Automatic Parallelization of Side-Effecting Higher-Order Scheme Programs

Jens Nicolay, Coen De Roover,
Wolfgang De Meuter, Viviane Jonckers

Software Languages Lab

Vrije Universiteit Brussel

Why?

- design parallel programs or parallelize sequential programs?
- parallelization ≈ garbage collection?

Research Question

How far can we get by using static analysis for brute-force automatic parallelization?

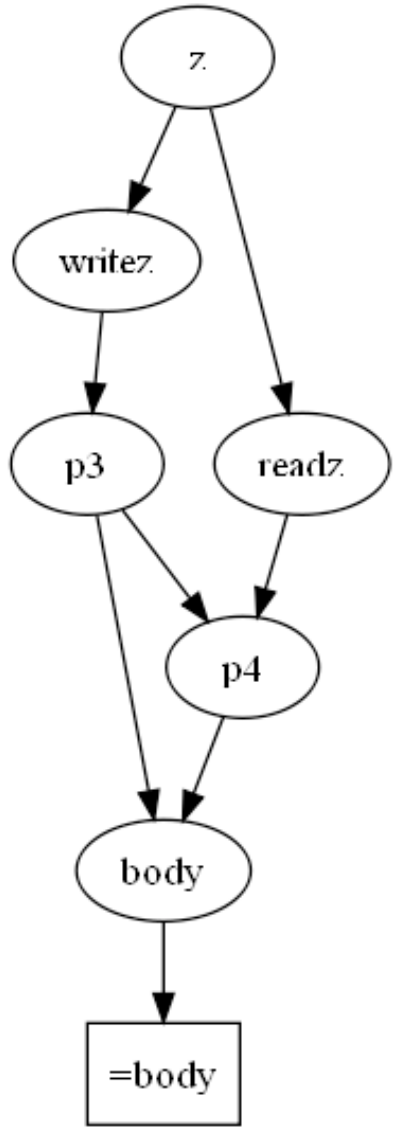
1. Convert into ANF

```
(if (< n 2)
    n
    (+ (fib (- n 1)) (fib (- n 2))))
```

```
(let ((p0 (< n 2)))
  (if p0
      n
      (let ((p1 (- n 1)))
        (let ((p2 (fib p1)))
          (let ((p3 (- n 2)))
            (let ((p4 (fib p3)))
              (+ p2 p4))))))))
```

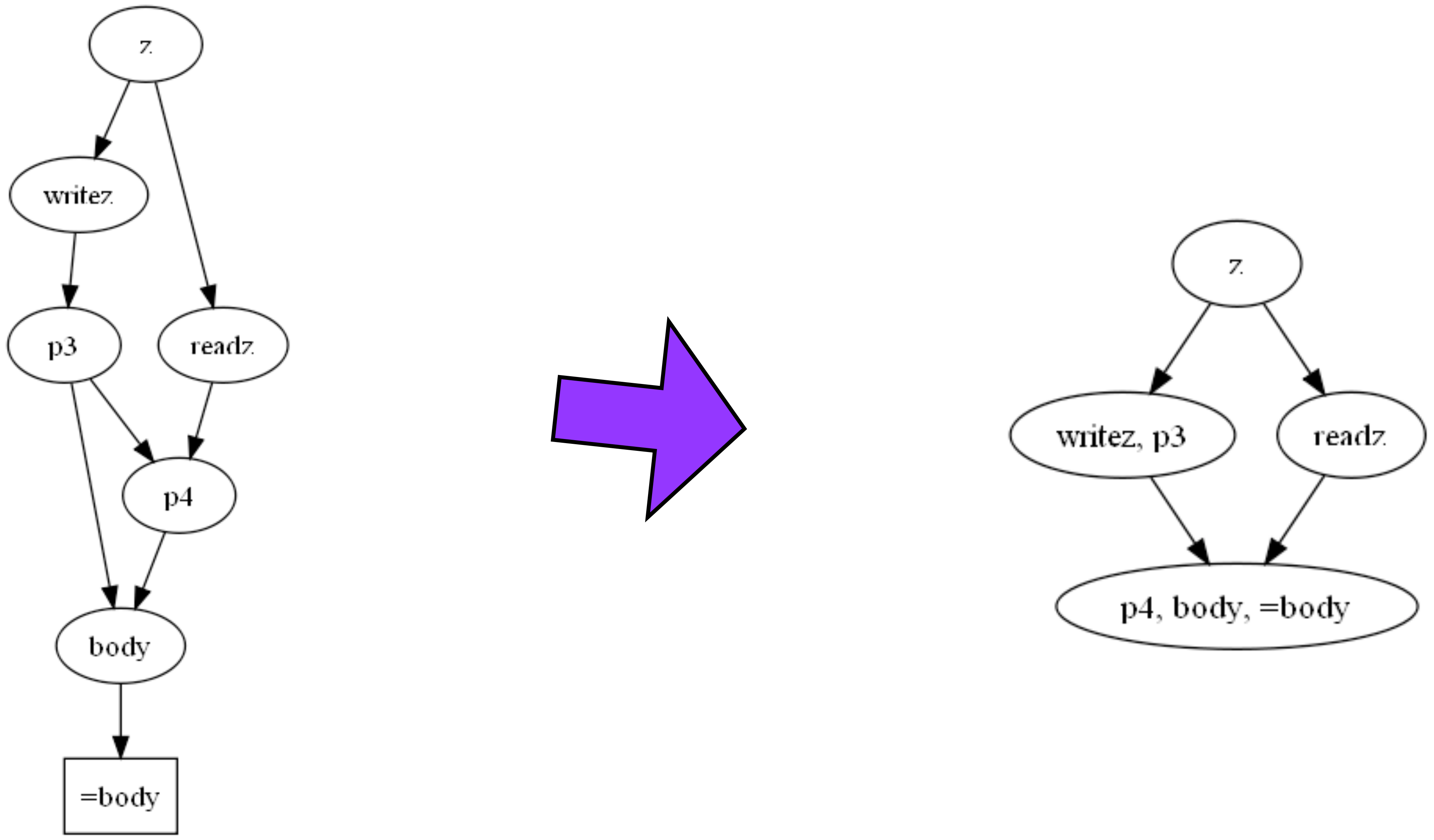
2. Create dependency graph for `lets`

```
(let ((z 0))
  (let ((writez (lambda () (set! z 123))))
    (let ((readz (lambda () z)))
      (let ((p3 (writez)))
        (let ((p4 (readz)))
          (cons p3 p4))))))
```



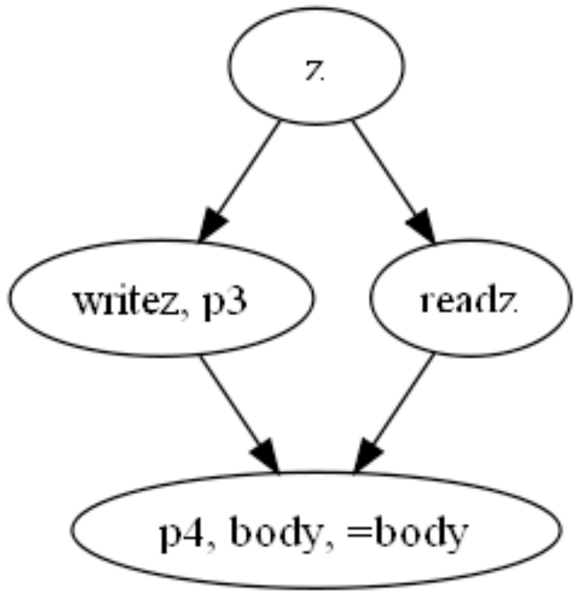
Interprocedural Dependence Analysis of Higher-Order Programs via Stack Reachability – Might&Prabhu (2009)

3. Rewrite graph



pruning edges and grouping vertices brings out optimal binding order

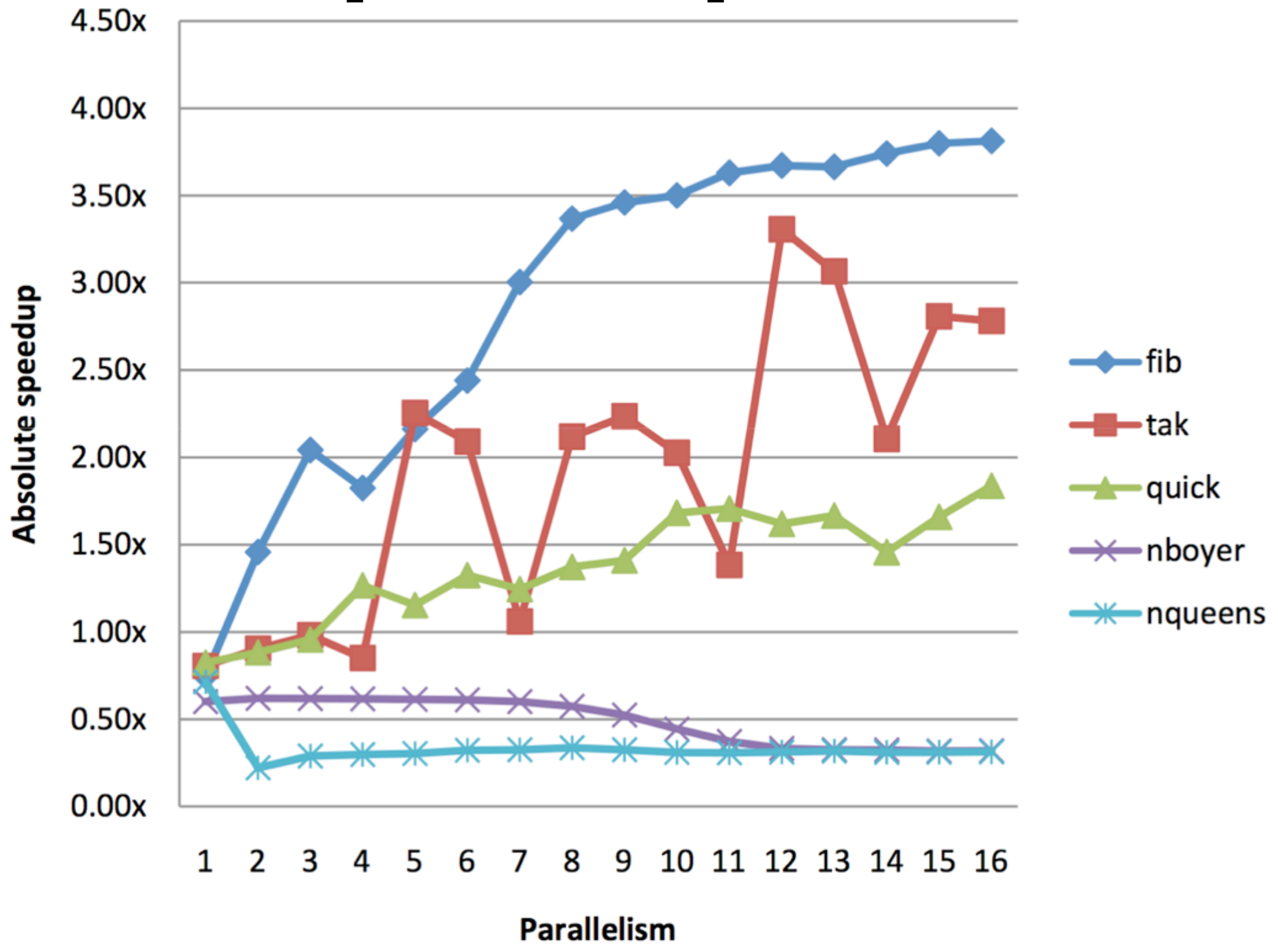
4. Generate code



introduce future and touch

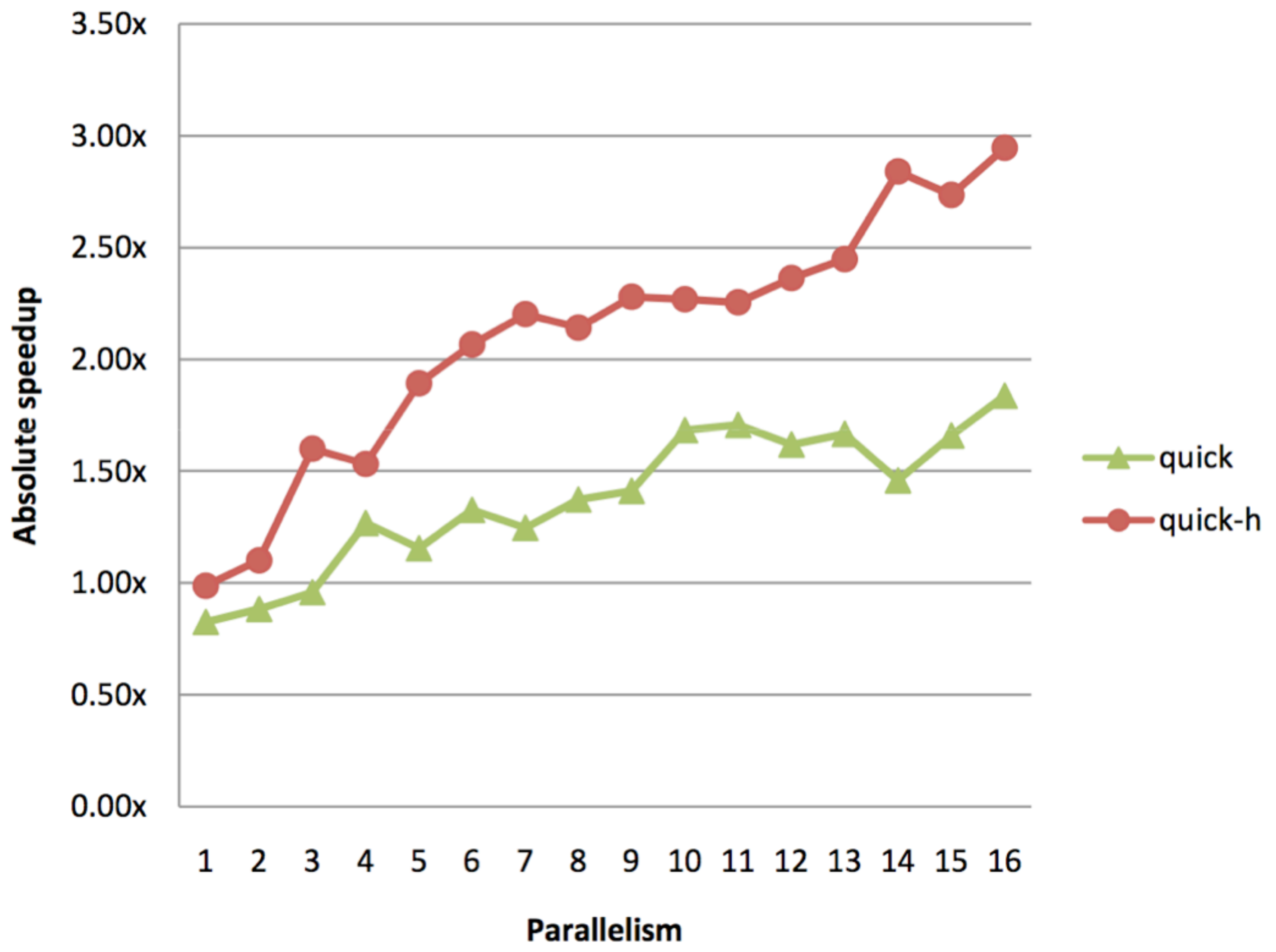
```
(let ((z 0))
 (let ((writez undefined))
  (let ((fp3 (future
               (begin
                (set! writez (lambda ()
                               (set! z 123)))
                (writez)))))
   (let ((readz (lambda () z)))
    (let ((p3 (touch fp3)))
     (let ((p4 (readz)))
      (let ((body (cons p3 p4)))
       body)))))))
```

Speedup vs. parallelism



absolute speedup w.r.t. sequential version

Heuristic



only parallelize when invocations of non-primitive procedures are involved

Automatic parallelization

- our approach works well for divide-and-conquer algorithms
- automatic fork-join parallelism
- brute-force ≠ beneficial
- static analysis good enough, but programs in general must be inherently parallel