Topic 4:

Syntax

Q: How do you comfort a grammar expert?

A: “There, their, they’re”

Syntax v. Semantics

Definition: Syntax

Semantics are related but distinct:

Tokens and Lexemes

Definition: Token

Definition: Lexeme

Sometimes a token and a lexeme are the same:

Token:	Lexeme:

But usually not:

Token:	Lexemes:

Lexical Structure, Scanning, and Parsing

Definition: Lexical Structure

During language translation:

Identifying Tokens (1 / 2)

Where does one lexeme stop and the next start?

Definition: Principle of Longest Substring

Example from Java:

```
int i=0;  doi++;  while (i<10);
```


Identifying Tokens (2 / 2)

Example from FORTRAN 66:

```
DO 99 I=1,10   v.   DO 99 I=1.10
```

Context–Free Grammars & Backus–Naur Form (1 / 2)

In the mid–1950s, linguist Noam Chomsky defined a relationship, known as the Chomsky Hierarchy, between four classes of languages:

Type 0:

Type 1:

Type 2:

Type 3:

Context–Free Grammars & Backus–Naur Form (2 / 2)

Definition: Grammar

John Backus and Peter Naur defined and refined a notation for describing programming language syntax while developing the language ALGOL.

The connection:

Regular and Context–Free Languages (1 / 2)

Regular languages can be expressed by regular expressions.

Be Aware! — Regular expressions, as provided in programming languages, add additional features that go beyond true regular languages.

Regular and Context–Free Languages (2 / 2)

Context–free languages are defined by context–free grammars (CFGs).

BNF Productions (1 / 3)

Note: This style of writing BNF productions (rules) differs from our textbook's style. Why not be consistent? Italics are hard to write by hand!

Productions have this form:

Example(s): BNF for a C language assignment statement

BNF Productions (2 / 3)

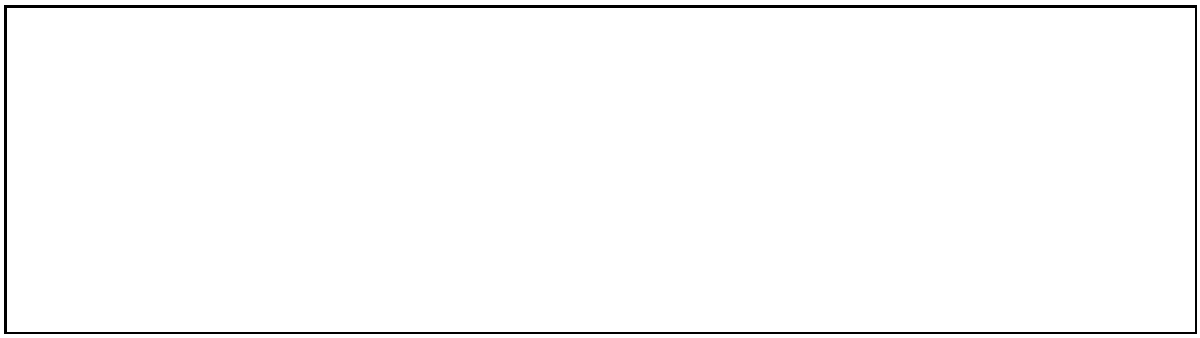
Non–terminals may have multiple definitions.

Example(s): Java IF statements

BNF Productions (3 / 3)

BNF rules look simple, but hide a lot of power!

Example(s): BNF can describe comma–separated lists



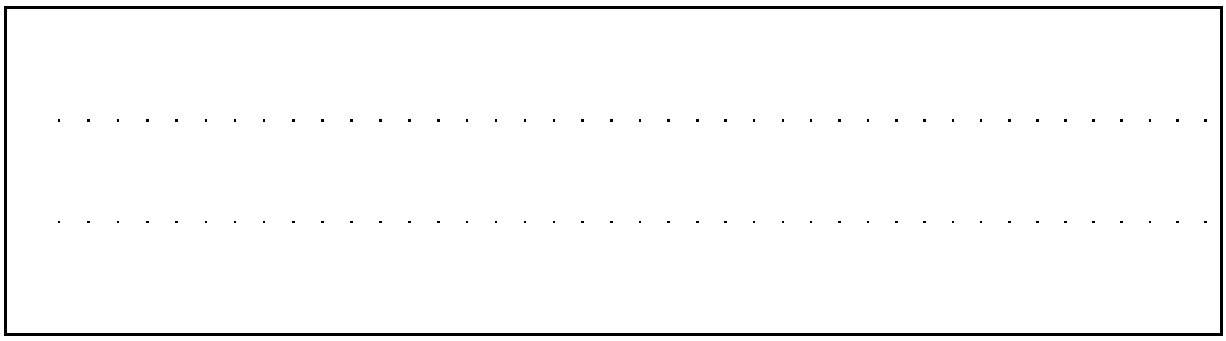
A Larger BNF Example

Consider this grammar, in which `<block>` is the start symbol.

An expression of the language defined by this grammar:

Constructing a Derivation (1 / 2)

Definition: Derivation



One possible derivation algorithm ('RHS' ≡ Right–Hand Side):

- Write down the start symbol's rule
- Repeat until all non–terminals have been removed:
  - Find the left–most non–terminal on the RHS of the rule
  - Replace it with the RHS of one of its rules

Constructing a Derivation (2 / 2)

Note: The symbol ⇒ means "derives."

Let's derive our expression: `start I = J * K stop`

```
<block> ⇒ start <stmt-list> stop
        ⇒
        ⇒
        ⇒
        ⇒
        ⇒
        ⇒
```

Parse Trees (a.k.a. Concrete Syntax Trees)

Here's a hierarchical view of the derivation process:

(Recall: `start I = J * K stop`)

Abstract Syntax Trees

Most of the non–terminals in Concrete Parse Trees aren't very helpful, so why keep drawing them?

Ambiguous Grammars (1 / 6)

Consider this enhanced version of our assignment statement grammar:

```
<stmt> → <var> = <expr>
 <var> → I | J | K
```

We'll use it to derive this expression:

Ambiguous Grammars (2 / 6)

Problem! That grammar can derive two syntax trees:

```
K = I * J / K
```

Ambiguous Grammars (3 / 6)

How can we remove this ambiguity?

Ambiguous Grammars (4 / 6)

Let's look at a rewriting option for this grammar:

```
<stmt> → <var> = <expr>
 <var> → I | J | K
```

Ambiguous Grammars (5 / 6)

```
I / J * K
```

v.

```
I / (J * K)
```


Ambiguous Grammars (6 / 6)

Notes:

Extended BNF (EBNF) (1 / 4)

Ref: Wirth, Niklaus. "What Can We Do about the Unnecessary Diversity of Notation for Syntactic Definitions?" CACM, 11/1977.

We've seen BNF notations like this:

<expr> → <expr> * <term> | <term>

Question:

Extended BNF (EBNF) (2 / 4)

Added notations (but not power!) in EBNF include:

{ } —

[ ] —

( ) —

Extended BNF (EBNF) (3 / 4)

Example(s):

Wirth's language Pascal optionally lists files on the first line:

```
program foo (input, output) ;
```

We can express this in BNF:

Or more compactly in EBNF:

Extended BNF (EBNF) (4 / 4)

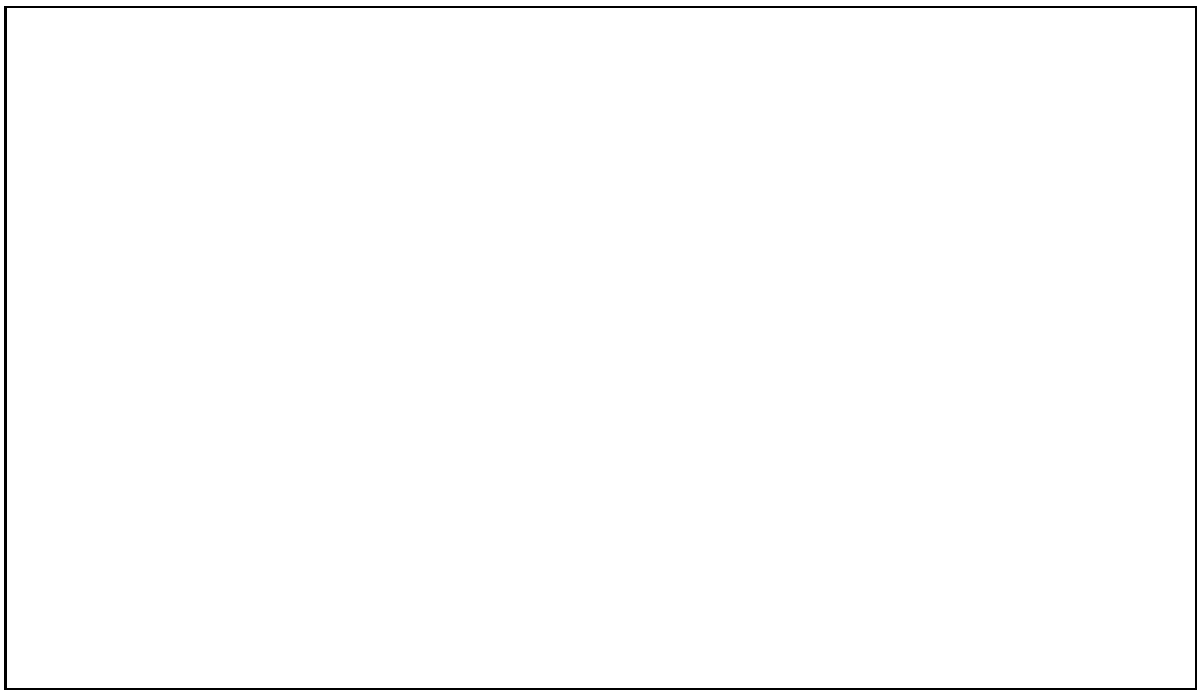
Problem! EBNF parse trees are sometimes ambiguous.

Example(s):

Syntax Diagrams (a.k.a. Railroad Diagrams)

Looking for a visualization of (E)BNF?

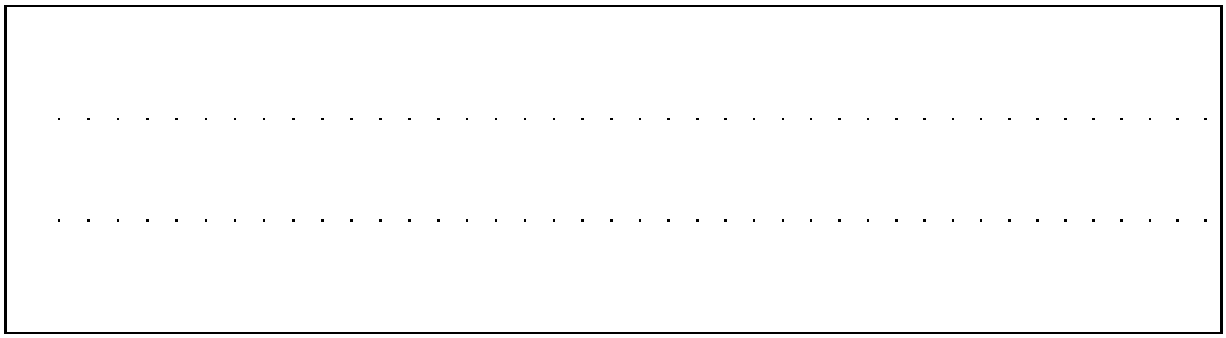
Example(s):



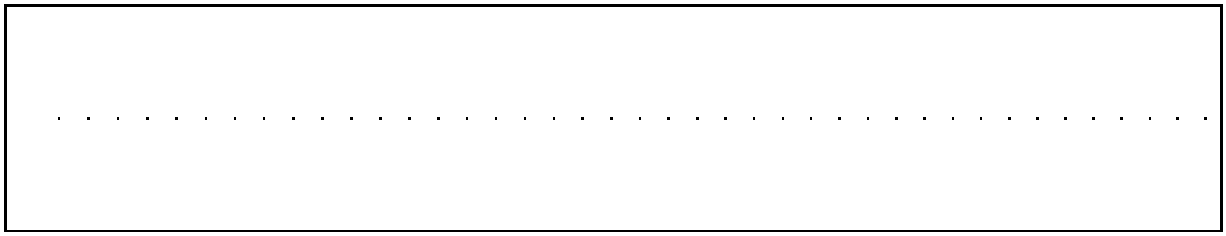
Parsing (1 / 3)

Which programs test code against a language's expectations?

Definition: Parser



Definition: Recognizer



Parsing (2 / 3)

There are two major types of parsers.

1.

Parsing (3 / 3)

2.

Concluding Notes on Syntax

Which program checks that the tokens are correctly constructed?

CFGs are capable of more than mere syntax–checking