Syntax and Semantics

Aditya Thakur

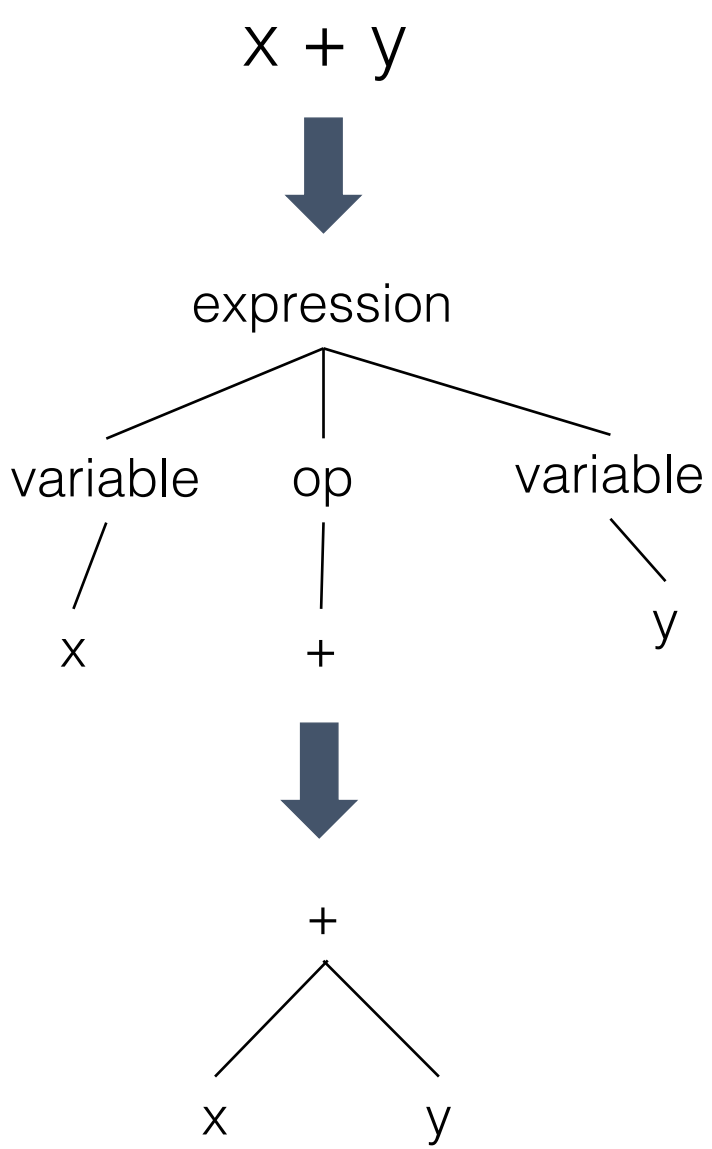
What Characterizes a Language

- Syntax: form and structure
  - Rules for combining characters into symbols (lexical)
    - Keywords: if, while, for …
    - Identifiers: variable names (x, i), function names (foo) …
    - Literals: 100, 1.0, “foo” …
    - Operators: +, -, *, / …
    - Others: (,), {, }, >, =, …
  - Rules for combining the symbols (syntactic)
    - To determine the legal sentences (programs) in a language
    - Example: if <conditional> then <statement> else <statement>
- Semantics: meaning of a sentence

The Structure of a Compiler

1. Lexical Analysis
2. Syntactic Analysis (Parsing)
3. Semantic Analysis
4. Optimization
5. Code Generation

At least the first 3 could be understood by analogy to how humans comprehend English


x + y
string
expression
variable
op
variable
x
+
y
Parse tree
+
x
y
Abstract syntax tree

Lexical Analysis

- First step: recognize words
  - Smallest unit above letters

This is a sentence.

- Note the
  - Capital “T” (start of sentence symbol)
  - Blank “ ” (word separator)
  - Period “.” (end of sentence symbol)

More Lexical Analysis

- Lexical analyzer divides program text into "words" or "tokens"

  If x == y then z = 1; else z = 21;

- Tokens:

  if, x, ==, y, then, z, =, 1, ;, else, z, =, 21, ;

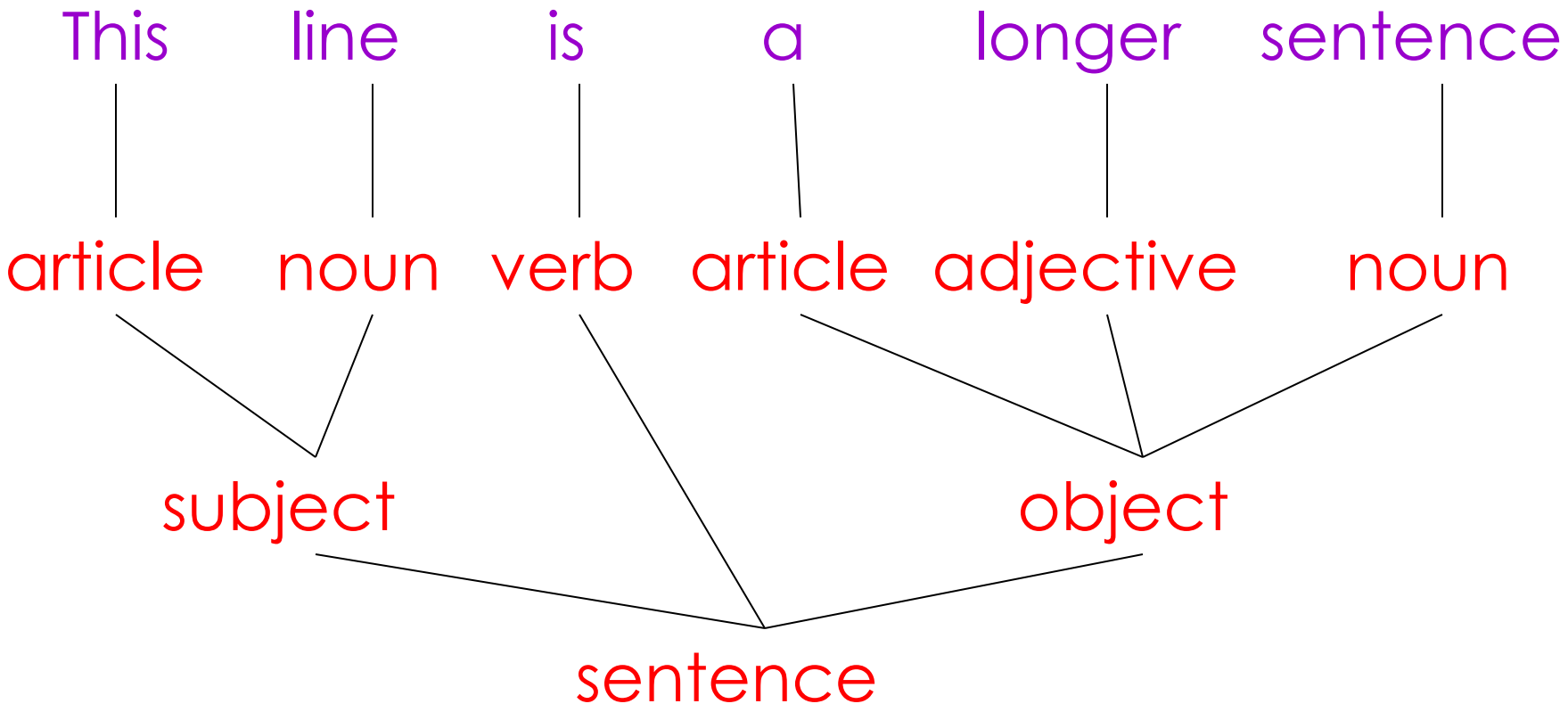
- How to specify token classes?
  - Use regular expressions and finite state automata (ECS 142/120)
  - Tools such as lex, and flex (learn to use them in ECS 142)

Syntactic Analysis (Parsing)

- Once words are understood, the next step is to understand sentence structure

- Parsing corresponds to Diagramming Sentences
  - The diagram is a tree

Diagramming a Sentence

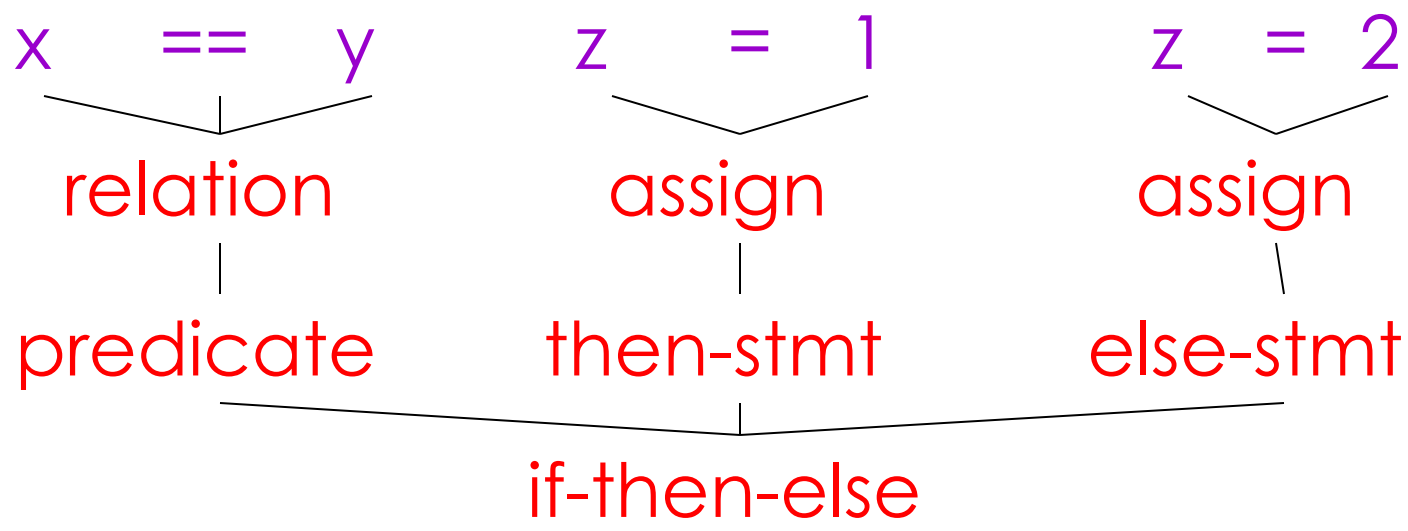


Parsing Programs

- Parsing program expressions is the same
- Consider

If x == y then z = 1; else z = 2;

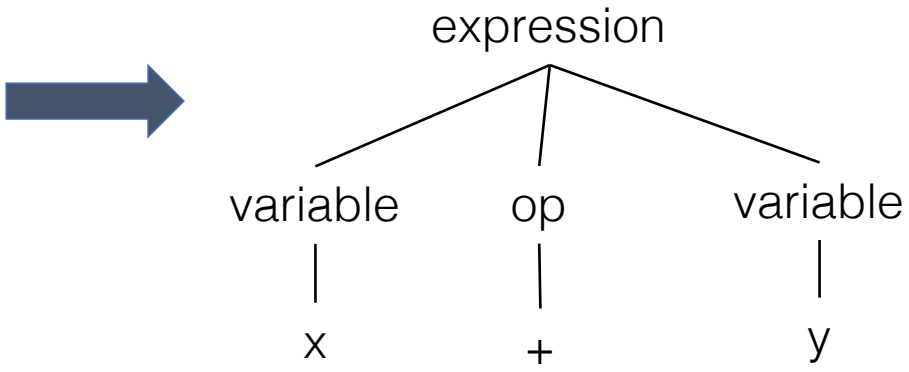
- Diagrammed



Semantic Analysis

- Once sentence structure is understood, we can try to understand *meaning*

x + y



What is the meaning of '+'?

Suppose 'x' and 'y' are of type `string`?

Semantic Analysis

- Compilers perform limited analysis to catch inconsistencies
  - Type checking
  - Variable checking
  - Label checking

Semantic Analysis in English

- Example:

Jack said Jerry left his assignment at home.

What does “his” refer to? Jack or Jerry?

Jack said Jack left his assignment at home.

How many Jacks are there?

Which one left the assignment?

Semantic Analysis in Programming

- Programming languages define strict rules to avoid such ambiguities
- This C++ code prints “4”; the inner definition is used

```
{
  int Jack = 3;
  {
      int Jack = 4;
      std::cout << Jack;
  }
}
```

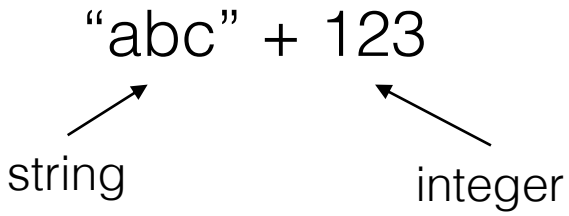
More Semantic Analysis

- Compilers perform many semantic checks besides variable bindings
- Example:

  Jack left its homework at home.

- A “type mismatch” between its and Jack

More Semantic Analysis



Specifying Syntax and Semantics

- These need to be precise to avoid ambiguities
- Both can be described in formal ways
- In this class
  - We focus on formally specifying the syntax
    - Although not much into the language theory
    - More in-depth coverage in ECS 142 and ECS 120
  - We will use informal ways to specify semantics
    - Generally just in terms of English descriptions
    - More formal approaches covered in ECS 142 and ECS 240
- We will look at a few formal notations for specifying syntax