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
\[ e ::= x \]

<table>
<thead>
<tr>
<th>( i )</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>( e_1 + e_2 )</th>
</tr>
</thead>
</table>

| \( \text{let } x = e_1 \text{ in } e_2 \) |

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:

\[
e ::= x \\
  \mid i \\
  \mid e_1 + e_2 \\
  \mid \text{let } x = e_1 \text{ in } e_2
\]

**Type** `expr` =

\[
\mid \text{Var of string} \\
\mid \text{Int of int} \\
\mid \text{Add of expr} \times \text{expr} \\
\mid \text{Let of string} \times \text{expr} \times \text{expr}
\]
single-step relation
values never step
$e \rightarrow^* e'$

multi-step relation
\[e_1 + e_2 \rightarrow e_1' + e_2\]
\[\text{if } e_1 \rightarrow e_1'\]

\[v_1 + e_2 \rightarrow v_1 + e_2'\]
\[\text{if } e_2 \rightarrow e_2'\]

\[v_1 + v_2 \rightarrow i\]
\[\text{if } i \text{ is the result of primitive operation } v_1 + v_2\]
let \( x = e_1 \) in \( e_2 \)

\[\rightarrow\] let \( x = e_1' \) in \( e_2 \)

\[\text{if } e_1 \rightarrow e_1'\]

let \( x = v_1 \) in \( e_2 \) \(\rightarrow\) \( e_2\{v_1/x\} \)
Booleans

e ::= \ x \mid \ i \mid \ b \\
    \mid e_1 + e_2 \mid e_1 \&\& e_2 \\
    \mid \text{let } x = e_1 \text{ in } e_2 \\
    \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\

v ::= \ i \mid \ b
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