Plan for Today

PA3 Status
- Let's look at PA2 pics!!
- Should have PA2 grammar AST specified
- Should have code generation for example PA2 ASTs working
- Have you found the regular expressions for Java comments yet?

Ambiguous Grammars and Happy
- Inline expressions
- Unary minus
- Dangling else
Ambiguous Grammars

Ambiguous grammar:

>2+ parse trees for 1 sentence

Expression grammar           parse tree 1

\[ E \rightarrow E \ast E \]
\[ E \rightarrow E + E \]
\[ E \rightarrow E - E \]
\[ E \rightarrow ( E ) \]
\[ E \rightarrow ID \]
\[ E \rightarrow NUM \]

String                                       parse tree 2

42 + 7 * 6

what about 42-7-6?
Goal: disambiguate the grammar

Cause
– the grammar did not specify the precedence nor the associativity of the operators +, -, *

Two Options
– keep the ambiguous grammar, but add extra directives to the parser (many LALR parser generators can do this)
– Rewrite the grammar, making the precedence and associativity explicit in the grammar.
Unambiguous grammar for simple expressions

Grammar

\[
E \rightarrow E + T \mid E - T \mid T \\
T \rightarrow T * F \mid F \\
F \rightarrow (E) \mid \text{ID} \mid \text{NUM}
\]

String

42 + 7 * 6

How is the precedence encoded?

How is the associativity encoded?
Let’s add ! and && to the grammar

Grammar

\[
E \rightarrow E + T \mid E - T \mid T \\
T \rightarrow T * F \mid F \\
F \rightarrow (E) \mid \text{ID} \mid \text{NUM}
\]
Instead Let’s Specify Precedence and Associativity to Happy

List operators in order of lower precedence to higher precedence
- Yes, this is opposite of how typically shown in precedence tables.
- However, it is the same order as typically shown when precedence levels are added to a grammar by hand.

Syntax
%left ‘+’ ‘-’
%left ‘*’

How this works
- Operator on stack and look-ahead operator are compared.
  - If on stack is higher, then reduce.
  - If look-ahead higher, then shift.
  - If same, then left-assoc reduce, right-assoc shift, nonassoc fail.
**Unary Minus**

**Higher precedence than subtraction**

- How should we parenthesize 10 - - - 4 ?
- How about 9 - - 3 - - 4 ?

**Syntax**

```%
left `+` `-`
%
left `*`
%
left NEG
```

Exp : ... 
   | `-' Exp %prec NEG { - $2 }
   ...
```
Dangling else problem: ambiguity

Given

\[
S \rightarrow \text{IF } t \text{ THEN } S \text{ X } | o \\
X \rightarrow \text{ELSE } S | \varepsilon
\]

construct two parse trees for

\[
\text{IF } t \text{ THEN } \text{IF } t \text{ THEN } o \text{ ELSE } o
\]

Which is the correct parse tree? (C, Java rules)
Dangling else disambiguation

The correct parse tree is:

LALR parser generators will create this grammar by default by preferring to shift over doing a reduce.