CSC 520, Spring 2020

Principles of Programming Languages

Michelle Strout

The University of Arizona

Version: 1

February 26, 2020

Plan

- Yesterday
  - Closures to create "private" variables
  - High-order function curry
  - Reasoning about functions
  - Useful higher-order functions: exists? all? filter map foldr
- Today
  - Continuations

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Continuations

- Code that represents "the rest of the computation."

- Not a normal function call because continuations never return (think "goto with arguments")

- Different coding styles
  - Direct style: Last action of a function is to return a value.
  - Continuation-passing style (CPS): Last action of a function is to "throw" a value to a continuation.

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Uses of continuations

- A style of coding that can mimic exceptions
- Callbacks in GUI frameworks
- Some languages
  - provide a construct for capturing the current continuation and giving it a name k.
  - Control can be resumed at captured continuation by throwing to k.
- Compiler representation
  - Compilers for functional languages often convert direct-style user code to CPS…
  - Because CPS matches control flow of assembly

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Implementation

- **First-class continuations require compiler support**
- **We are going to simulation continuations with function calls in tail position**
- **Tail position is defined inductively:**
  - The body of a function is in tail position.
  - When (if e1 e2 e3) is in tail position, so are e2 and e3.
  - When (let (…) e) is in taile position, so is e, similar for letrec and let*.
  - When (begin e1 … en) is in tail position, so is en.
- **Idea: The last thing that is executed**

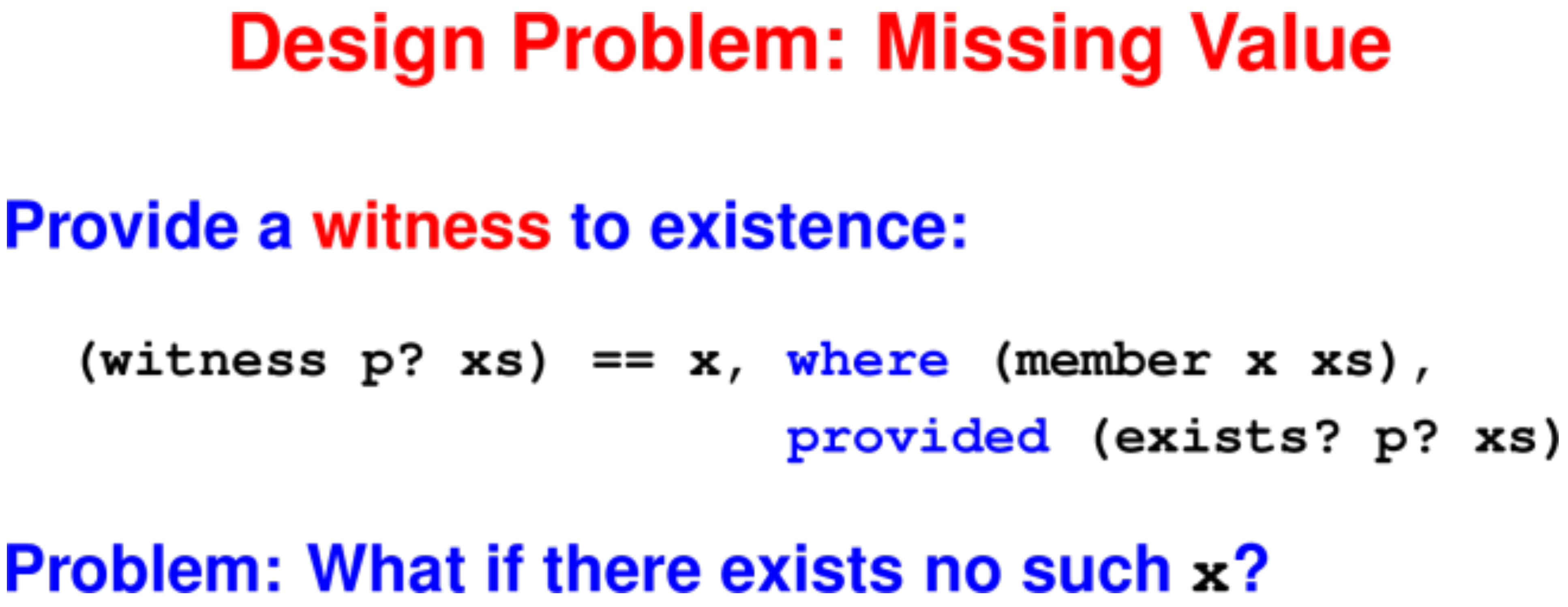
Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

How functions finish

Direct:	`return answer;`
True CPS:	`throw k answer;`
uScheme:	`(k answer)`

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Motivating Ex: From existence to witness



Design Problem: Missing Value

Provide a witness to existence:

```
(witness p? xs) == x, where (member x xs),
                      provided (exists? p? xs)
```

Problem: What if there exists no such x?

Ideas?

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Solution: A New Interface

Success and failure continuations!

Contract written using properties (not algorithmic):

```
(witness-cps p? xs succ fail) = (succ x)
      ; where x is in xs and (p? x)

(witness-cps p? xs succ fail) = (fail)
      ; where (not (exists? p? xs))
```

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Your turn: Refine the laws

```
(witness-cps p? xs succ fail) = (succ x)
    ; where x is in xs and (p? x)

(witness-cps p? xs succ fail) = (fail)
    ; where (not (exists? p? xs))

(witness-cps p? '() succ fail) = ?

(witness-cps p? (cons z zs) succ fail) = ?
    ; when (p? z)

(witness-cps p? (cons z zs) succ fail) = ?
    ; when (not (p? z))
```

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Coding with continuations

```
(define witness-cps (p? xs succ fail)
    (if (null? xs)
        (fail)
        (let ((x (car xs)))
            (if (p? x)
                (succ x)
                (witness-cps p? (cdr xs) succ fail)))))
```

Are all tail positions continuations or recursive calls?

➔ *Do activity*

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Extended example: A SAT Solver

```
;; Find a satisfying assignment if one exists

(val f1 '(and x y z w p q (not x)))

(val f2 '(not (or x y)))

(val f3 '(not (and x y z)))

(val f4 '(and (or x y z) (or (not x) (not y) (not z))))
```

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Continuations for Search

```
    start    +------------+    succeed
---------->|              |------------->
           |    solver    |
<----------|              |<------------
     fail    +------------+     resume
```

start   Gets partial solution, `fail`, `succeed`
(On homework, "solution" is assignment)

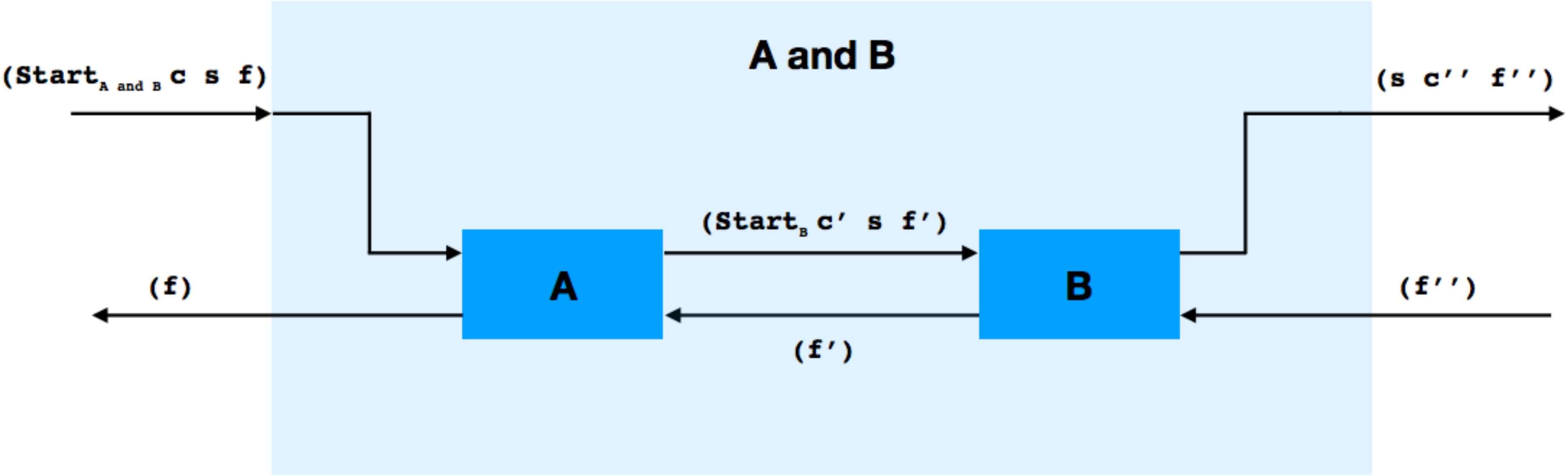
`fail`   Partial solution won't work (no params)

`succeed`   Gets improved solution + `resume`

`resume`   If improved solution won't work,
try another (no params)

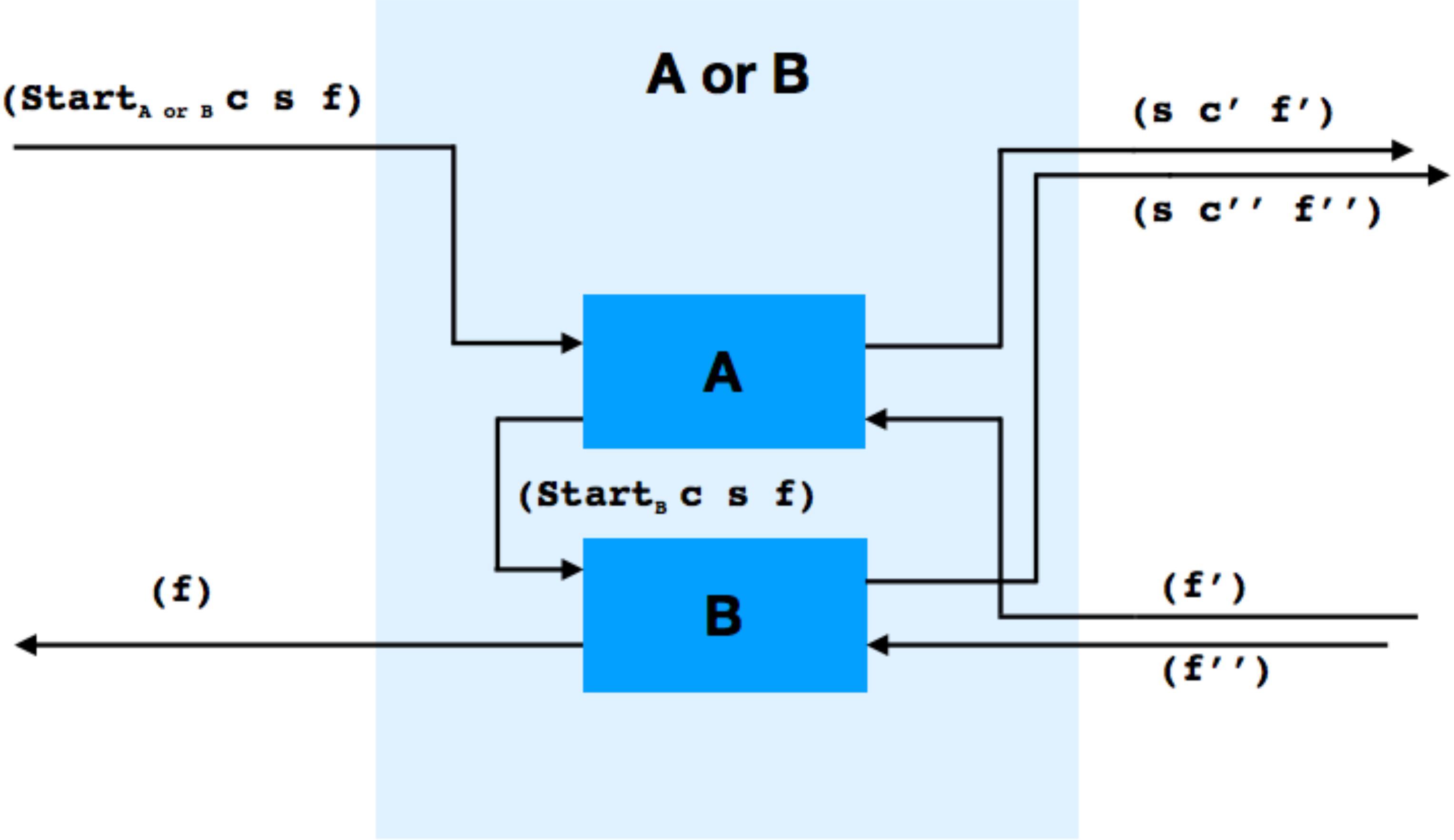
Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Given boxes for "A" and "B", we can build a box for "A and B"



Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Given boxes for “A” and “B”, we can build a box for “A or B



Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher