2\}\rangle \mapsto^* \langle \text{skip}, \{x=4\}\rangle \\ \mapsto^* \langle x := x+1 \| \text{skip}, \{x=2\rangle \mapsto^* \langle \text{skip}, \{x=3\}\rangle \end{array}\right\rangle !$$

Except there are also more!

$$\begin{array}{l} \langle x := x+\overline{1} \| x := x*\overline{2}, \{x=1\}\rangle \\ \mapsto \langle x := \overline{1}+\overline{1} \| x := x*\overline{2}, \{x=1\}\rangle \\ \mapsto \langle x := \overline{1}+\overline{1} \| x := \overline{1}*\overline{2}, \{x=1\}\rangle \\ \mapsto^* \langle x := \overline{2} \| x := \overline{2}, \{x=1\}\rangle \\ \mapsto^* \langle \text{skip}, \{x=2\}\rangle \end{array}$$

$$\begin{array}{l} \langle (\text{while } x \text{ do skip od}) \| x := 1, \{x=0\}\rangle \\ \mapsto^* \ldots \qquad\qquad\qquad \text{or } \mapsto^* \ldots \text{ (forever)} \\ \mapsto^* \langle \text{while } x \text{ do skip od} \| \text{skip}, \{x=1\}\rangle \\ \mapsto^* \langle \text{skip}, \{x=1\}\rangle \end{array}$$

A more realistic model

$s ::= \dots \mid \overset{x:=}{}\text{spawn } s \mid \text{wait}(a)$

$e ::= \dots \mid a$

thread "name"

$$\frac{a \text{ fresh} \quad \text{not used before}}{\langle \cancel{x:=}\text{spawn } s, \sigma \rangle \mapsto \langle \overset{x:=a}{\cancel{\text{skip}}}, \sigma[a \mapsto s] \rangle}$$

$$\frac{\sigma(a) = \text{skip}}{\langle \text{wait}(a), \sigma \rangle \mapsto \langle \text{skip}, \sigma \rangle}$$

(*) If $\sigma(a) \neq \text{skip}$, wait$(a)$ is stuck

$$\frac{\langle s_t, \sigma \rangle \mapsto \langle s_t', \sigma' \rangle}{\langle s, \sigma[a \mapsto s_t] \rangle \mapsto \langle s, \sigma'[a \mapsto s_t'] \rangle}$$

Why is the main thread special?

Instead of $\langle s, \sigma \rangle$, start with $\sigma[\text{main} \mapsto s]$

$$\frac{\langle s, \sigma \rangle \mapsto \langle s', \sigma' \rangle}{\sigma[a \mapsto s] \mapsto \sigma'[a \mapsto s']}$$

– Choose any thread

"Actual" parallelism – run n threads at once

$$\frac{\forall i \in [1,n].\ \langle s_i, \sigma \rangle \mapsto \langle s_i', \sigma_i' \rangle \qquad n \leq \underset{\text{\# of processors}}{P} \qquad \sigma' = \underset{\text{resolve data races}}{\text{merge}}(\sigma_1', \dots, \sigma_n')}{\sigma[a_1 \mapsto s_1] \cdots [a_n \mapsto s_n] \mapsto \sigma'[a_1 \mapsto s_1] \cdots [a_n \mapsto s_n']}$$

This model subsumes nested parallelism

$s_1 \| s_2 \triangleq \overset{x:=}{\text{spawn}}\ s_1 ; s_2 ; \text{wait}(x)$

And more:

```
while 1
do
  client := listen();
  _ := spawn (handle_conn client)
od
```

(*) Progress needs to change!

Local progress: If $\Gamma \vdash \sigma$ and $\Gamma \vdash s$ ok then $s = \text{skip}$ or $\langle s, \sigma \rangle \mapsto \langle s', \sigma' \rangle$

<u>or</u> $s = \text{wait}(a)$ and $\sigma(a) \neq \text{skip}$

So then do we know that $\langle \sigma(a), \sigma \rangle \mapsto \langle s_a', \sigma' \rangle$? Maybe not!

```
X := spawn skip          * Dummy thread to bind X
y := spawn (wait (x))
z := spawn (wait (y))    Could deadlock!
```