CS 536 Announcements for Thursday, February 17, 2022

Programming Assignment 2
- due Friday, February 25

Last Time
- Makefiles
- ambiguous grammars
- grammars for expressions
  - precedence
  - associativity

Today
- wrap up CFGs
- list grammars
- syntax-directed translation

Next Time
- continue syntax-directed translation
- ASTs

Recall our expression grammar

Write an unambiguous grammar for integer expressions involving only addition, multiplication, and parentheses that correctly handles precedence and associativity.

\[
\begin{align*}
\text{expr} & \rightarrow \text{expr} \quad \text{PLUS} \quad \text{term} \\
& \quad | \quad \text{term} \\
\text{term} & \rightarrow \text{term} \quad \text{TIMES} \quad \text{factor} \\
& \quad | \quad \text{factor} \\
\text{factor} & \rightarrow \text{INTLIT} \\
& \quad | \quad \text{LPAREN} \quad \text{expr} \quad \text{RPAREN}
\end{align*}
\]

Extend this grammar to add exponentiation (POW)

Add exponentiation (POW) to this grammar, with the correct precedence and associativity.
List grammars

Example a list with no separators, e.g., A B C D E F G

Another ambiguous example

stmt → IF cond THEN stmt
  | IF cond THEN stmt ELSE stmt
  | . . .

Given this word in this grammar: if a then if b then s1 else s2
How would you derive it?
Overview of CFGs

CFGs for language definition
- the CFGs we've discussed can generate/define languages of valid strings

CFGs for language recognition

CFGs for parsing

Syntax-directed translation
= translating from a sequence of tokens into a sequence of actions/other form, based on underlying syntax

To define a syntax-directed translation
- augment CFG with translation rules
  - define translation of LHS non-terminal as a function of

To translate a sequence of tokens using SDT
**Example: grammar for language of binary numbers**

<table>
<thead>
<tr>
<th>CFG</th>
<th>translation rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>b → 0</td>
<td>b.trans = 0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>b 0</td>
</tr>
<tr>
<td></td>
<td>b 1</td>
</tr>
</tbody>
</table>
Example: grammar for language of variable declarations

CFG
declList \rightarrow \epsilon \\
| \quad decl \ declList \\
dcl \rightarrow \text{type ID ;} \\
type \rightarrow \text{INT} \\
| \quad \text{BOOL}
Example: grammar for language of variable declarations

CFG
declList → ε
| decl declList
decl → type ID ;
type → INT
| BOOL
Abstract syntax trees

\( \text{AST} = \text{condensed form of parse tree} \)

- 
- 
- 
- 
- 

Example