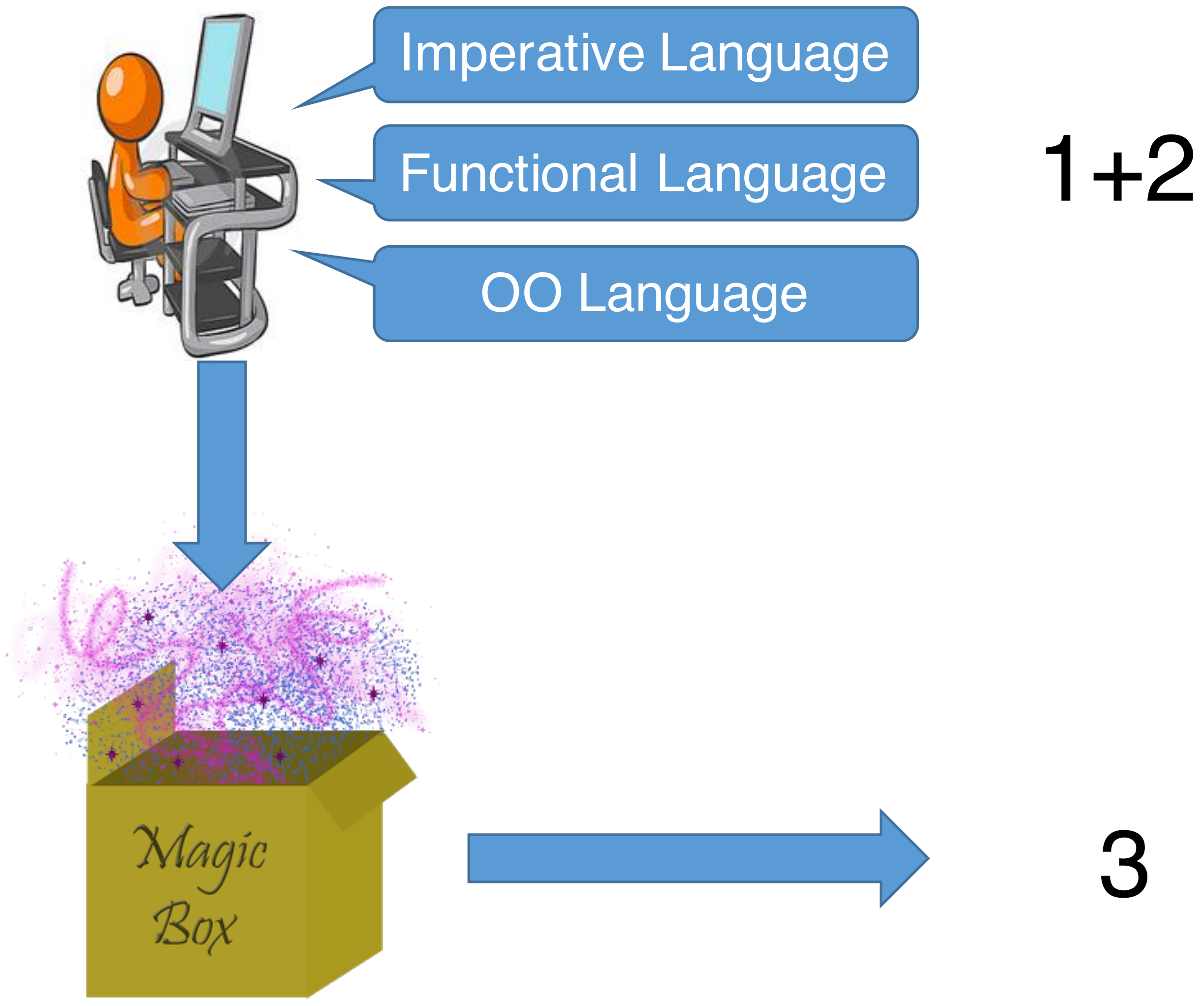
Syntax

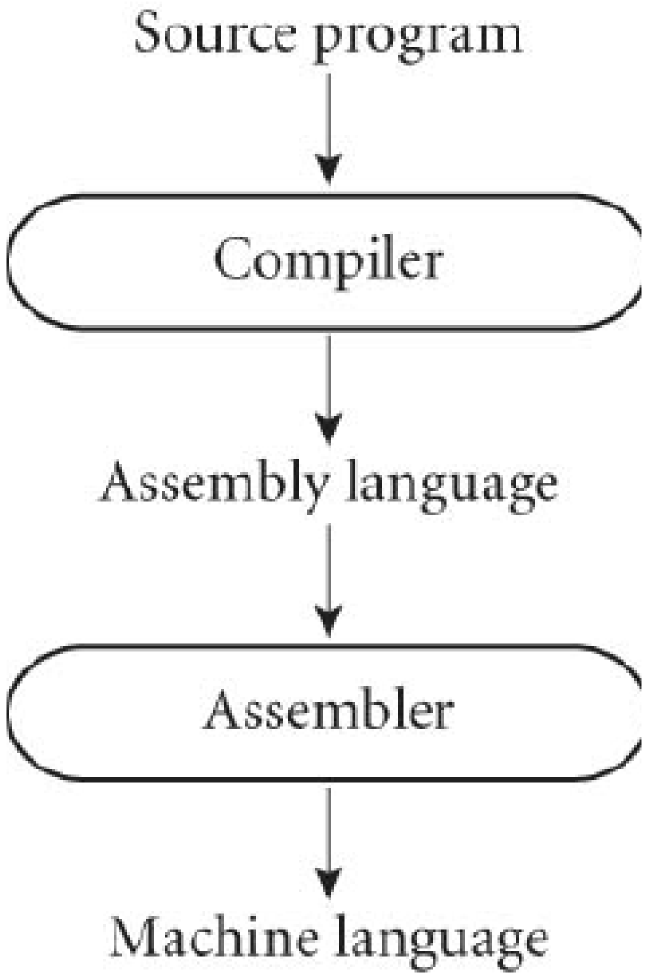
CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016



How does the Magic Box work?

Models of Program Execution

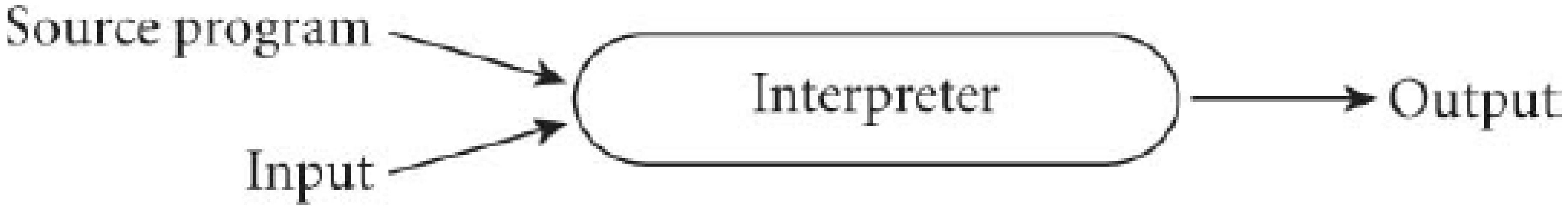
Compile to machine code (e.g., C, C++)



Translates the ***entire*** program ***before*** execution

Models of Program Execution

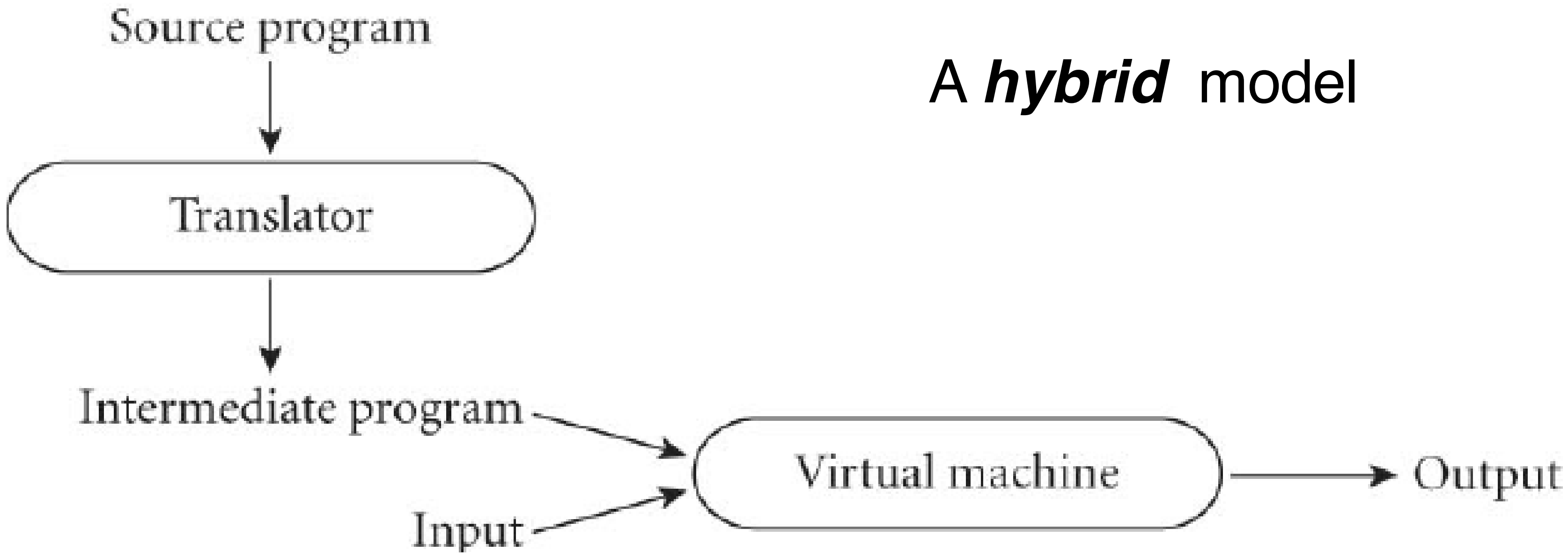
Execute the source code (e.g., Scheme, Python)



Translates ***one line*** at a time
during execution

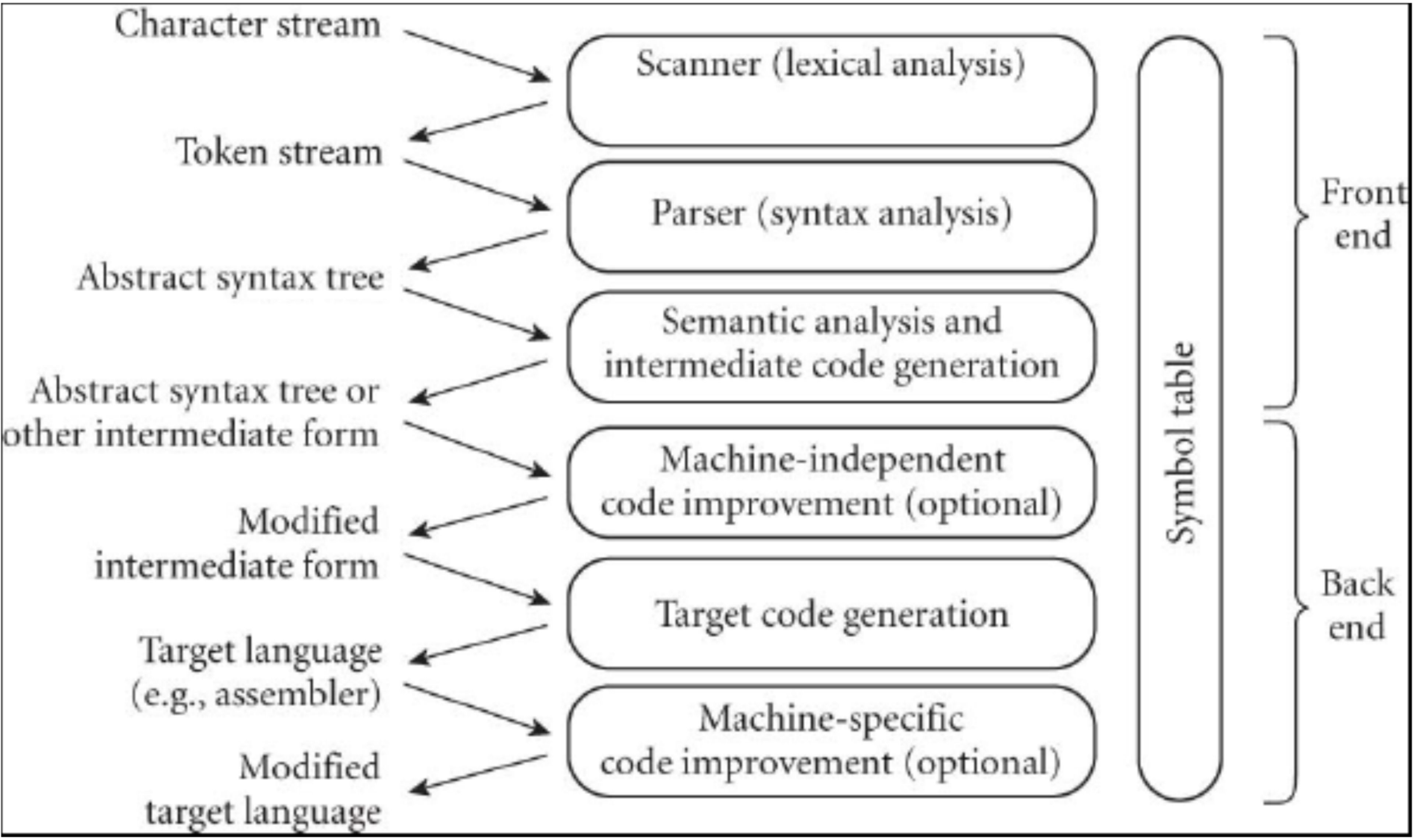
Models of Program Execution

Compile to virtual machine (e.g., Java, C#)

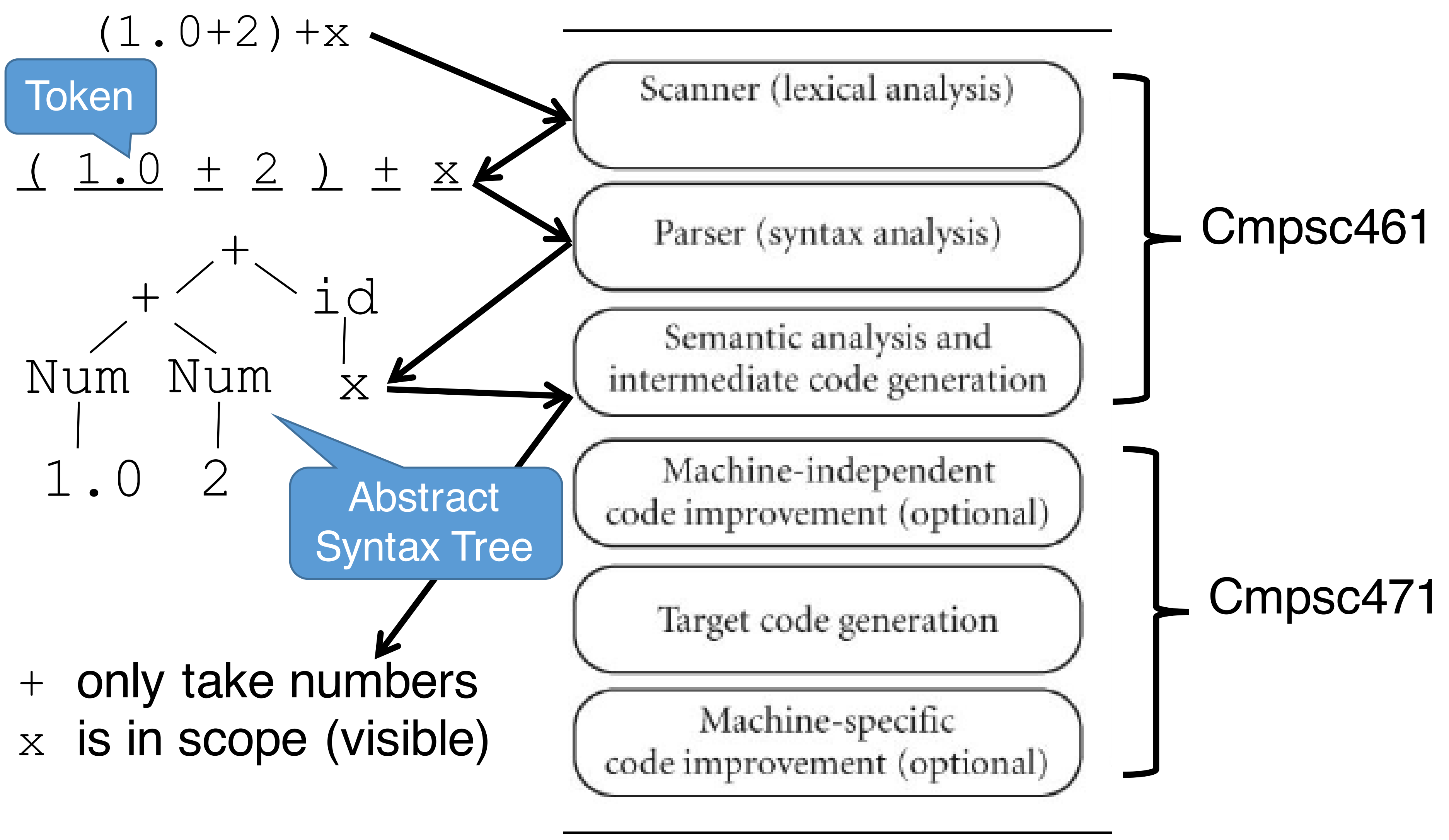


A ***hybrid*** model

Source Code to Target Code


Character stream
Scanner (lexical analysis)
Token stream
Parser (syntax analysis)
Abstract syntax tree
Semantic analysis and intermediate code generation
Abstract syntax tree or other intermediate form
Machine-independent code improvement (optional)
Modified intermediate form
Target code generation
Target language (e.g., assembler)
Machine-specific code improvement (optional)
Modified target language
Symbol table
Front end
Back end

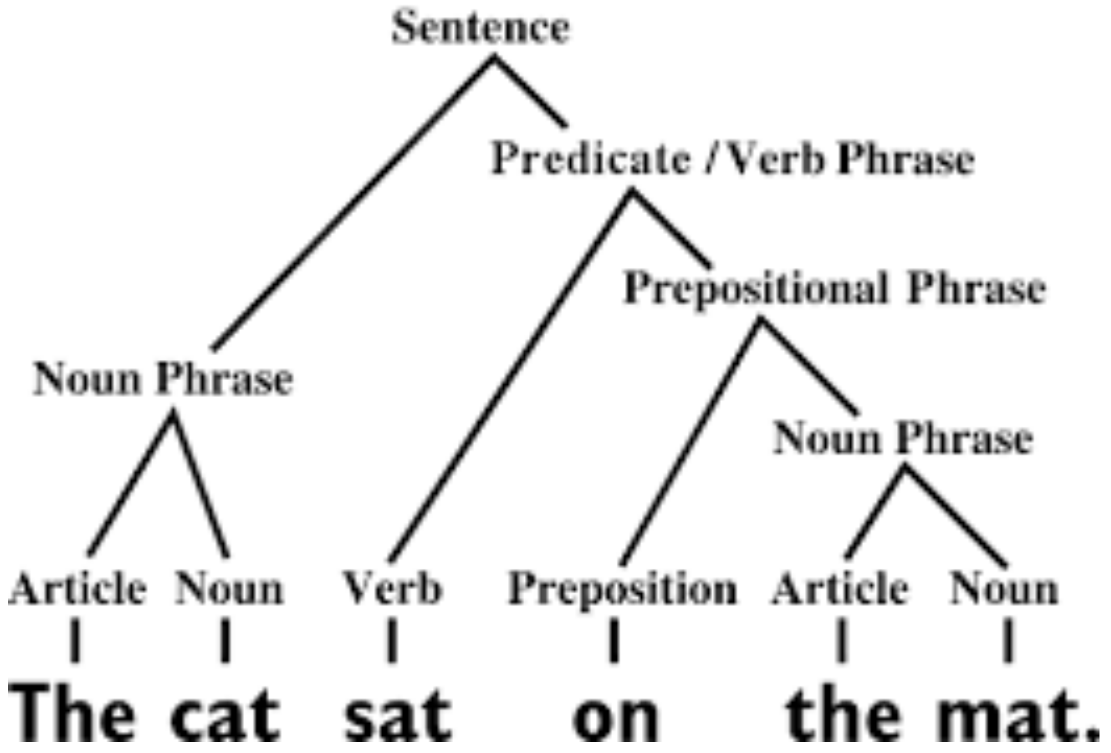
Source Code to Target Code



Syntax vs. Semantics

	Scheme	C
Syntax: how a program is written	`(+ 1 2)`	`(1 + 2)`
Semantics: what’s the meaning of a program	Sum of 1 and 2	

Languages Have Rules



Formal Languages

Language: a set of (legal) strings

Goal: a concise & precise notation for specifying a language

Four levels of languages [Chomsky]:

$$\left.\begin{array}{l}\text{1. Regular}\\ \text{2. Context-Free}\end{array}\right\}\quad\text{programming languages}$$

3. Context-Sensitive
4. Unrestricted

Lexical Syntax

Rules for basic symbols, such as
identifier, literals (e.g., numbers), operators,
keywords, punctuation

C language:

Identifier: letters, digits and underscore '_' only. The first character must be an underscore or a letter

literals: digits, decimal point, suffix such as “l”, “u”

operators: + - * / …

keywords : if, while, for, int, …

punctuation: { } [ ] ; …

Concrete Syntax

Actual representation of programs, using lexical symbols as its alphabet

C language:

IfStatement is a sequence of:

1. IF LPAREN Expression RPAREN Statement, or
2. IF LPAREN Expression RPAREN Statement ELSE statement

Abstract Syntax

A syntax carries only the essential program information (ignores useless info. such as punctuation)

C language:

IfStatement has:
One branch condition (Expression) and one or two statements

Scanner (Lexical Analysis)

From string of characters

```
// hello world
main()  /* main */
{for(;;)
 {printf ("Hello World!\n");}
}
```

To stream of tokens

```
ident(“main”) lparen rparen lbrace for lparen semi semi
rparen lbrace printf lparen string(“Hello World!\n”)
rparen semi rbrace rbrace
```

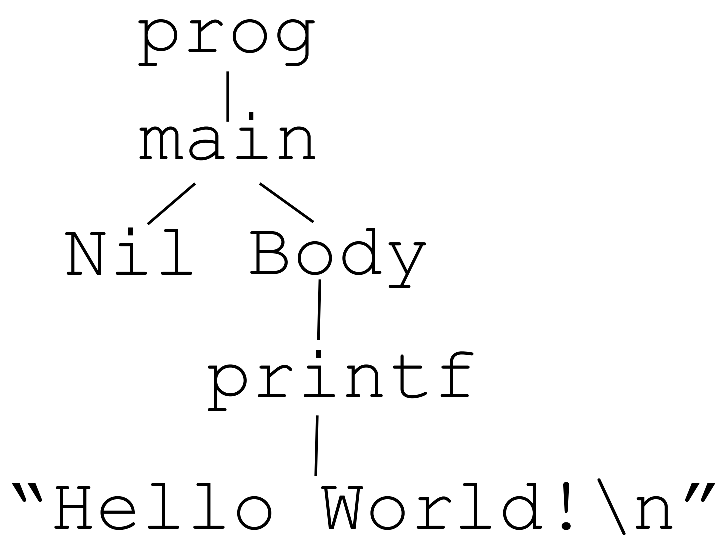
How does the scanner know?
What does the scanner needs to know?

Parser (Syntax Analysis)

From string of characters

```
ident("main") lparen rparen lbrace for lparen semi semi
rparen lbrace printf lparen string("Hello World!\n")
rparen semi rbrace rbrace
```

To *parse tree*



How does the parser know?

What does the parser needs to know?