

CS415 Compilers

Syntax Analysis

These slides are based on slides copyrighted by
Keith Cooper, Ken Kennedy & Linda Torczon at Rice
University

Not all languages are regular

$$RL's \subset CFL's \subset CSL's$$

You cannot construct DFA's to recognize these languages

- $L = \{p^k q^k\}$ *(parenthesis languages)*
- $L = \{wcw^r \mid w \in \Sigma^*\}$

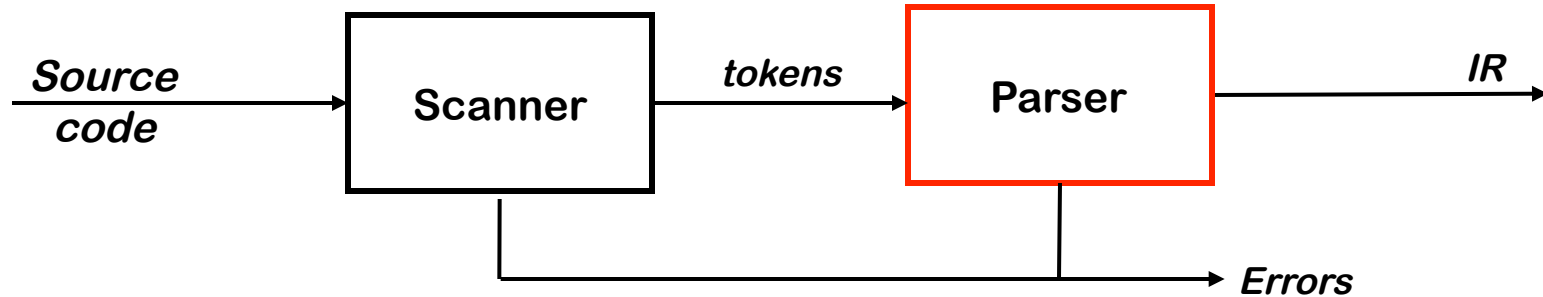
Neither of these is a regular language *(nor an RE)*

But, this is a little subtle. You can construct DFA's for

- Strings with alternating 0's and 1's
 $(\varepsilon \mid 1)(01)^*(\varepsilon \mid 0)$
- Strings with an even number of 0's and 1's
- Strings of bit patterns that represent binary numbers which are divisible by 5

Parsing (Syntax Analysis)

EAC Chapters 3.1 - 3.2



Parser

- Checks the stream of words and their parts of speech (produced by the scanner) for grammatical correctness
- Determines if the input is syntactically well formed
- Guides checking at deeper levels than syntax
- Builds an IR representation of the code

The process of discovering a *derivation* for some sentence

- Need a mathematical model of syntax — a grammar G
- Need an algorithm for testing membership in $L(G)$
- Need to keep in mind that our goal is building parsers, not studying the mathematics of arbitrary languages

Roadmap

- 1 Context-free grammars and derivations
- 2 Top-down parsing
 - LL(1) parsers, hand-coded recursive descent parsers
- 3 Bottom-up parsing
 - Automatically generated LR(1) parsers

Context-free syntax is specified with a context-free grammar

$$\textit{SheepNoise} \rightarrow \textit{SheepNoise} \underline{\textit{baa}}$$

$$\quad \quad \quad | \underline{\textit{baa}}$$

This *CFG* defines the set of noises sheep normally make

It is written in a variant of Backus-Naur form

Formally, a grammar is a four tuple, $G = (S, N, T, P)$

- S is the *start symbol* (*set of strings in $L(G)$*)
- N is a set of *non-terminal symbols* (*syntactic variables*)
- T is a set of *terminal symbols* (*words*)
- P is a set of *productions or rewrite rules* ($P: N \rightarrow (N \cup T)^+$)

We can use the *SheepNoise* grammar to create sentences

→ use the productions as *rewriting rules*

Rule	Sentential Form
—	<i>SheepNoise</i>
2	<u>baa</u>

Rule	Sentential Form
—	<i>SheepNoise</i>
1	<i>SheepNoise</i> <u>baa</u>
2	<u>baa</u> <u>baa</u>

Rule	Sentential Form
—	<i>SheepNoise</i>
1	<i>SheepNoise</i> <u>baa</u>
1	<i>SheepNoise</i> <u>baa</u> <u>baa</u>
2	<u>baa</u> <u>baa</u> <u>baa</u>

And so on ...

To explore the uses of CFGs, we need a more complex grammar

1	$Expr$	\rightarrow	$Expr Op Expr$
2			<u>number</u>
3			<u>id</u>
4	Op	\rightarrow	$+$
5			$-$
6			$*$
7			$/$

Rule	Sentential Form
—	$Expr$
1	$Expr Op Expr$

- Such a sequence of rewrites is called a *derivation*
- Process of discovering a derivation is called *parsing*

We denote this derivation: $Expr \Rightarrow^* \underline{id} - \underline{num} * \underline{id}$

To explore the uses of CFGs, we need a more complex grammar

1	$Expr$	\rightarrow	$Expr Op Expr$
2			<u>number</u>
3			<u>id</u>
4	Op	\rightarrow	+
5			-
6			*
7			/

Rule	Sentential Form
—	$Expr$
1	$Expr Op Expr$
3	$\langle id, \underline{x}, Op Expr \rangle$

- Such a sequence of rewrites is called a *derivation*
- Process of discovering a derivation is called *parsing*

We denote this derivation: $Expr \Rightarrow^* \underline{id} - \underline{num} * \underline{id}$

To explore the uses of CFGs, we need a more complex grammar

1	$Expr$	\rightarrow	$Expr Op Expr$
2			<u>number</u>
3			<u>id</u>
4	Op	\rightarrow	$+$
5			$-$
6			$*$
7			$/$

Rule	Sentential Form
—	$Expr$
1	$Expr Op Expr$
3	$\langle id, \underline{x}, Op Expr \rangle$
5	$\langle id, \underline{x}, - Expr \rangle$

- Such a sequence of rewrites is called a *derivation*
- Process of discovering a derivation is called *parsing*

We denote this derivation: $Expr \Rightarrow^* \underline{id} - \underline{num} * \underline{id}$

To explore the uses of CFGs, we need a more complex grammar

1	<i>Expr</i>	→	<i>Expr Op Expr</i>
2			<u>number</u>
3			<u>id</u>
4	<i>Op</i>	→	+
5			-
6			*
7			/

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

- Such a sequence of rewrites is called a *derivation*
- Process of discovering a derivation is called *parsing*

We denote this derivation: $Expr \Rightarrow^* \underline{id} - \underline{num} * \underline{id}$

- At each step, we choose a non-terminal to replace
- Different choices can lead to different derivations

Two derivations are of interest

- *Leftmost derivation* — replace leftmost NT at each step;
generates left sentential forms (\Rightarrow^*_{lm})
- *Rightmost derivation* — replace rightmost NT at each step;
generates right sentential forms (\Rightarrow^*_{rm})

These are the two *systematic* derivations

(We don't care about randomly-ordered derivations!)

The example on the preceding slide was a *leftmost* derivation

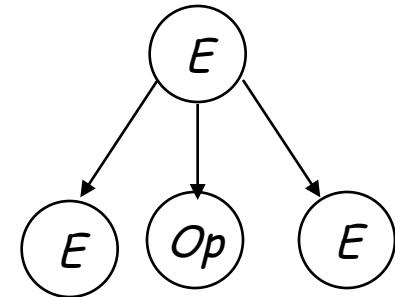
- Of course, there is also a *rightmost* derivation
- Interestingly, the resulting parse trees may be different

Rule in our grammar:
 $E \rightarrow E \text{ Op } E$

A single derivation step

$\dots E \dots \Rightarrow \dots E \text{ Op } E \dots$

can be represented as a tree structure with the left-hand side non-terminal as the root, and all right-hand side symbols as the children (ordered left to right).



The entire derivation of a sentence in the language can be represented as a **parse tree** with the start symbol as its root, and leaf nodes that are all terminal symbols.

NOTE: The structure of the parse tree has semantic significance!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
—	— — —

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
—	— — —

Rightmost derivation

In both cases, $Expr \Rightarrow^* \underline{id} - \underline{num} * \underline{id}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
.	— — —

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
.	- - -

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
.	- - -

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
.	- - -

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	<i>Expr Op</i> $\langle \text{id}, \underline{y} \rangle$

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	<i>Expr Op</i> $\langle \text{id}, \underline{y} \rangle$
6	<i>Expr</i> * $\langle \text{id}, \underline{y} \rangle$

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	<i>Expr Op</i> $\langle \text{id}, \underline{y} \rangle$
6	<i>Expr</i> * $\langle \text{id}, \underline{y} \rangle$
1	<i>Expr Op Expr</i> * $\langle \text{id}, \underline{y} \rangle$

Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Leftmost derivation

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	<i>Expr Op</i> $\langle \text{id}, \underline{y} \rangle$
6	<i>Expr</i> * $\langle \text{id}, \underline{y} \rangle$
1	<i>Expr Op Expr</i> * $\langle \text{id}, \underline{y} \rangle$
2	<i>Expr Op</i> $\langle \text{num}, \underline{2} \rangle$ * $\langle \text{id}, \underline{y} \rangle$
5	<i>Expr</i> - $\langle \text{num}, \underline{2} \rangle$ * $\langle \text{id}, \underline{y} \rangle$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

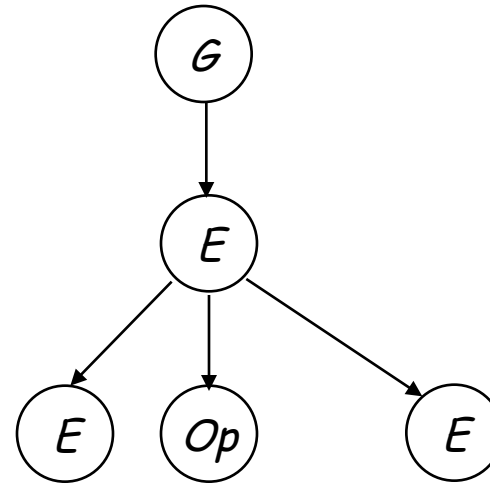
Rightmost derivation

In both cases, $\text{Expr} \Rightarrow^* \underline{\text{id}} - \underline{\text{num}} * \underline{\text{id}}$

- The two derivations produce different parse trees
- The parse trees imply different evaluation orders!

Leftmost derivation

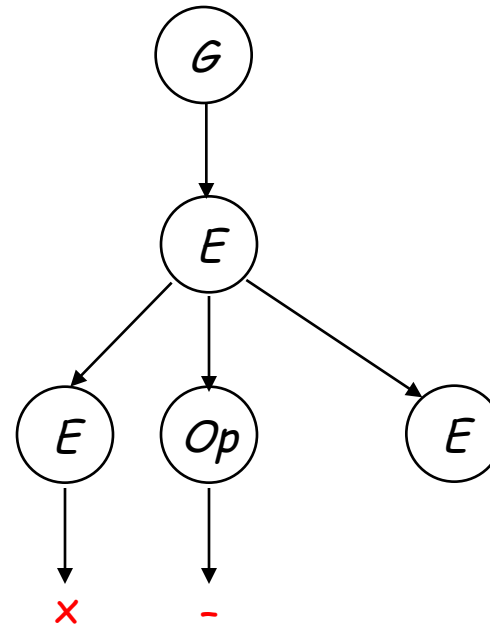
<i>Rule</i>	<i>Sentential Form</i>
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$



This evaluates as $\underline{x} - (\underline{2} * \underline{y})$

Leftmost derivation

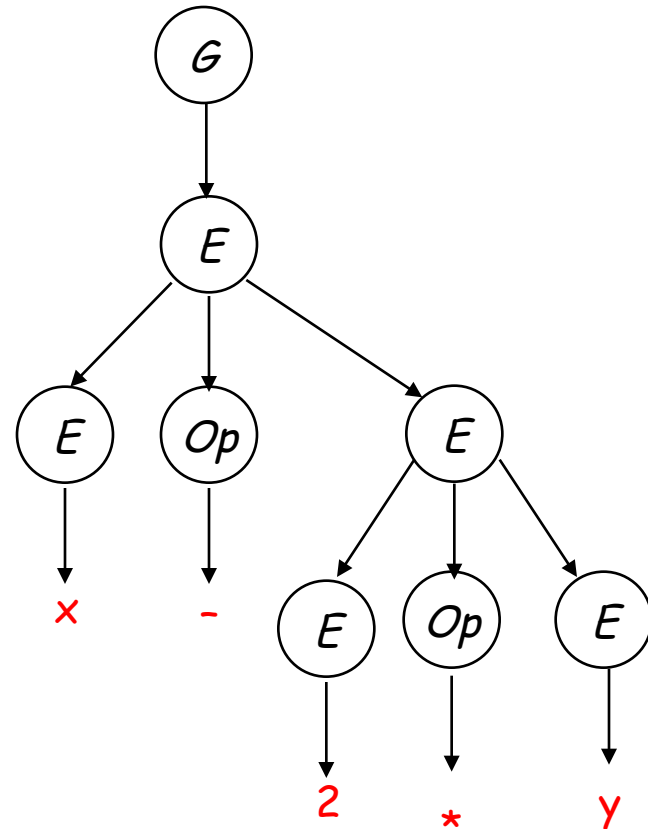
Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$



This evaluates as $\underline{x} - (\underline{2} * \underline{y})$

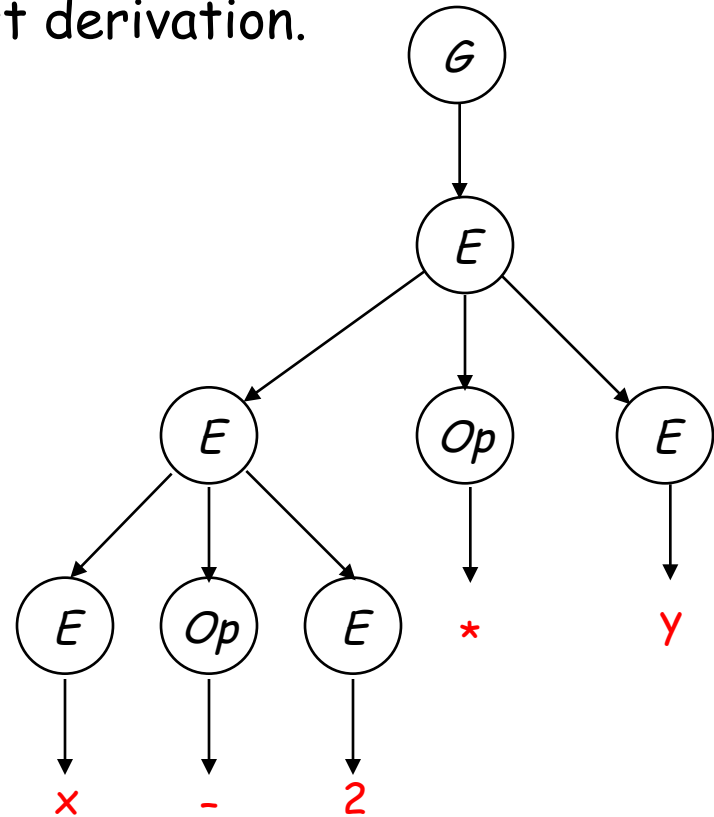
Leftmost derivation

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



This evaluates as $\underline{x} - (\underline{2} * \underline{y})$

This corresponds to our rightmost derivation.
 Can we get this with another
 leftmost derivation as well?



This evaluates as $(\underline{x} - \underline{2}) * y$

The Difference:

- Different productions chosen on the second step

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
③	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr}$
1	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

Original choice

Rule	Sentential Form
—	<i>Expr</i>
1	<i>Expr Op Expr</i>
①	<i>Expr Op Expr Op Expr</i>
3	$\langle \text{id}, \underline{x} \rangle \text{ Op Expr Op Expr}$
5	$\langle \text{id}, \underline{x} \rangle - \text{Expr Op Expr}$
2	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle \text{ Op Expr}$
6	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \text{Expr}$
3	$\langle \text{id}, \underline{x} \rangle - \langle \text{num}, \underline{2} \rangle * \langle \text{id}, \underline{y} \rangle$

New choice

- Both derivations succeed in producing $x - 2 * y$

- This grammar allows multiple leftmost derivations for $x - 2 * y$
- Hard to automate derivation if > 1 choice
- The grammar is *ambiguous*

1	$Expr \rightarrow$	$Expr Op Expr$
2		<u>number</u>
3		<u>id</u>
4	$Op \rightarrow$	$+$
5		$-$
6		$*$
7		$/$

Rule	Sentential Form
—	$Expr$
1	$Expr Op Expr$
①	$Expr Op Expr Op Expr$
3	$\langle id, \underline{x} \rangle Op Expr Op Expr$
5	$\langle id, \underline{x} \rangle - Expr Op Expr$
2	$\langle id, \underline{x} \rangle - \langle num, \underline{2} \rangle Op Expr$
6	$\langle id, \underline{x} \rangle - \langle num, \underline{2} \rangle * Expr$
3	$\langle id, \underline{x} \rangle - \langle num, \underline{2} \rangle * \langle id, \underline{y} \rangle$

different choice
than the first time

*These two derivations point out a problem with the grammar:
It has no notion of precedence, or implied order of evaluation*

To add precedence

- Create a non-terminal for each *level of precedence*
- Isolate the corresponding part of the grammar
- Force the parser to recognize high precedence subexpressions first

For algebraic expressions

- Multiplication and division, first *(level one)*
- Subtraction and addition, next *(level two)*

Adding the standard algebraic precedence produces:

level two	1	<i>Goal</i>	→	<i>Expr</i>
	2	<i>Expr</i>	→	<i>Expr</i> + <i>Term</i>
	3			<i>Expr</i> - <i>Term</i>
level one	4			<i>Term</i>
	5	<i>Term</i>	→	<i>Term</i> * <i>Factor</i>
	6			<i>Term</i> / <i>Factor</i>
	7			<i>Factor</i>
	8	<i>Factor</i>	→	<u>number</u>
	9			<u>id</u>

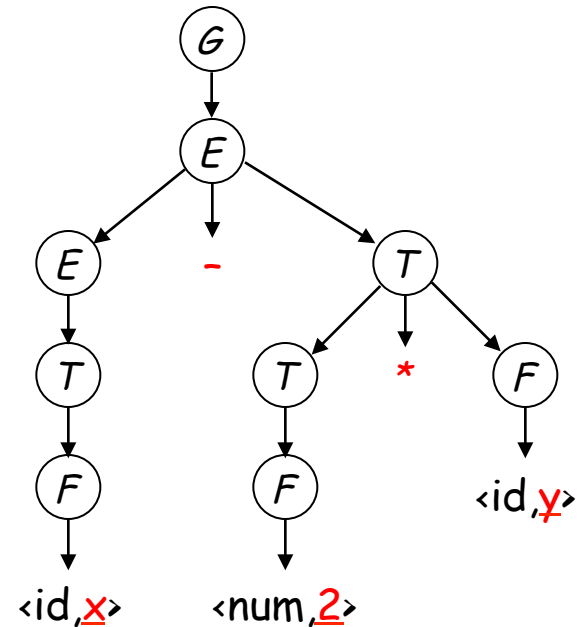
This grammar is slightly larger

- Takes more rewriting to reach some of the terminal symbols
- Encodes expected precedence
- Produces same parse tree under leftmost & rightmost derivations

*Let's see how it parses $x - 2 * y$*

Rule	Sentential Form
—	Goal
1	Expr
3	Expr - Term
5	Expr - Term * Factor
9	Expr - Term * <id,y>
7	Expr - Factor * <id,y>
8	Expr - <num,z> * <id,y>
4	Term - <num,z> * <id,y>
7	Factor - <num,z> * <id,y>
9	<id,x> - <num,z> * <id,y>

The rightmost derivation



Its parse tree

This produces $x - (z * y)$, along with an appropriate parse tree. Both the leftmost and rightmost derivations give the same expression, because the grammar directly encodes the desired precedence.

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

Classic example — the *if-then-else* problem

$$\begin{array}{l} Stmt \rightarrow \underline{\text{if}} \text{ Expr } \underline{\text{then}} \text{ Stmt} \\ \quad | \underline{\text{if}} \text{ Expr } \underline{\text{then}} \text{ Stmt } \underline{\text{else}} \text{ Stmt} \\ \quad | \dots \text{ other stmts } \dots \end{array}$$

This ambiguity is entirely grammatical in nature

This sentential form has two derivations

if $Expr_1$ then if $Expr_2$ then $Stmt_1$ else $Stmt_2$

Removing the ambiguity

- Must rewrite the grammar to avoid generating the problem
- Match each else to innermost unmatched if (*common sense rule*)

1	$Stmt \rightarrow WithElse$
2	$NoElse$
3	$WithElse \rightarrow \underline{if} \ Expr \ \underline{then} \ WithElse \ \underline{else} \ WithElse$
4	$OtherStmt$
5	$NoElse \rightarrow \underline{if} \ Expr \ \underline{then} \ Stmt$
6	$\underline{if} \ Expr \ \underline{then} \ WithElse \ \underline{else} \ NoElse$

 $\underline{if} \ Expr_1 \ \underline{then} \ \underline{if} \ Expr_2 \ \underline{then} \ Stmt_1 \ \underline{else} \ Stmt_2$

With this grammar, the example has only one derivation

if $Expr_1$ then if $Expr_2$ then $Stmt_1$ else $Stmt_2$

Rule	Sentential Form
—	$Stmt$
2	$NoElse$
5	<u>if</u> $Expr$ <u>then</u> $Stmt$
?	<u>if</u> E_1 <u>then</u> $Stmt$
1	<u>if</u> E_1 <u>then</u> $WithElse$
3	<u>if</u> E_1 <u>then</u> <u>if</u> $Expr$ <u>then</u> $WithElse$ <u>else</u> $WithElse$
?	<u>if</u> E_1 <u>then</u> <u>if</u> E_2 <u>then</u> $WithElse$ <u>else</u> $WithElse$
4	<u>if</u> E_1 <u>then</u> <u>if</u> E_2 <u>then</u> S_1 <u>else</u> $WithElse$
4	<u>if</u> E_1 <u>then</u> <u>if</u> E_2 <u>then</u> S_1 <u>else</u> S_2

This binds the else controlling S_2 to the inner if

Ambiguity usually refers to confusion in the CFG

Overloading can create deeper ambiguity

$$a = f(17)$$

In many Algol-like languages, f could be either a function or a subscripted variable

Disambiguating this one requires context

- Need values of declarations
- Really an issue of *type*, not context-free syntax
- Requires an extra-grammatical solution (not in CFG)
- Must handle these with a different mechanism
 - Step outside grammar rather than use a more complex grammar

Ambiguity arises from two distinct sources

- Confusion in the context-free syntax (*if-then-else*)
- Confusion that requires context to resolve (*overloading*)

Resolving ambiguity

- To remove context-free ambiguity, rewrite the grammar
- Change language (e.g.: if ... endif)
- To handle context-sensitive ambiguity takes cooperation
 - Knowledge of declarations, types, ...
 - Accept a superset of $L(G)$ & check it by other means[†]
 - This is a language design problem

[†]See Chapter 4

Top-down parsers (LL(1), recursive descent)

- Start at the root of the parse tree and grow toward leaves
- Pick a production & try to match the input
- Bad “pick” \Rightarrow may need to backtrack
- Some grammars are backtrack-free *(predictive parsing)*

Bottom-up parsers (LR(1), operator precedence)

- Start at the leaves and grow toward root
- As input is consumed, encode possibilities in an internal state
- Start in a state valid for legal first tokens
- Bottom-up parsers handle a large class of grammars

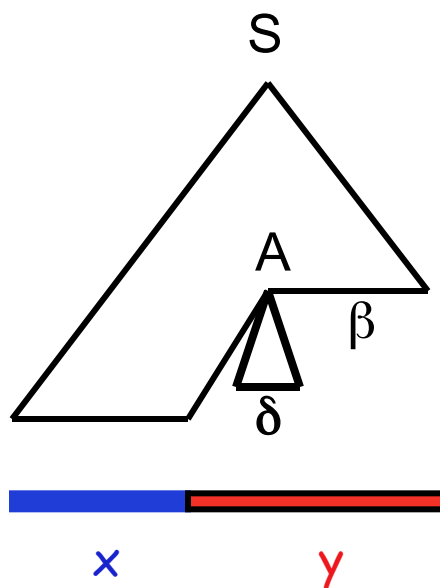
LL(1), recursive descent

1 input symbol lookahead

construct leftmost derivation (forwards)

input: read left-to-right

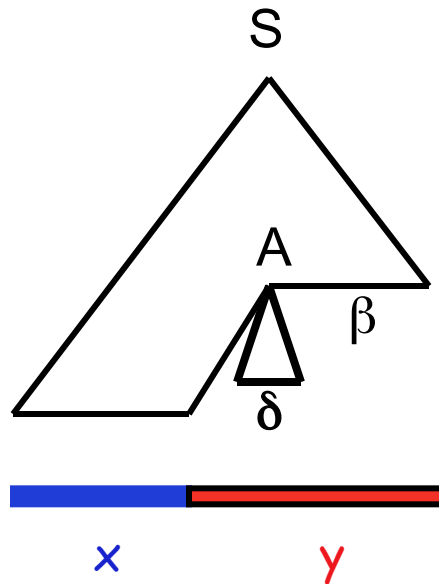
$$S \Rightarrow_{lm}^* x A \beta \Rightarrow_{lm} x \delta \beta \Rightarrow_{lm}^* x y$$



LL(1), recursive descent

1 input symbol lookahead
 construct leftmost derivation (forwards)
 input: read left-to-right

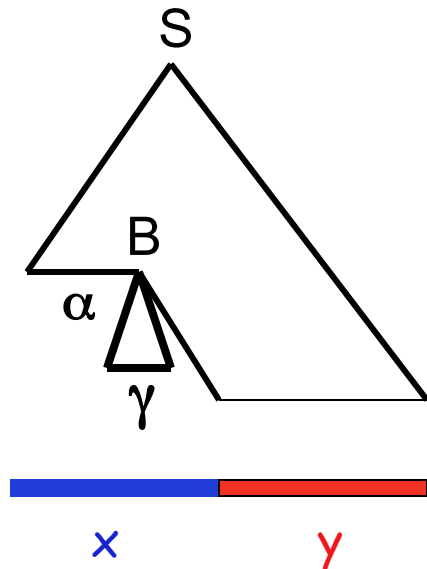
$$S \Rightarrow_{lm}^* x \textcircled{A} \beta \xrightarrow{\text{rule } A \rightarrow \delta} \Rightarrow_{lm} x \textcircled{\delta} \beta \Rightarrow_{lm}^* x y$$



LR(1), operator precedence

1 input symbol lookahead
 construct rightmost derivation (backwards)
 input: read left-to-right

$$S \Rightarrow_{rm}^* \alpha B y \Rightarrow_{rm} \alpha \gamma y \Rightarrow_{rm}^* x y$$

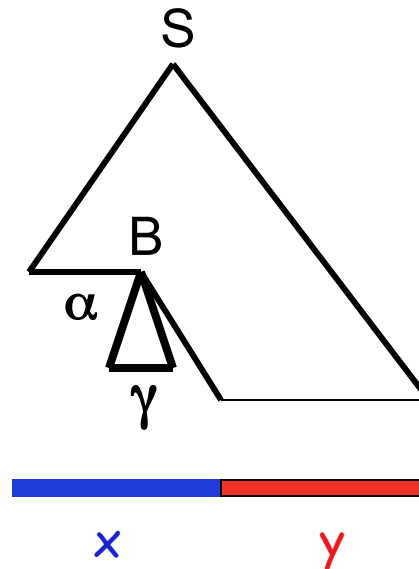


LR(1), operator precedence

1 input symbol lookahead
 construct rightmost derivation (backwards)
 input: read left-to-right

$$S \Rightarrow_{rm}^* \alpha(B)y \xRightarrow{rm} \alpha(\gamma)y \xRightarrow{rm}^* xy$$

rule $B \rightarrow \gamma$



- Homework 3 posted
 - Deadline extended to this Sunday 2/14 11:59pm EST
- Next class

Top-down and bottom-up Parsing

Read EaC: 3.1 - 3.3