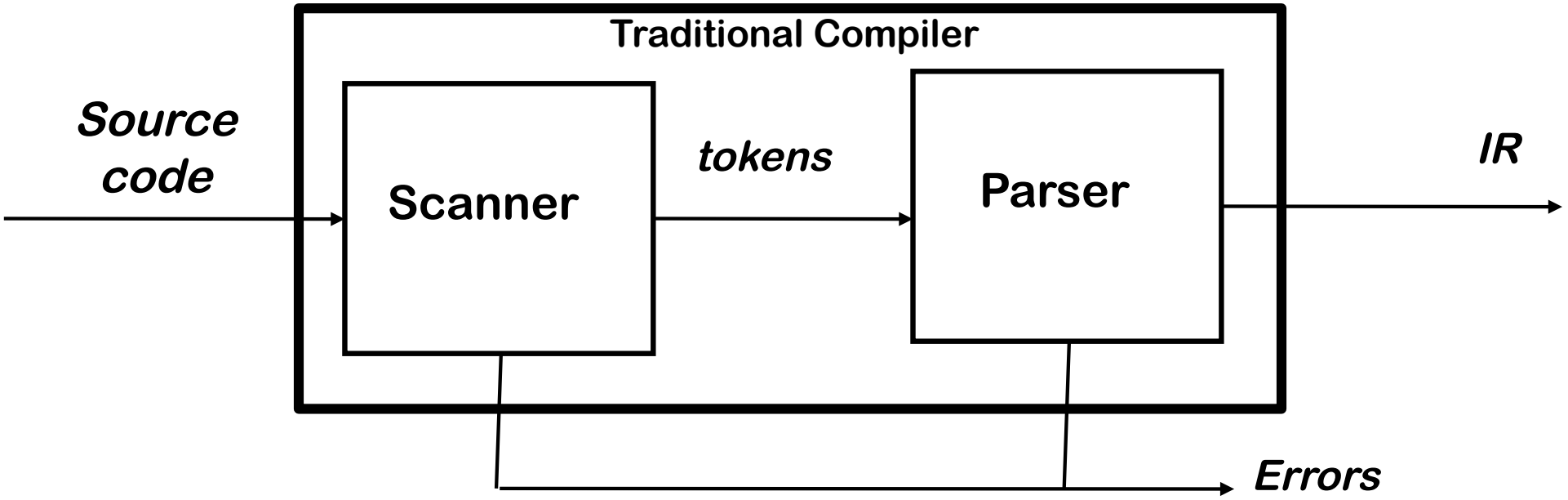
Midterm Review

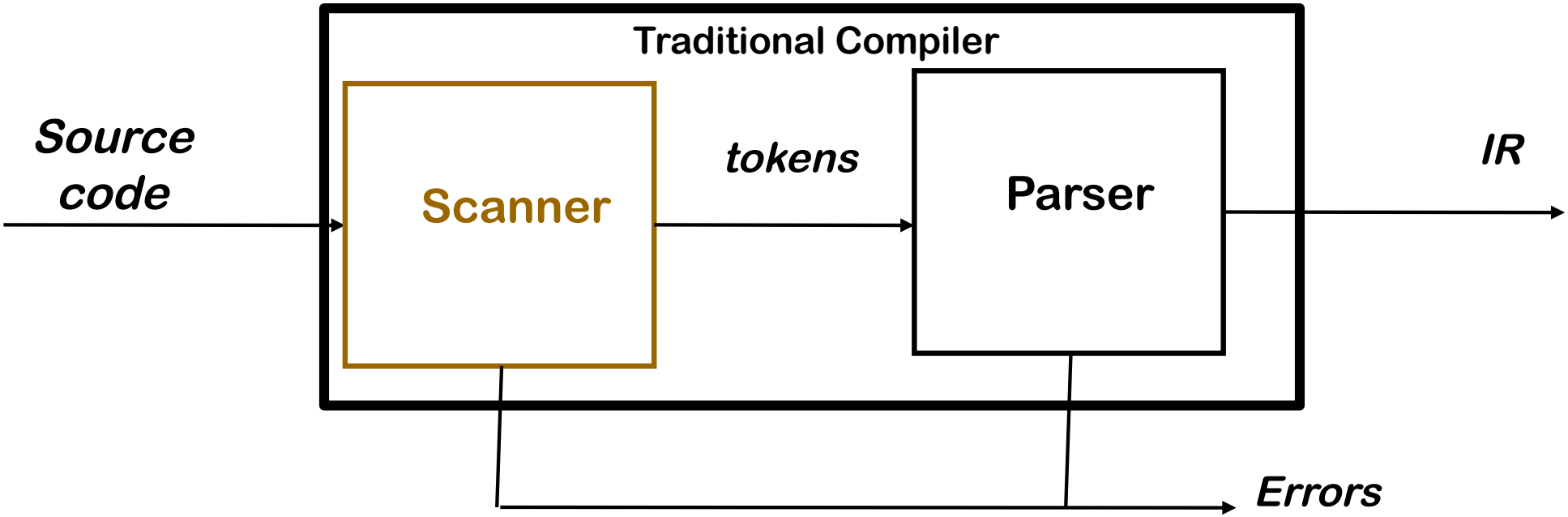
The Front End



Responsibilities

- Recognize legal (and illegal) programs
- Produces IR

The Front End



Scanner

- Maps character stream into words
  - the basic unit of syntax
- Produces pairs — a word & its part of speech

Scanning refers to Lexical Analysis

- Regular Expressions
- Closure, Concatenation, Alternation

RE→ NFA *(Thompson's construction)*

- Build an NFA for each term
- Combine them with ε-transitions

NFA → DFA *(Subset construction)*

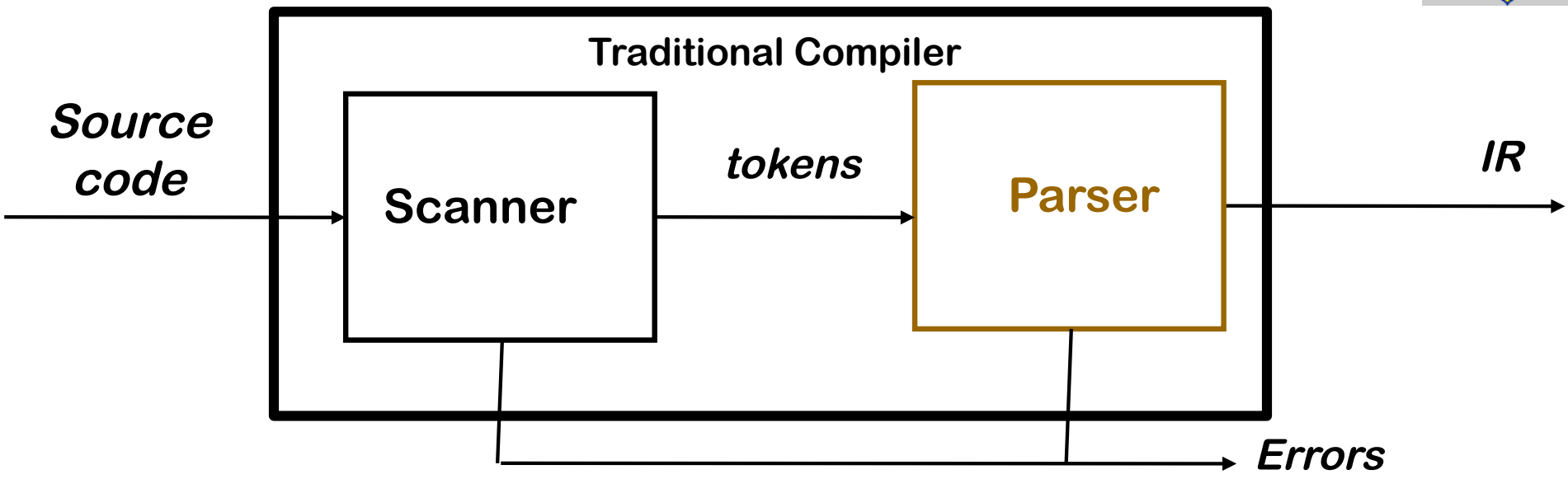
- Build the simulation

DFA → Minimal DFA

- Hopcroft's algorithm

UNDERSTAND THE LEXICAL PHASE OF THE PROJECT!

The Front End



Parser

- Recognizes syntax (context-free) and reports errors
- Builds IR for source program

Backus-Naur Form (BNF)

Formally, a grammar *G = (S,N,T,P)*

- *S* is the *start symbol*
- *N* is a set of *non-terminal symbols*
- *T* is a set of *terminal symbols* or *words*
- *P* is a set of *productions* or *rewrite rules*

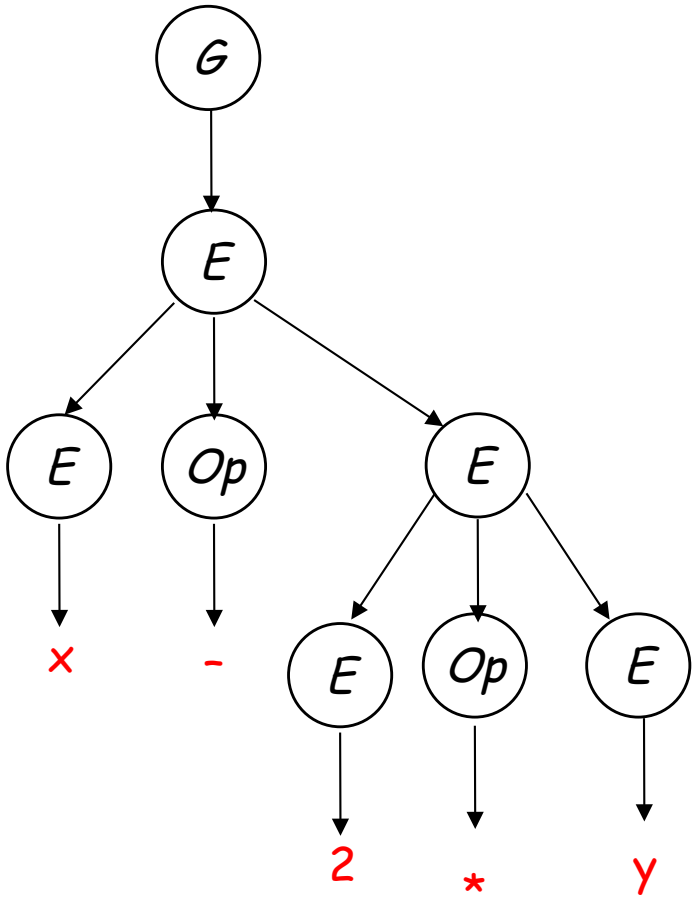
Can you extract these from a complex grammar?

Derivations and Parse Trees

x - 2 * y

Leftmost derivation

Rule	*Sentential Form*
—	*Expr*
1	*Expr Op Expr*
3	<id,x> *Op Expr*
5	<id,x> - *Expr*
1	<id,x> - *Expr Op Expr*
2	<id,x> - <num,2> *Op Expr*
6	<id,x> - <num,2> * *Expr*
3	<id,x> - <num,2> * <id,y>



Given a grammar, can you generate a derivation and a tree?

Ambiguous Grammars

Definitions

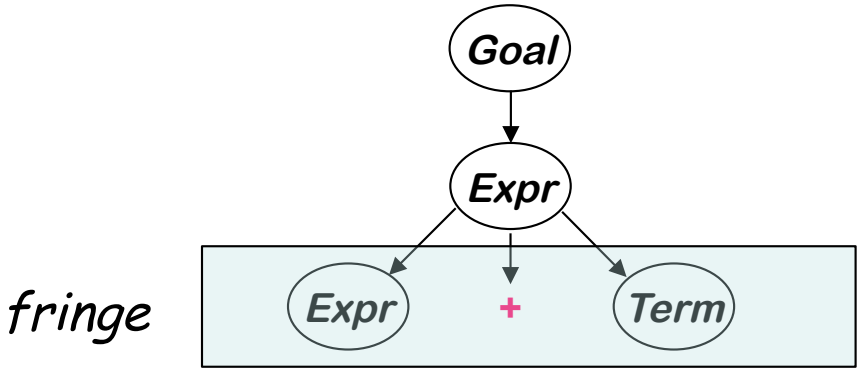
- If a grammar has more than one leftmost derivation for a single *sentential form*, the grammar is *ambiguous*
- If a grammar has more than one rightmost derivation for a single sentential form, the grammar is *ambiguous*
- The leftmost and rightmost derivations for a sentential form may differ, even in an unambiguous grammar

Study how to fix

ambiguous grammar problems

Top-down Parsing

- *Starts with root of parse tree*
- *Root node is labeled with goal symbol*
- *Expand all non-terminals (NT) at fringe of tree*



Top-down parsing algorithm

Construct the root node of parse tree

Repeat until lower fringe matches input string

1. *At node labeled A, select production with A on LHS and, for each symbol on RHS, construct appropriate child*
2. *If terminal symbol added to fringe doesn't match input, backtrack*
3. *Find the next node (NT) to be expanded*

The key is picking the right production in step 1

→ *That choice should be guided by the input string*

Left Recursion

Top-down parsers cannot handle left-recursive grammars

Formally,

A grammar is *left recursive* if $\exists\ A \in NT$ such that

$\exists$ a derivation $A \Rightarrow^{+} A\alpha$, for some string $\alpha \in (NT \cup T)^{+}$

Study how to fix this problem!

What to Study

- Things I've highlighted in green
- Read the book
  - Up to Top-Down Parsing (Section 3.3)
- Study the project solutions