CS 3110

The Substitution Model

Nate Foster

Spring 2019

Today's music: *Substitute* by The Who

Review

Previously in 3110: simple interpreter
- abstract syntax tree (AST)
- evaluation based on single steps

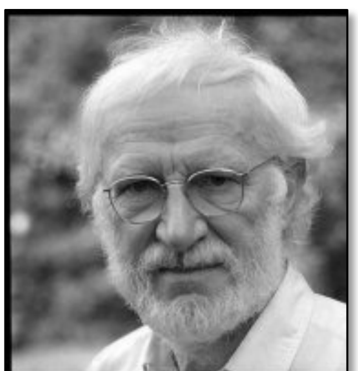
Today:
- Formal syntax: BNF
- Formal dynamic semantics: small-step, substitution model
- Formal static semantics

FORMAL SYNTAX

```
e ::= x
    | i
    | e1 + e2
    | let x = e1 in e2
```

Backus-Naur Form (BNF)

John Backus (1924-2007)
ACM Turing Award Winner 1977
“For profound, influential, and lasting contributions to the design of practical high-level programming systems”



Peter Naur (1928-2016)
ACM Turing Award Winner 2005
“For fundamental contributions to programming language design”

BNF

Note resemblance:

```
e ::= x
    | i
    | e1 + e2
    | let x = e1 in e2

type expr =
  | Var of string
  | Int of int
  | Add of expr * expr
  | Let of string * expr * expr
```

FORMAL DYNAMIC SEMANTICS

$$e \longrightarrow e'$$

single-step relation

$$\mathbf{v} \not\rightarrow$$

values never step

$$e \longrightarrow^{*} e'$$

multi-step relation

```
e1 + e2 --> e1' + e2
   if e1 --> e1'

v1 + e2 --> v1 + e2'
   if e2 --> e2'

v1 + v2 --> i
   if i is the result of primitive operation v1+v2
```

Demo

```
let x = e1 in e2
--> let x = e1' in e2
```

if `e1 --> e1'`

```
let x = v1 in e2 --> e2{v1/x}
```

Demo

Booleans

```
e ::= x | i | b
    | e1 + e2 | e1 && e2
    | let x = e1 in e2
    | if e1 then e2 else e3

v ::= i | b
```

Demo

Evaluation models

Small-step substitution model:

- Substitute value for variable
- Good mental model for evaluation
- Inefficient: too much work at run time
- Not really what OCaml does

Big-step environment model:

- Maintain data structure binding variables to values
- At the heart of what OCaml really does
- **(next lecture)**

Upcoming events

- [today] Foster OH @ 1:15pm
- [Friday @ 11:59pm] Team Evaluation Due

This is not a substitute.

THIS IS 3110