Announcements

- HW6 Graded
- Late Days column on HuskyCT updated
- HW7 due tonight (Saturday with 2 late days)
  - I’ll post solutions Sunday so please be on time!

Final Exam, Tue. 12/9 8-10am, ITE 125

- Content: Everything!
  - Including this week's lectures, though these lectures were high-level so the questions will be too.
  - Focus will be on material since the midterm, though this built on pre-midterm stuff.
- Format: Similar to midterm (short answer + 4-5 longer questions)
- Reference material (posted today or tomorrow) will be provided
- You can bring **three** 8.5x11" note sheets with any content
- Best way to study: review homeworks and the midterm

Continuations and Wrap-up

CSE 5095-002, Fall 2025

Continuations capture “the rest of the stuff to do”

Kind of like evaluation contexts E[e]

Ex. `(fst (fst (fst (○))))[fst (7, 8)]`

First-class continuations

- Type `α cont` = continuation expecting a `α.`
- `throw : α cont` -> `α -> β`
  - `throw k v` "calls" the continuation k with value v

Example: tail recursion

```
fix mult_all = λl : intlist.
  case unroll l of
     {_. 1;
      (h, t). h * (mult_all t) }
```

Stack overflow!

Example: tail calls/recursion

```
fix mult_all = λl : intlist. λk : int cont.
  case unroll l of
    {_. throw k 1;
    (h, t). mult_all t (λv. throw k (v * h))}
```

All calls are “tail calls”
(nothing to do after return)

Example: short-circuiting

```
fix mult_all = λl : intlist. λk : int cont.
  case unroll l of
     {_. throw k 1;
     (h, t). if h = 0 then throw k 0
             else mult_all t (λv. throw k (v * h))}
```

Example: exceptions

Normal control flow

Exceptional control flow

```
λa : int. λb : int. λk : int cont. λf : string cont.
  if b <> 0 then throw k (a / b)
  else throw f “Divide by zero”
```

But how do we get a continuation?

- `call/cc : (α cont -> β) -> α`
  - “Call with current continuation”
  - `call/cc f` captures the current continuation as an object and passes it to f.
  - `E[call/cc f] -> f` “E”

- e.g., `call/cc (mult_all [1; 2; 3; 4; 5])`

A continuation is a function

- From α to… what?
- Whatever the result of the computation is: need some designated "result type"
- Things work out surprisingly nicely if we choose `void`

So, under Curry-Howard, α cont = $\alpha \rightarrow \perp = \neg\alpha$

The typing rule for call/cc is interesting under Curry-Howard…

$$\frac{\Gamma \vdash e : (\alpha \to void) \to void}{\Gamma \vdash callcc\ e : \alpha}$$

$$\frac{\Gamma \vdash \neg\neg A\ true}{\Gamma \vdash A\ true}$$

STLC with call/cc = Classical Logic!

- Just to be sure: $A \vee \neg A = \alpha + \alpha\ cont$

- `call/cc (λk : (`$\alpha + \alpha$` cont) cont.`
  `throw k (inr (λv : `α`. throw k (inl v))))`

Continuation Passing Style (CPS): All function calls are tail calls

- Difficult to program in, but some efficiency benefits (and helpful for implementing first-class continuations)

- Can be done automatically (and is by some compilers for functional languages) (“CPS transformation”)

With CPS translation, we don't need call/cc but types change a little

call/cc (λk : $(\alpha + \alpha$ cont$)$ cont.
    throw k (inr (λv : α. throw k (inl v))))

$$\alpha + \alpha \; cont = A \vee \neg A$$

Not true constructively

(λk : $(\alpha + \alpha$ cont$)$ cont. k (inr (λv : α. k (inl v))))

$$(\alpha + \alpha \; cont) \; cont \; cont = \neg\neg(A \vee \neg A)$$

True constructively
(proof is above)

Im, H. (2021). On Correspondence between Selective CPS Transformation and Selective Double Negation Translation. *Mathematics*, *9*(4), 385. https://doi.org/10.3390/math9040385

So does that mean that if A is true classically, ¬¬A is true constructively?

- Yes [Glivenko, 1929]

Another application of continuations: stateless web services

Black Friday: 

“Add to cart”

1 item in cart

amazon

Wednesday: 

“Checkout”

“Who are you?”

amazon

Another application of continuations: stateless web services

Black Friday:

“Add to cart”

(1 item in cart, λaction. …)

Wednesday:

(“Checkout”, λaction. …)

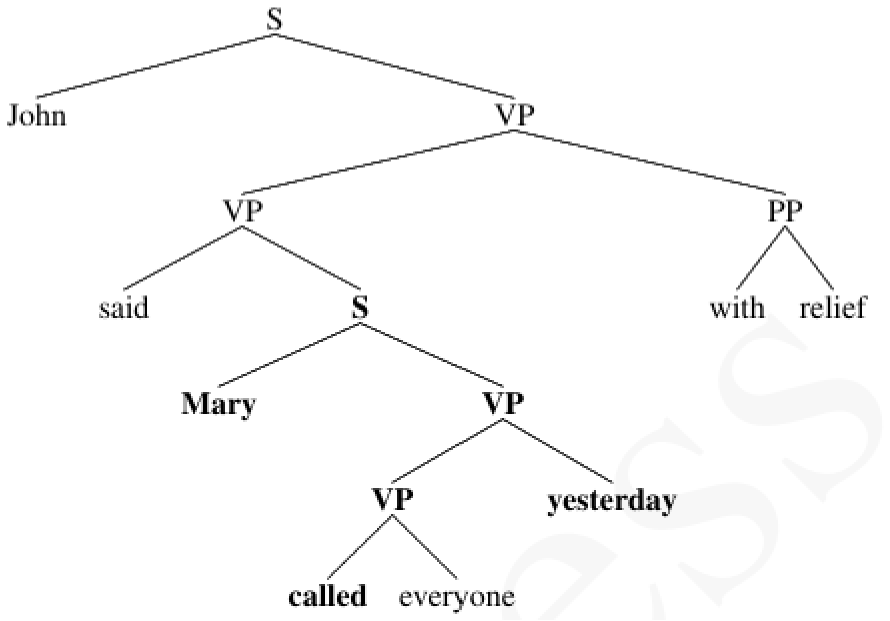
(λaction. …) “Checkout”

Queinnec, C. Continuations and Web Servers. *Higher-Order Symb Comput* **17**, 277–295 (2004). https://doi-org.ezproxy.lib.uconn.edu/10.1007/s10990-004-4866-z

Another application of continuations: natural language

- "John said Mary called everyone yesterday with relief"
- Q: What happened to everyone?

(λx. Mary called x yesterday)



Barker, Chris & Shan, Chung-chieh. (2014). Continuations and Natural Language. 10.1093/acprof:oso/9780199575015.001.0001.

Another application of continuations: natural language

An occasional sailor passed by (misplaced modifier)

k occasional

Occasionally, *k* ($\lambda x.\ x$)

Occasionally, a sailor passed by

Barker, Chris. Continuations in Natural Language. In Hayo Thielecke (ed). Proceedings of the Fourth ACM SIGPLAN Continuations Workshop (CW’04). Technical Report CSR-04-1, School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom. 1–11.

($\lambda x.$?) this class

- … unfortunately not a lot of classes at Uconn
- Research/independent study!

- Advanced Topics in Types and Programming Languages (ed. Pierce)

- Software Foundations (Pierce, https://softwarefoundations.cis.upenn.edu)