Interprocedural Dependence Analysis of Higher-Order Programs via Stack-Reachability

Matthew Might, Tarun Prabhu
University of Utah
www.ucombinator.org

Goal

Determine when parallelization is safe.

Idea

- Dependencies block parallelization.
- Stack structure models dependencies.
- Static analysis can bound the stack.

Example

Is it safe to turn this...

```
(let ((a (f x))
      (b (g y)))
  ...)
```

Example

Is it safe to turn this...

```
(let ((a (f x))
      (b (g y)))
  ...)
```

...into this?

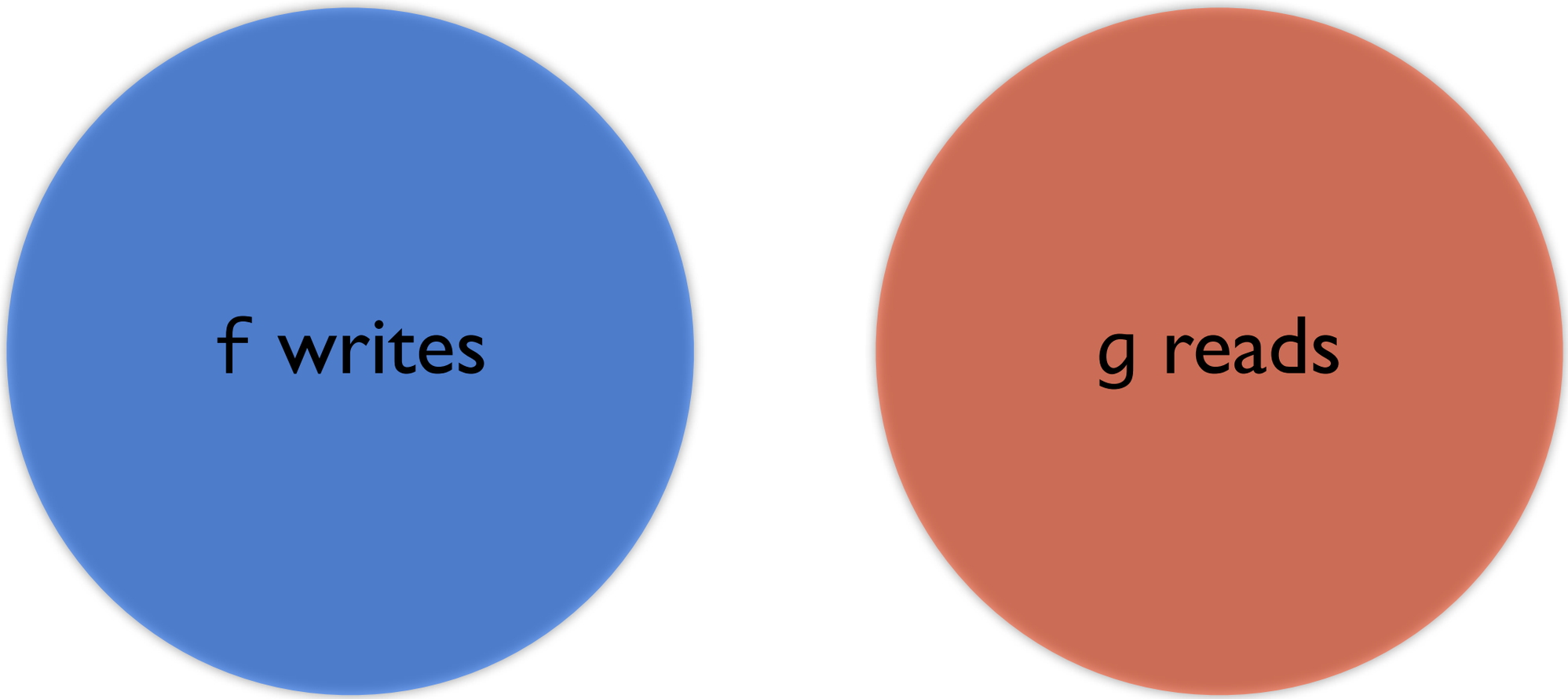
```
(let|| ((a (f x))
        (b (g y)))
  ...)
```

It *depends*...

It *depends*...

...on what depends.

Dependencies

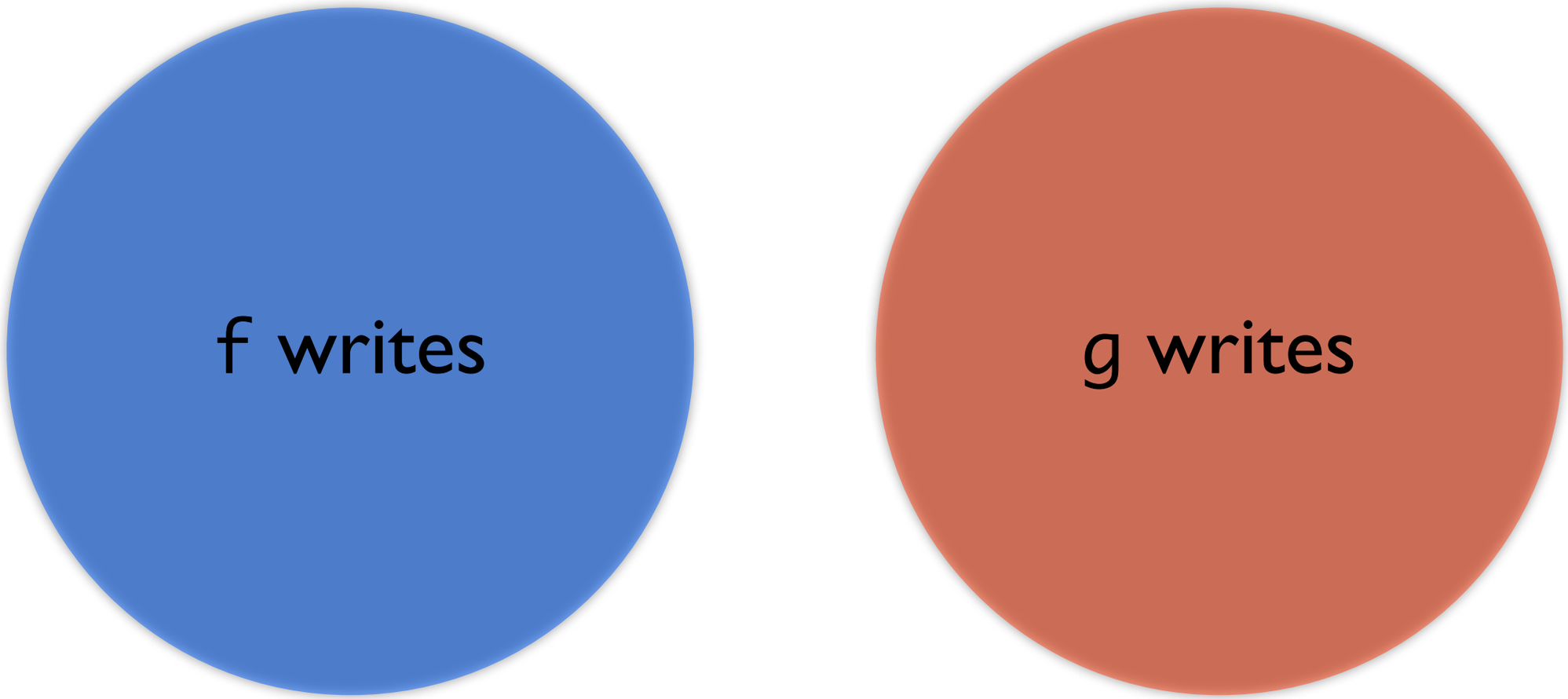


Not unsafe...

Dependencies

Not safe!

Dependencies



Not unsafe...

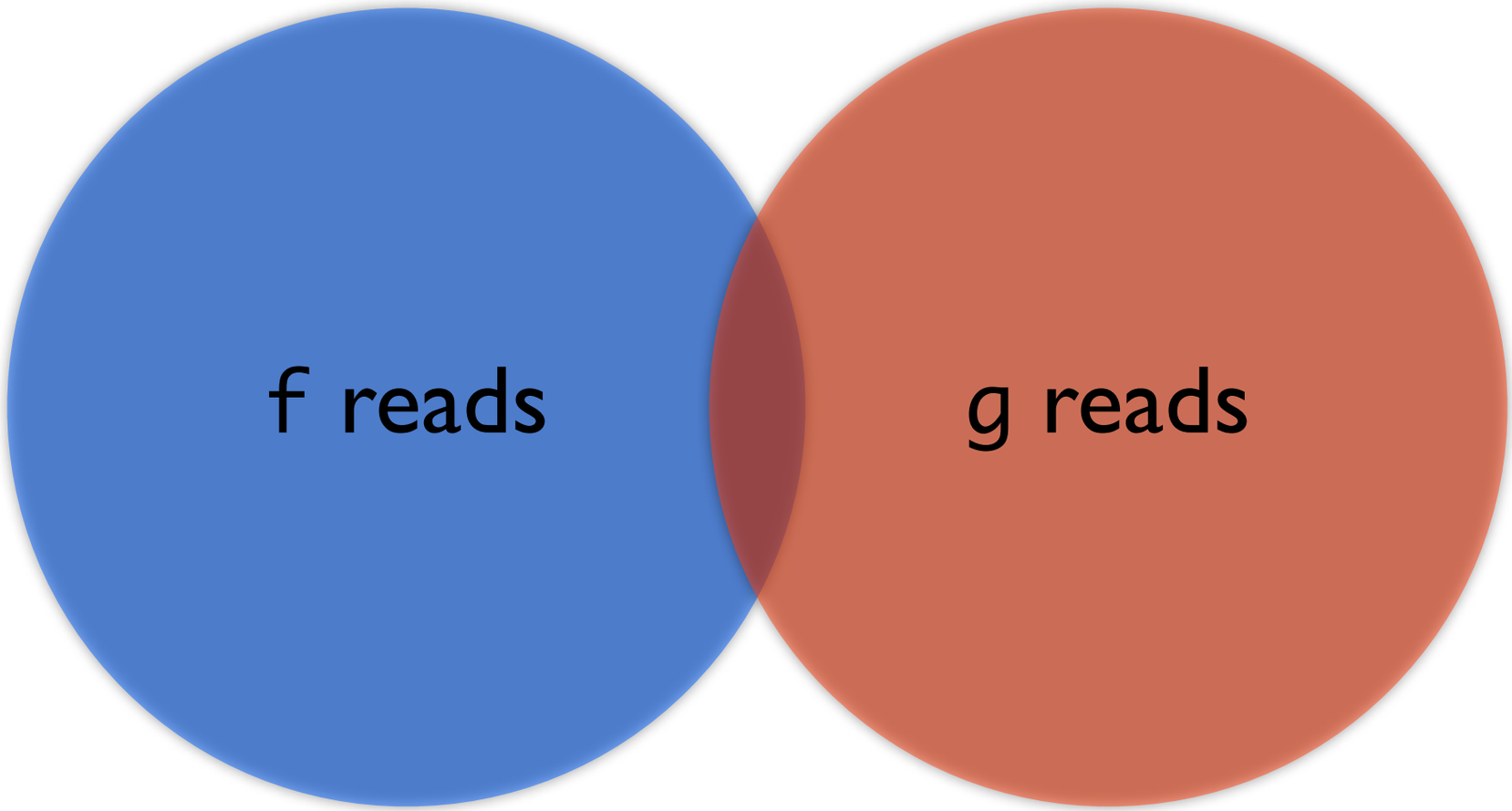
Dependencies

Not safe!

Dependencies

Not unsafe...

Dependencies



Not unsafe...

The Game

The Game

- What resources does a procedure write?

The Game

- What resources does a procedure write?
- What resources does a procedure read?

The Game

- What resources does a procedure write?
- What resources does a procedure read?
- ...when invoked while in context *k*?

Example: Context matters

```
(define (write-a) (set! a 1701))
(define (write-b) (set! b 42)
```

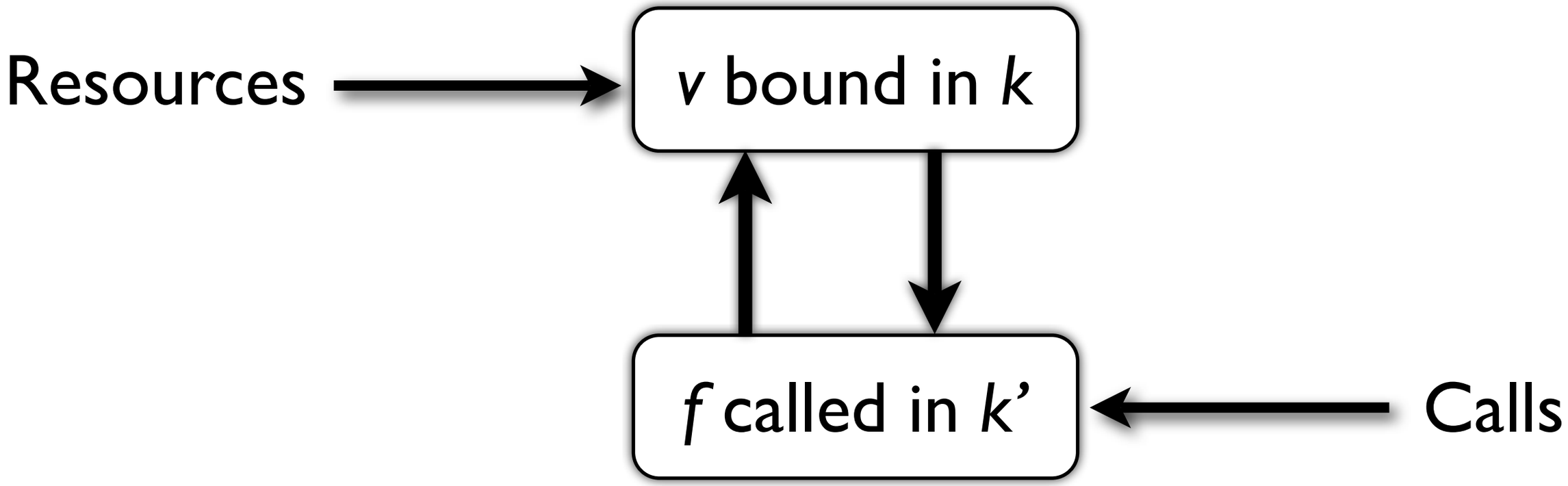
```
(define (call f) (f))

(call write-a) ; call writes a
(call write-b) ; call writes b
```

Example: Context matters

```
(define (loop g t)
  (set! t 10)           ; writes t
  (g)                   ; writes prior t
  (loop (lambda () (set! t 11))
        (+ t 1)))
```

Context-sensitive dependence graphs

Context-sensitive dependence graphs



Observation

- If *f* calls *g*, and
- *g* depends on *x*
- then *f* depends on *x*.

Harrison's principle

- When *x* is read/written,
- if *f* is live on the stack
- then *f* depends on *x*.

What about proper tail calls!?

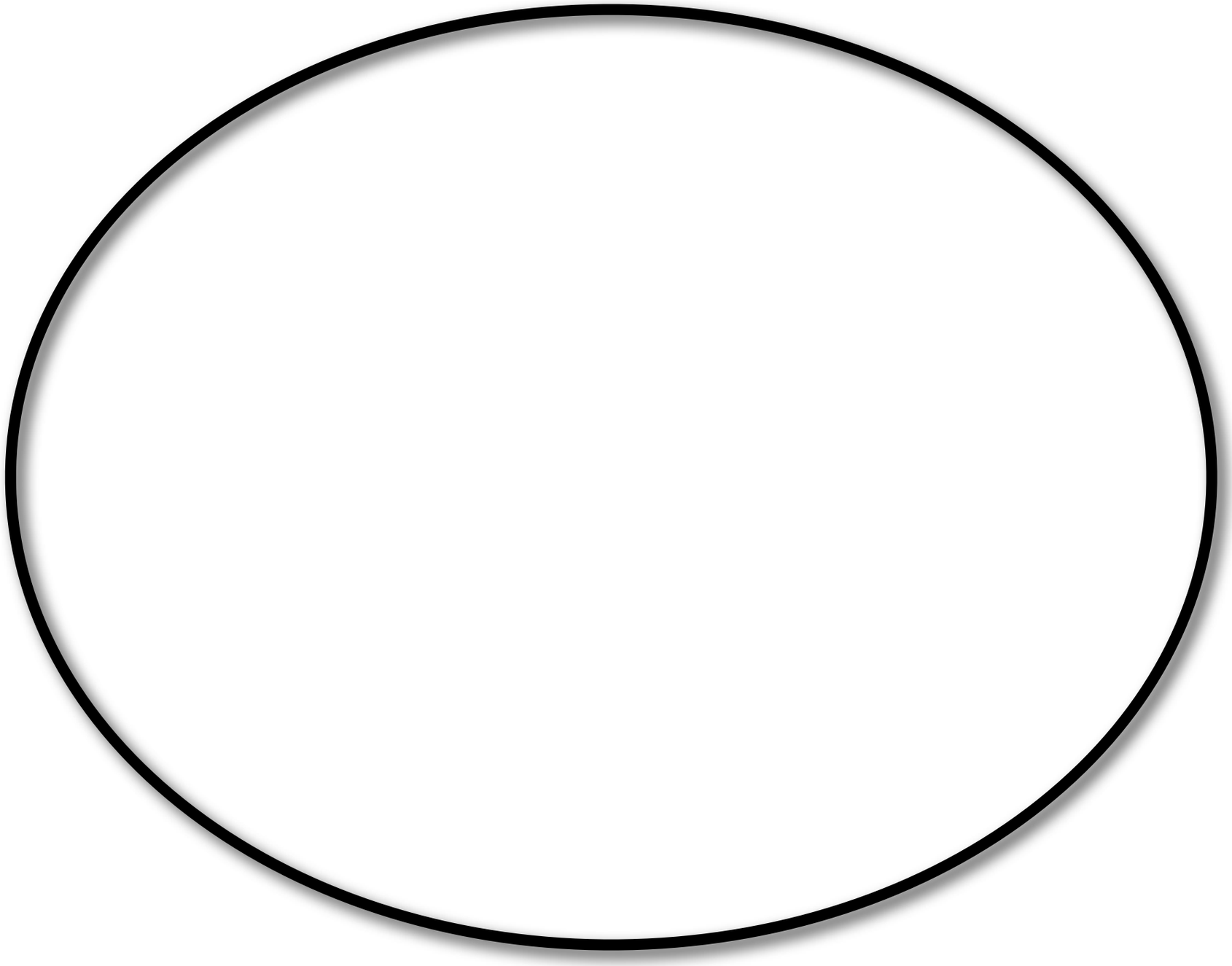
Continuation marks (Clements, Felleisen)

Just mark continuations with calling context.

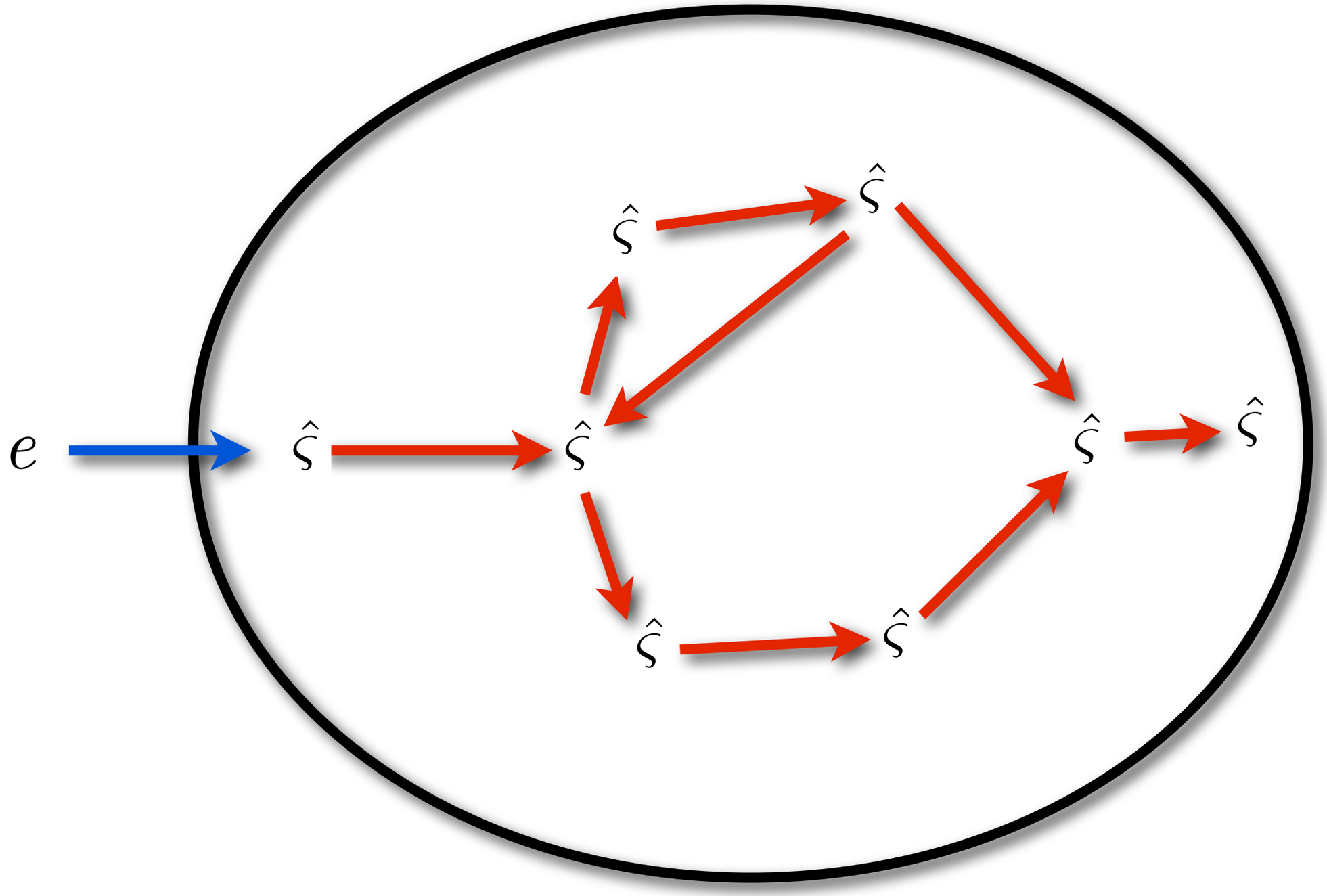
Building the analysis

- Construct CESK machine for ANF, but
- Heap-allocate the continuations, and then
- Abstract directly into *k*-CFA for ANF

Running the analysis



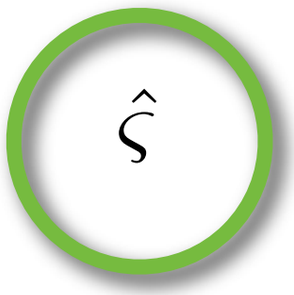
Running the analysis



Running the analysis

Running the analysis

What resources are written?



What resources are read?

Which calling contexts are live?

Make it feasible

Use abstract garbage collection (Might & Shivers, 2006).

What's in the paper?

- Abstract interpretation of CESP for ANF.
- Dependence analysis thereof.
- Abstract garbage collection for ANF.

Limits

- Analysis doesn't work on parallel programs.
- Analysis breaks in the presence of `call/cc`.

Future work

- Rinse, repeat with ΔCFA.
- Rinse, repeat with push-down CFA.
- Analysis for *profitable* parallelism.

Thanks!