S4.3: Expression Evaluation: Parallel Evaluation

CSci 2041:

Advanced Programming Principles

University of Minnesota,
Prof. Van Wyk,
Spring 2018

Exercise #1:An example, two concurrent processes.

Process 1	Process 2
1. `y = x;`	4. `z = x;`
2. `y = y + 1;`	5. `z = z + 1;`
3. `x = y;`	6. `x = z;`

Assume that global variable `x` begins with the value `1`.

What is its value after both threads of control finish for each of the following sequences?

- 1, 2, 3, 4, 5, 6.
- 4, 5, 6, 1, 2, 3.
- 1, 4, 2, 5, 3, 6.

Exercise #2:Relation to for loops

How do `map` and `fold` capture common loop idioms?

Write the loop that implements each of the following. Use `[ ]` to access a element of `a` or `b`. Assume `a.size` is the size of the array `a` (same for `b`).

```
(* b = map f a; *)
```

```
(* sum = fold (+) 0 a *)
```

Automatic parallelization is difficult when problems are specified in this way.

Exercise #3:Simplifications

How would we write the following with these restrictions?

```
fold (fun x y -> x + y)
  (map (fun x -> x * x) (mkArray (fun x -> x) 100))
```

Exercise #4:Restricted OCaml to C

How might we translate the following to C?

```
let plus x y = x + y
let square x = x * x
let id x = x
let main () =
  let a1 = mkArray id 100 in
  let a2 = map square a1 in
  let v = fold plus a2 in
  let () = print_endline v in
  ()
```

We'll do these one at a time and discuss them.

Exercise #5:Map in Cilk

How might we translate

```
let a2 = map square a1 in
```

to C code that uses Cilk?

Exercise #6:Map in Cilk

How might we translate

```
let v = fold plus a2 in
```

to C code that uses Cilk?