CS310 : Automata Theory 2019

Lecture 15: Parse tree

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2019-02-05

Words extending at many places

In regular languages, words are extended at the end depending on the finite information collected on the word so far.

In CFLs, words are extended at unboundedly many points, which gives CFLs more power.

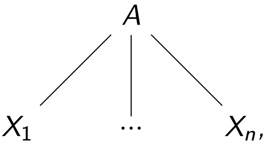
To understand the above intuition, we view the words in derivations as tree.

Parse tree

Definition 15.1

For a grammar $G = (N, T, P, S)$, a parse tree for G is a labelled tree with the following conditions

- *leaf label is $X \in N \cup T \cup \{\epsilon\}$. If $X = \epsilon$, the leaf has no siblings.*
- *internal node label is $A \in N$*
- *If an internal node label is $A \in N$ and its children labels are*

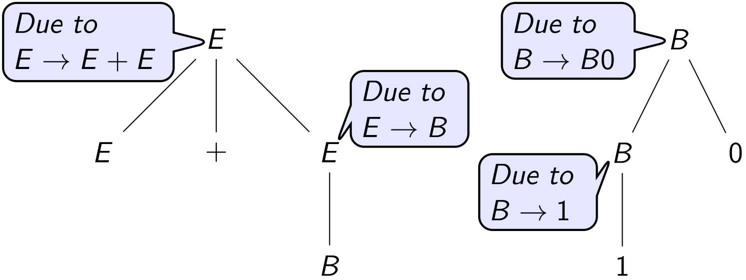


then $A \rightarrow X_1 ... X_n \in P$.

Example : parse tree

Example 15.1

The following are parse trees for G_{arith}.



- *Parse tree need not be fully expanded, i.e., may be nonterminal leafs*
- *Parse tree root need not be the start symbol*

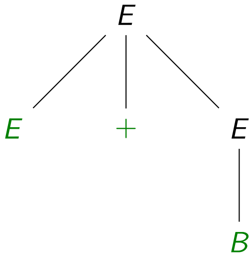
Yield of a parse tree

Definition 15.2

The yield *of a parse tree is the word formed by all the leaves from left to right*

Example 15.2

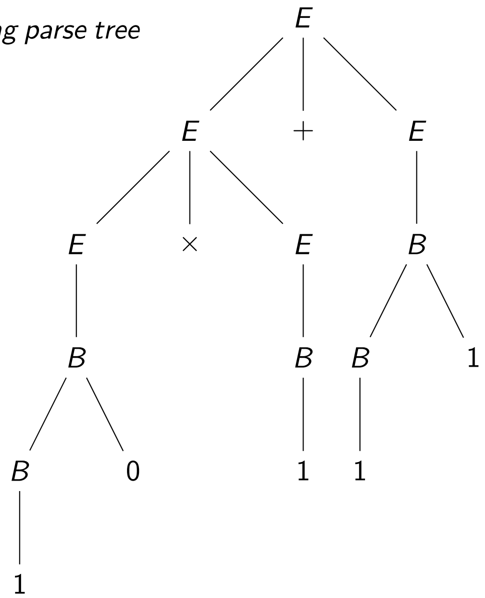
The yield of the following parse tree is $E + B$



Example : parse tree full

Example 15.3

The yield of the following parse tree is $10 \times 1 + 11$.



Projected derivations lemma

Theorem 15.1

Let $G = (N, T, P, S)$ be a CFG. Let $X_i \in (N \cup T)$.
If $\underbrace{X_1....X_k \stackrel{*}{\Rightarrow} \alpha}_{n \text{ steps}}$, then $\alpha = \alpha_1 \dots \alpha_k$ such that $\underbrace{X_i \stackrel{*}{\Rightarrow} \alpha_i}_{\leq n \text{ steps}}$ for each $i \in 1..k$.

Proof.

We argue this via induction on the number of derivations.

base case:

After zero derivations, $\alpha_i = X_i$.

induction step:

Let $X_1....X_k \stackrel{*}{\Rightarrow} \alpha_1...\alpha_k$ in n steps. Due to induction hypothesis, $\underbrace{X_i \stackrel{*}{\Rightarrow} \alpha_i}_{\leq n \text{ steps}}$.

In the next derivation, let α_ℓ be expanded to α'_ℓ (no other α_i will change).

Therefore, $X_1....X_k \stackrel{*}{\Rightarrow} \alpha_1...\alpha_{\ell-1}\alpha'_\ell\alpha_{\ell+1}..\alpha_k$

Therefore, $\underbrace{X_\ell \stackrel{*}{\Rightarrow} \alpha'_\ell}_{\leq n+1 \text{ steps}}$. For $i \neq \ell$, the goal trivially holds.(why?) □

Example : projection derivation

Example 15.4

Consider the following three derivations

$$E + E \times E \Rightarrow E + E \times B \Rightarrow E + E \times E \times B \Rightarrow B + E \times E \times B$$

The following are the projected derivations for each symbol in the initial word.

- $E \Rightarrow E \Rightarrow E \Rightarrow B$
- $+ \Rightarrow + \Rightarrow + \Rightarrow +$
- $E \Rightarrow E \Rightarrow E \times E \Rightarrow E \times E$

After removing redundant steps

- $E \Rightarrow B$
- $+$
- $E \Rightarrow E \times E$

Exercise 15.1

Give projected derivations of symbols E *and* $\times$ *from the initial word*

Derivations $\Rightarrow$ parse tree

Theorem 15.2

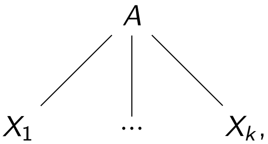
Let $G = (N, T, P, S)$ be a CFG. Let $\alpha \in (N \cup T)^*$ and $A \in N$.
If $A \stackrel{*}{\Rightarrow} \alpha$, then there is a parse tree with root A and the yield of the tree is α.

Proof.

We will use induction on the length of derivations.

base case:

Consider a single step derivation. $A \Rightarrow X_1 ... X_k$ due to production rule $A \rightarrow X_1 ... X_k$. By the definition of the parse tree, the following is a parse tree.



...

Exercise 15.2

Why is base case chosen to be one but not zero derivation?

Derivations $\Rightarrow$ parse tree II

Proof(contd.).

induction step:

Consider derivation $A \Rightarrow \underbrace{X_1....X_k \stackrel{*}{\Rightarrow} \alpha}_{n \text{ steps}}$

Due to projected derivation lemma,

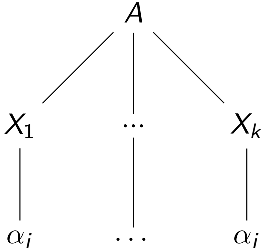
$\alpha = \alpha_1 \dots \alpha_k$ such that $\underbrace{X_i \stackrel{*}{\Rightarrow} \alpha_i}_{\leq n \text{ steps}}$ for each $i \in 1..k$. ...

Derivations $\Rightarrow$ parse tree II

Proof(contd.).

Due to induction hypothesis for each $i \in 1..k$, there is a parse tree with X_i root and it yields α_i.

We can construct a parse tree for derivation $A \overset{*}{\Rightarrow} \alpha$ as follows



□

Parse tree $\Rightarrow$ derivation

Theorem 15.3

Let $G = (N, T, P, S)$ be a CFG. Let $\alpha \in (N \cup T)^*$ and $A \in N$. If there is a parse tree with root A and the yield of the tree is α, then $A \stackrel{*}{\Rightarrow} \alpha$.

Proof.

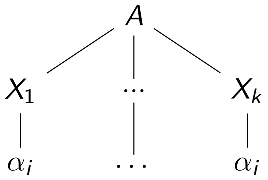
We will prove again by induction over height of the parse tree.

base case:

For the height zero, $A = \alpha$. Trivially, $A \stackrel{*}{\Rightarrow} A$.

induction step:

Root of the tree is A and its children $X_1, ..., X_k$. Therefore, $A \Rightarrow X_1...X_k$.
X_i is root of a parse tree with yield α_i such that $\alpha = \alpha_1...\alpha_k$.



Parse tree $\Rightarrow$ derivation

Proof(contd.).

Due to the induction hypothesis for each $i \in 1..k$, $X_i \stackrel{*}{\Rightarrow} \alpha_i$, which we can embed in the following derivation.

$$\alpha_1 ... \alpha_{i-1} X_i X_{i+1} ... X_k \stackrel{*}{\Rightarrow} \alpha_1 ... \alpha_{i-1} \alpha_i X_{i+1} ... X_k$$

By stitching the above(how?), we obtain $A \Rightarrow X_1 .... X_k \stackrel{*}{\Rightarrow} \alpha_1 ... \alpha_k$. $\square$

Exercise 15.3

Let $\alpha \in T^*$ *in the above proof. Prove that there exists* $A \xrightarrow{lm*} \alpha$.

Exercise 15.4

Let $\alpha \in T^*$ *in the above proof. Prove that there exists* $A \xrightarrow{rm*} \alpha$.

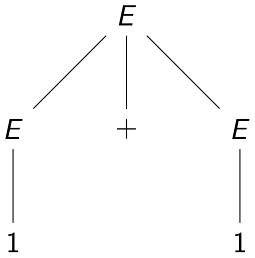
Parse tree to interpretation

Looks like parse trees are doing the same job as derivations

Actually, they fully record the "understanding" of the word under a grammar.

Example 15.5

Consider the grammar of linear expressions. We may be interested in interpreting $1 + 1$ *as sum of two 1s.*



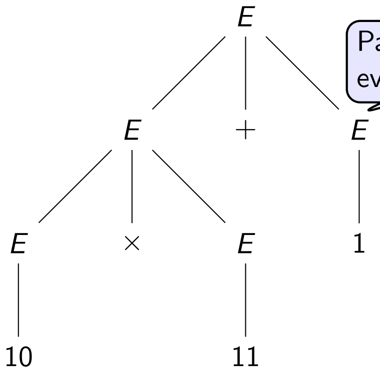
For brevity, we show no expansions due to B

In the parse tree we have the information. We can apply the addition.

Example: parse tree to interpretation

Example 15.6

The following is a parse tree for word $10 \times 11 + 1$.



Parse tree are used to evaluate the expressions

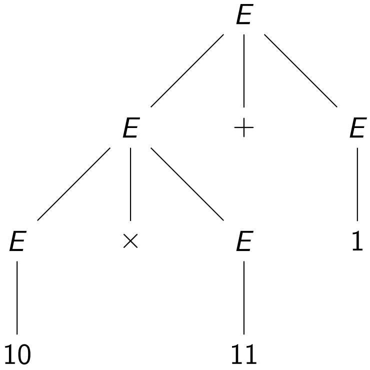
The parse tree tells us first multiply 10 and 11, which is 110.
Afterwards add 1, which results in 111.

Ambiguity

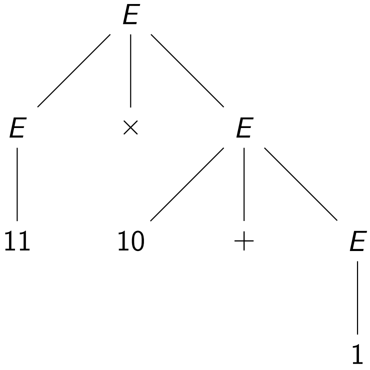
Sometimes a word may have more than one parse trees.

Example 15.7

In G_{arith}, $10 \times 11 + 1$ has following two parse trees.



Multiply first



Add first

Ambiguity and interpretation

Definition 15.3

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.

Not good for building compilers.

Example 15.8

$10 \times 11 + 1$ *can have the following two interpretations*

- 111 *(binary 7)* *multiply first*
- 101 *(binary 5)* *add first*

Ambiguity and derivations

Having multiple derivations does not imply ambiguity.

Example 15.9

Consider the following derivations

- $E \Rightarrow E + E \Rightarrow E + B \Rightarrow E + 1 \Rightarrow B + 1 \Rightarrow 1 + 1$
- $E \Rightarrow E + E \Rightarrow B + E \Rightarrow 1 + E \Rightarrow 1 + B \Rightarrow 1 + 1$

Both the derivations result in same parse tree.

Exercise 15.5

Give another derivation of $1 + 1$*?*

Causes of ambiguity

There are two kinds of choices in derivations

1. Order of expansions
2. Choice of production rules

The order does not cause ambiguity.

The choice of production rules to expand a symbol causes the ambiguity.

Leftmost derivations and ambiguity

The leftmost derivations eliminates the order issues.

Theorem 15.4

A grammar is ambiguous iff there are multiple leftmost derivations.

Proof.

($\Leftarrow$)

If we have two different leftmost derivations, there must be a leftmost symbol in an intermediate word that was expanded two different ways. Due to the translation in theorem 15.2 to parse trees, we will have a path in the two parse trees that lead to two different symbols.(why?)

($\Rightarrow$)

Theorem 15.3 presented construction of leftmost derivations from parse trees. Two different parse trees will lead to two different leftmost derivations. $\square$

Exercise 15.6

Formally write *(why?)*

Removing ambiguity

Can we remove ambiguity from a grammar?

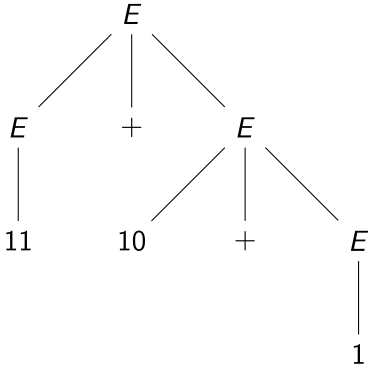
The problem is **impossible** to solve.

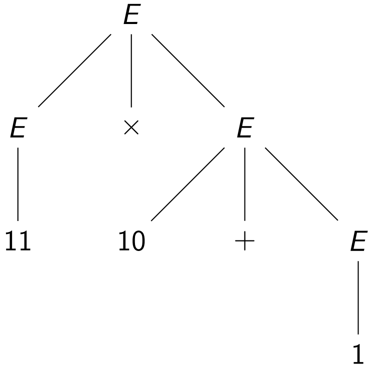
In some cases, we can remove ambiguity with specialized observations.

Removing ambiguity in arithmetic expressions

There are two sources of ambiguity in our G_{arith}.

1. Associativity of $\times$ and $+$
2. Precedence between $\times$ and $+$





In the grammar we need to say what to process first.

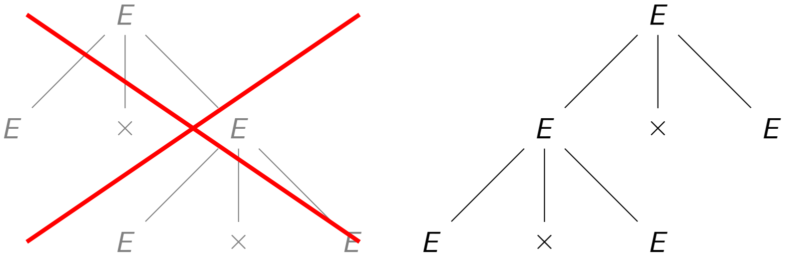
Making operators left associative

Consider multiplication operation first.

Let us suppose we have multiple multiplications in a row

$$E \times E \times E$$

We need to somehow disqualify one of the following parse tree.



We can resolve the ambiguity by giving preference to left most multiplication, which is called left associative operation.

Production rules for left associative multiplication

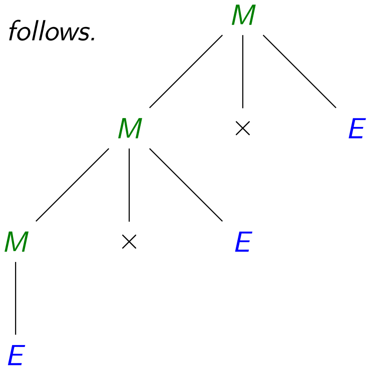
The following productions rules make $\times$ left associative.

$$M \rightarrow E \mid M \times E$$

M is multiplications of expressions that can only be extended from right.

Example 15.10

$E \times E \times E$ is parsed as follows.



Exercise 15.7

Give production rules that make $\times$ right associative.

Production rules for left associative sum of products

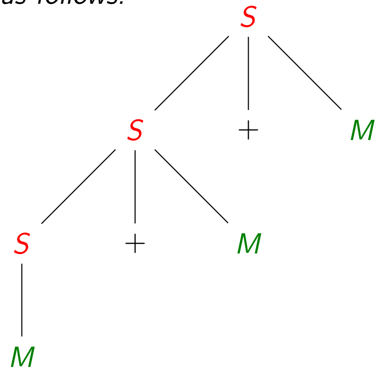
The following productions rules make $+$ left associative.

$$S \to M \mid S + M$$

Only sums of multiplications are allowed
(Therefore, multiplication of sums are disallowed).

Example 15.11

$M + M + M$ is parsed as follows.



Unambiguous arithmetic expressions

Putting it all together

$G'_{arith} = (\{B, M, S, E\}, \{+, \times, 0, 1, (,)\}, P, S)$, where P consists of

$$
\begin{aligned}
B &\rightarrow 1 \mid B0 \mid B1 \\
M &\rightarrow E \mid M \times E \\
S &\rightarrow M \mid S + M \\
E &\rightarrow B \mid (S).
\end{aligned}
$$

E ties all back, which is either a binary number or parenthesized expression, which are unambiguous subwords.

Inherently ambiguous languages

There are CFLs that have only ambiguous grammars

For example,

$$\{a^n b^m c^\ell d^k | (n = m \land \ell = k) \lor (n = k \land m = \ell)\}$$

Hard to prove and we will not cover this in the course!

Our first encounter with impossible problems and very hard proofs!

End of Lecture 15