Derivations

Grammar for Finite Language {bne, beq}

- Generating a finite language with a CFG is straight-forward

G: (R1) $S \rightarrow$ *bne*
(R2) $S \rightarrow$ *beq*

- derive bne: $S \Rightarrow$ *bne*
- derive beq: $S \Rightarrow$ *beq*

Derivations

Grammar for Language on {a, b} that Contains at Least One *a*

- The terminals (a, b) can appear to the *left* of the non-terminals.

G:
(R1) $S \rightarrow bS$
(R2) $S \rightarrow aA$
(R3) $A \rightarrow aA$
(R4) $A \rightarrow bA$
(R5) $A \rightarrow \varepsilon$

Think of the non-terminal S as representing "have not generated an *a* yet" and A as "have generated an *a*."

- derive bbab:

$$S \underset{R1}{\Rightarrow} bS \underset{R1}{\Rightarrow} bbS \underset{R2}{\Rightarrow} bbaA \underset{R4}{\Rightarrow} bbabA \underset{R5}{\Rightarrow} bbab$$

- derive aaba:

$$S \underset{R2}{\Rightarrow} aA \underset{R3}{\Rightarrow} aaA \underset{R4}{\Rightarrow} aabA \underset{R3}{\Rightarrow} aabaA \underset{R5}{\Rightarrow} aaba$$

Derivations

Grammar for Language on {a, b} that Contains at Least One *a*

- The terminals (*a*, *b*) can appear to the *right* of the non-terminals.

G:	(R1)	S	$\rightarrow$	S*b*
	(R2)	S	$\rightarrow$	A*a*
	(R3)	A	$\rightarrow$	A*a*
	(R4)	A	$\rightarrow$	A*b*
	(R5)	A	$\rightarrow$	ε

Think of the non-terminal S as representing "have not generated an *a* yet" and A as "have generated an *a*."

- derive bbab:

$$S \underset{R1}{\Rightarrow} Sb \underset{R2}{\Rightarrow} Aab \underset{R4}{\Rightarrow} Abab \underset{R4}{\Rightarrow} Abbab \underset{R5}{\Rightarrow} bbab$$

- derive aaba:

$$S \underset{R2}{\Rightarrow} Aa \underset{R4}{\Rightarrow} Aba \underset{R3}{\Rightarrow} Aaba \underset{R3}{\Rightarrow} Aaaba \underset{R5}{\Rightarrow} aaba$$

Derivations

Grammar for Language on {a, b} that Contains an Even # of *a's*

G:	(R1)	S	$\rightarrow$	*b*S
	(R2)	S	$\rightarrow$	S*b*
	(R3)	S	$\rightarrow$	*a*S*a*
	(R4)	S	$\rightarrow$	ε

$\}$ The *a*'s are generated in pairs, from the centre outwards.

- derive baa: S $\Rightarrow$ *b*S $\Rightarrow$ *ba*S*a* $\Rightarrow$ *baa*
- derive aab: S $\Rightarrow$ S*b* $\Rightarrow$ *a*S*ab* $\Rightarrow$ *aab*
- derive babaaba:
  hint: since *a*'s are generated in pairs start at the outside and work your way towards the middle of the *a*'s
  S $\Rightarrow$ *b*S $\Rightarrow$ *ba*S*a* $\Rightarrow$ *bab*S*a* $\Rightarrow$ *bab*S*ba* $\Rightarrow$ *baba*S*aba* $\Rightarrow$ *babaaba*

Derivations

Grammar for Language on {a, b} that Contains an Even # of *a's*

G: (R1) $S \rightarrow bS$
(R2) $S \rightarrow Sb$
(R3) $S \rightarrow aSa$
(R4) $S \rightarrow \varepsilon$

The *a*'s are generated in pairs, from the centre outwards.

- The string *aba* has *two different* derivations

1. $S \underset{R3}{\Rightarrow} aSa \underset{R1}{\Rightarrow} abSa \underset{R4}{\Rightarrow} aba$

2. $S \underset{R3}{\Rightarrow} aSa \underset{R2}{\Rightarrow} aSba \underset{R4}{\Rightarrow} aba$

- When a grammar has two different derivations for the same string the grammar is called *ambiguous.* More on this later.

Another Example CFG

Binary Expressions

- In this language the words are binary numbers with no leading 0's (other than 0) and with + or – operators using infix notation (between numbers, not before them)

1. $E \rightarrow E + E$
2. $E \rightarrow E - E$
3. $E \rightarrow B$
4. $B \rightarrow 0$
5. $B \rightarrow D$
6. $D \rightarrow 1$
7. $D \rightarrow D0$
8. $D \rightarrow D1$

Here

- E means expression
- B means generate a 0 or D
- D means generate a number with a leading 1

Another Example CFG

Binary Expressions

- Derive: 0

  $E \Rightarrow B \Rightarrow 0$

- Derive: 1

  $E \Rightarrow B \Rightarrow D \Rightarrow 1$

- Derive: 10

  $E \Rightarrow B \Rightarrow D \Rightarrow D0 \Rightarrow 10$

- Derive: 101

  $E \Rightarrow B \Rightarrow D \Rightarrow D1 \Rightarrow D01 \Rightarrow 101$

Another Example CFG

Binary Expressions

- Derive: 10+1 using a *leftmost derivation* (i.e. always expand the leftmost non-terminal first)

- $E \overset{(1)}{\Rightarrow} E + E \overset{(3)}{\Rightarrow} B + E \overset{(5)}{\Rightarrow} D + E \overset{(7)}{\Rightarrow} D0 + E \overset{(6)}{\Rightarrow} 10 + E \overset{(3)}{\Rightarrow}$
  $10 + B \overset{(5)}{\Rightarrow} 10 + D \overset{(6)}{\Rightarrow} 10 + 1$

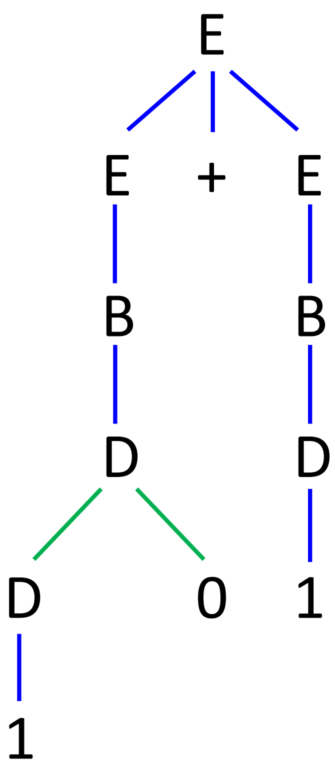
- Derive: 10+1 using a *rightmost derivation* (i.e. always expand the rightmost non-terminal first)

- $E \overset{(1)}{\Rightarrow} E + E \overset{(3)}{\Rightarrow} E + B \overset{(5)}{\Rightarrow} E + D \overset{(6)}{\Rightarrow} E + 1 \overset{(3)}{\Rightarrow}$
  $B + 1 \overset{(5)}{\Rightarrow} D + 1 \overset{(7)}{\Rightarrow} D0 + 1 \overset{(6)}{\Rightarrow} 10 + 1$

Parse Tree

A Parse Tree for E ⇒* 10 + 1

The derivation E ⇒* 10 + 1 can be represented as a *parse tree.*



R1. E → E + E

R3. E → B

R5. B → D

R7. D → D0, R6. D → 1

R6. D → 1

Parse Trees

Creating a Parse Tree

- also called derivation trees
- visualize entire derivation at once
- *internal nodes* are the non-terminals: E, B, D
- the *root of the tree* is the start symbol: E
- *children* of a node are given by derivation rule
- *leaf nodes* are the terminals and show their value: 1, 0, +, 1 in the same order as in the expression
- parse trees (among other things) help visualize *ambiguous grammars*…

Ambiguous Grammars

Grammars

- Statements in English can be *ambiguous*.
- Chris was given a book by J. K. Rowlings.
  - Does *by* refer to *a book*?
    - i.e. The book was by J. K. Rowlings.
  - Does *by* refer to *was given*?
    - i.e. The book was given by J. K. Rowlings.
- Grammars for computer languages are at risk of being ambiguous: e.g. 1 - 10 + 11
- Does the grammar interpret the statement as "(1 – 10) + 11" or "1 – (10 + 11)" or both?

Ambiguous Grammars

Parse Trees for E $\Rightarrow^*$ 1 - 10 + 11

- The same string can have two different parse trees.
- This grammar is *ambiguous.*

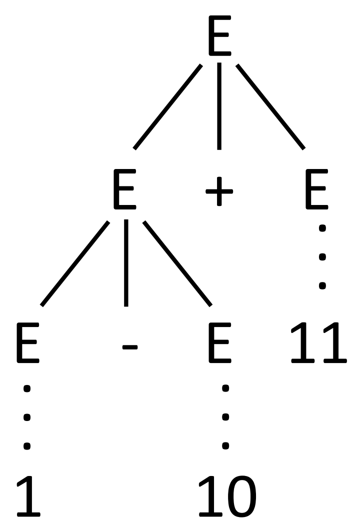
  R1. E $\rightarrow$ E + E

  R2. E $\rightarrow$ E - E

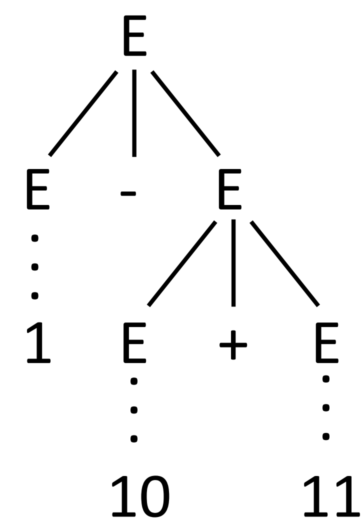
- You can use

  a) R1 then R2 or

  b) R2 then R1.



a) R1 then R2



b) R2 then R1

Ambiguous Grammars

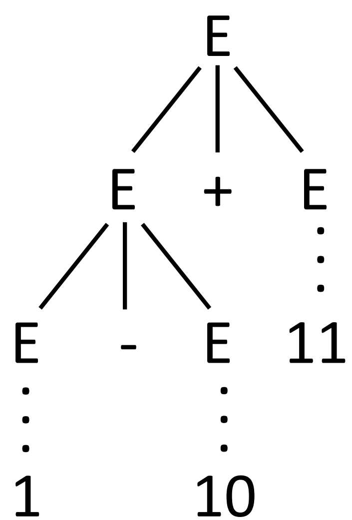
Parse Trees for $E \Rightarrow^* 1 - 10 + 11$

- You may also have *two or more leftmost derivations* (and two or more rightmost derivations)

$E \Rightarrow E{+}E \Rightarrow E{-}E{+}E \Rightarrow B{-}E{+}E$

$\Rightarrow D{-}E{+}E \Rightarrow 1{-}E{+}E \Rightarrow \ldots$

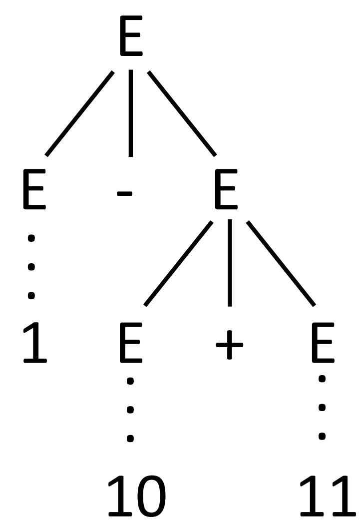
yields the left tree



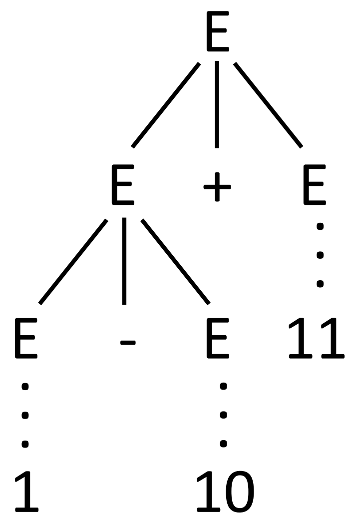
$E \Rightarrow E{-}E \Rightarrow B{-}E \Rightarrow D{-}E \Rightarrow 1{-}E$

$\Rightarrow 1{-}E{+}E \Rightarrow 1{-}B{+}E \Rightarrow 1{-}D{+}E \ldots$

yields the right tree



Processing Order in a Parse Tree

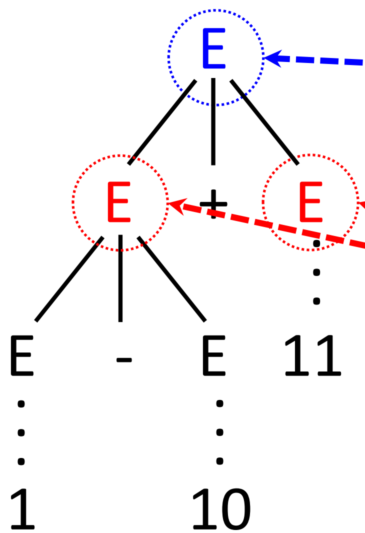


Trees are processed *post order* with a *depth first* traversal.

depth first – go as deep as you can with your first child before visiting your second child.

post order – process all your children before you processing yourself.

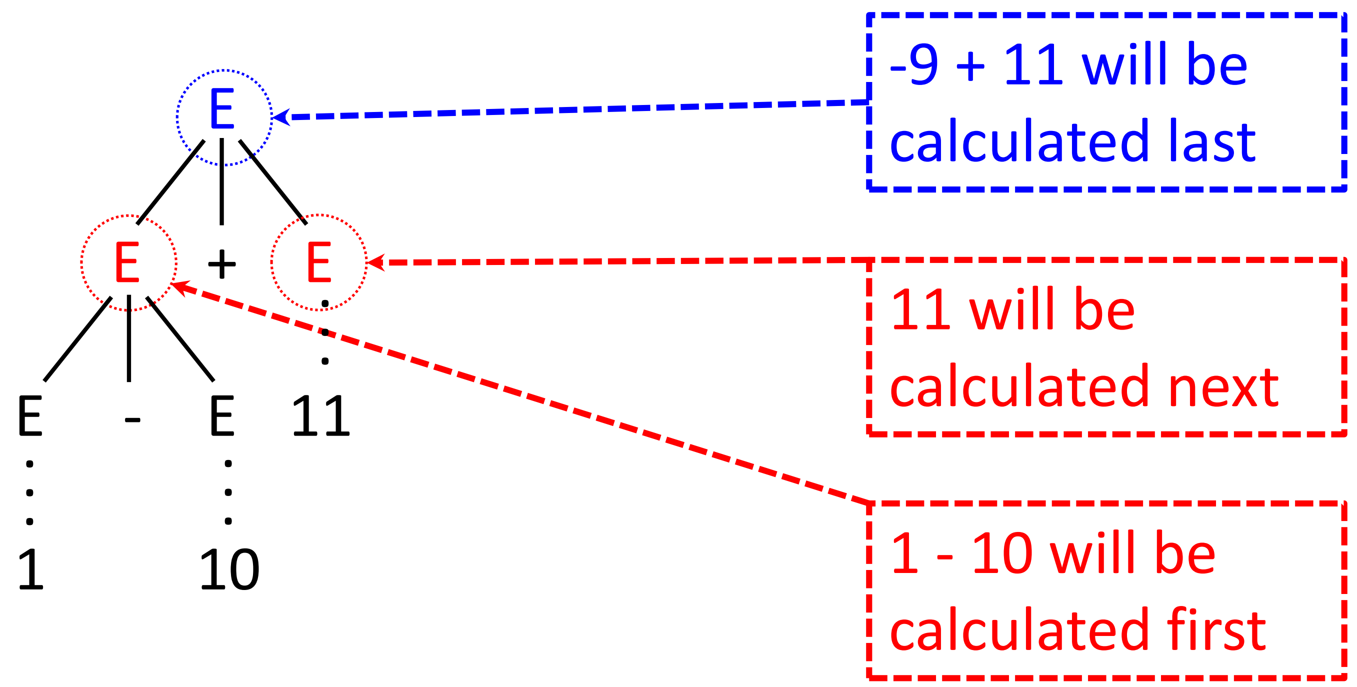
Processing Order in a Parse Tree



This parent will be processed *after* all its children are processed.

The children will be processed *before* the parent.

Processing Order in a Parse Tree



Ambiguous Grammars

Formal Definition

- A *string w is ambiguous* if there is more than one parse tree for w.
- e.g. the string "1 - 10 + 11" is ambiguous.
- A *context-free grammar G is ambiguous* if there exists at least one string w such that $w \in \mathcal{L}(G)$ and w is ambiguous.
- e.g. the grammar that generated the string "1 - 10 + 11" is ambiguous.

Ambiguous Grammars

Ambiguity

- *Ambiguous grammars means there is no unique derivation.*
- When is a CFG ambiguous?
  - ultimately undecidable (like the Halting Problem)
  - certain ambiguities can be spotted
  - e.g. the same non-terminals in the RHS of a rule, as seen is rules 1 and 2 below:
    1. $E \rightarrow E + E$
    2. $E \rightarrow E - E$
- i.e. either the operator ‘+’ or ‘-’ can come first
- left recursion and right recursion causes ambiguity

Unambiguous Grammars

Binary Expressions

- Change the first two productions

1. $E \rightarrow \cancel{E + E}\ B + E$
2. $E \rightarrow \cancel{E - E}\ B - E$
3. $E \rightarrow B$
4. $B \rightarrow 0$
5. $B \rightarrow D$
6. $D \rightarrow 1$
7. $D \rightarrow D0$
8. $D \rightarrow D1$

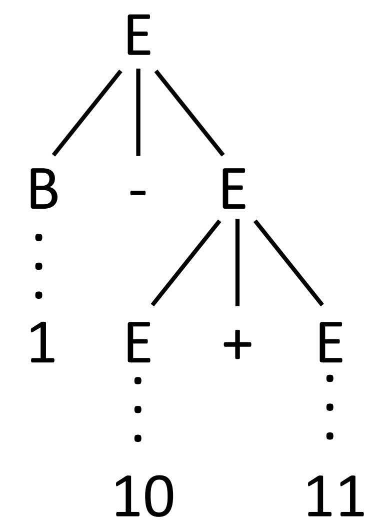
- This change forces the leftmost non-terminal to derive a binary number rather than another expression.
- Generates the same words as the previous grammar but the parse tree for each derivation is unique.

Unambiguous Grammars

Binary Expressions

- Change the first two productions

1. $E \rightarrow B + E$
2. $E \rightarrow B - E$
3. $E \rightarrow B$
4. $B \rightarrow 0$
5. $B \rightarrow D$
6. $D \rightarrow 1$
7. $D \rightarrow D0$
8. $D \rightarrow D1$



- The expression gets longer by adding more operators and digits (i.e. expressions) on the right hand side.

CFGs and Derivations

Formal Definition

- $\alpha A\beta$ *directly derives* $\alpha\gamma\beta$ if there *is a production* (a.k.a. a production rule) $A \rightarrow \gamma$ where
  - $A \in N$ (non-terminals) and
  - $\alpha, \beta, \gamma \in (N \cup T)^*$ (non-terminals, terminals, empty string)
- e.g. E-B+E $\Rightarrow$ E-D+E uses rule 5: $B \rightarrow E$
  $\alpha A\beta \Rightarrow \alpha\gamma\beta$ $\quad A \rightarrow \gamma$
- e.g. E-D+E $\Rightarrow$ E-D0+E uses rule 7: $D \rightarrow D0$
  $\alpha\ A\ \beta \Rightarrow \alpha\ \gamma\ \beta$ $\quad A \rightarrow \gamma$
- e.g. 10+D $\Rightarrow$ 10+1 uses rule 6: $D \rightarrow 1$
  $\alpha\ A\beta \Rightarrow \alpha\ \gamma\beta$ $\quad A \rightarrow \gamma$, i.e. here $\beta = \varepsilon$
- Informally, *directly derives* means it takes one derivation step or one application of a production rule.

CFGs and Derivations

Formal Definition

- $\alpha A\beta$ *derives* $\alpha\gamma\beta$ if there *is a finite sequence of productions* $\alpha A\beta \Rightarrow \alpha\Theta_1\beta \Rightarrow \alpha\Theta_2\beta \Rightarrow \ldots \Rightarrow \alpha\gamma\beta$
  - again $A \in$ N and $\alpha, \beta, \gamma \in (N \cup T)^*$
  - written as $\alpha A\beta \Rightarrow^* \alpha\gamma\beta$
- e.g. with $E \overset{(1)}{\Rightarrow} E + E \overset{(3)}{\Rightarrow} B + E \overset{(5)}{\Rightarrow} D + E \overset{(7)}{\Rightarrow} D0 + E \overset{(6)}{\Rightarrow}$

  $$10 + E \overset{(3)}{\Rightarrow} 10 + B \overset{(5)}{\Rightarrow} 10 + D \overset{(6)}{\Rightarrow} 10 + 1$$

  - $E \Rightarrow^* D0 + E$ w/ productions: 1, 3, 5, 7
  - $E \Rightarrow^* 10 + B$ w/ productions: 1, 3, 5, 7, 6, 3,
  - $E \Rightarrow^* 10 + 1$ w/ productions: 1, 3, 5, 7, 6, 3, 5, 6
- Informally, *derives* means it takes many derivation steps.

CFGs and Derivations

Formal Definition

- The grammar G *derives the word* $w \in T^*$ if $S \Rightarrow^* w$
  - w is a concatenation of terminals
  - S is the start symbol
- *Informally*, the grammar G *derives a word*, w, if you can derive that word from the start symbol.
  - e.g. $E \Rightarrow^* 10 + 1$ w/ productions: 1, 3, 5, 7, 6, 3, 5, 6

CFGs and Derivations

Formal Definition

- The *language* $\mathcal{L}(G)$ = $\{w \in T^* \text{ i: } S \Rightarrow^* w\}$.
- *Informally,* the *language described by the grammar G* is the set of concatenations of terminal symbols that can be derived from the start symbol.
- Given a CFG G and word *w*, you can think of $S \Rightarrow^* w$ as a proof that *w* is in the language $\mathcal{L}$(G).

CFGs and Derivations

Formal Definition

- A *language L is context-free* if there exists a context-free grammar *G, such that* $\mathcal{L}(G) = L$.
- *Informally,* a set of strings is context-free if there is some context free grammar that describes the language.
- Given $\alpha A \beta C \gamma$ where
  - $\alpha, \beta \in T^*$ i.e. a finite number of terminals
  - $A, C \in N$ i.e. a single non-terminals
  - $\gamma \in (N \cup T)^*$ i.e. a mixture of both
  - then a *leftmost derivation* must rewrite *A*.
- *Informally*, rewrite the leftmost non-terminal first.