Flexible & Ambiguous Grammars

- Let's work on one more example.

**Example**: Build a parse tree for $3 - 1 + 2$ based on the following rules

$$Expr \rightarrow Expr\ Op\ Expr|(Expr)|Integer$$

$$Op \rightarrow +|−|∗|/$$

$$Integer \rightarrow Digit\{Digit\}$$

$$Digit \rightarrow 0|⋯|9$$

We can generate two distinct parse tree using above rules.

We call a grammar *ambiguous grammar* if its language contains at least one string that has two or more distinct parse trees.

Ambiguous grammars are not good as they may confuse the compilers. Compilers can generate two different results for the same expression.

However, sometimes, we may want to use an ambiguous grammar to simplify the number of rules required. In this case, ambiguities in grammars are generally resolved using additional rules.

- For example, if we have a table of precedence and a default left-to-right ordering of operators of equal precedence, then we can resolve any ambiguities that arise.

Dangling else is a common ambiguity in language syntax.

- When an if statement is contained inside an if statement, which if statement does a subsequent else belong to?

- Consider the following code snippet
Without inserting curly braces, it seems like that the else branch could match either if condition. Different languages address this ambiguity differently.

- Solution of C
  - C addresses by including a description in the nature language in its documentation that an else clause is associated with the textually nearest if statement in any ambiguous case.
  - So, the output of the follow code is “there.”

```c
#include <stdio.h>

int main (int args, char *argv[]) {
    int a = -1;
    int b = 1;

    if (a < 0)
        if (b < 0)
            printf("here\n");
        else
            printf("there\n");
    printf("there\n");
}
```

- Solution of Java
  - Clearly defined in grammar to address the ambiguity.
  - It is not permitted that an if statement without an else clause as the single statement after an if.
  - The following code snippet, for example, will not do what the tabbing implies. The actual output is “there”, since Java consider the else branch belongs to the second if statement.

```java
public class Ambiguity {
    public static void main (String args[]) {
        int a = -1;
        int b = 1;

        if (a < 0)
            if (b < 0)
                System.out.println("here");
            else
                System.out.println("there");
    }
}
```

- Solution of Python
  - Require nested if statements to be indented. The actual output is “"
a = -1
b = 1

if (a < 0):
    if (b < 0):
        print "here\n"
    else:
        print "there\n"