Session 16

Ambiguity

Ambiguity

- If there is more than one structure with the same root and the same yield generated by a grammar then the grammar is ambiguous
- Two kinds of ambiguity
  - In the grammar
  - In the language
- If a grammar is ambiguous there might be an unambiguous grammar for the same language
- A language is inherently ambiguous if all its grammar are ambiguous
- There is no algorithm to tell whether a grammar is ambiguous

An ambiguous grammar

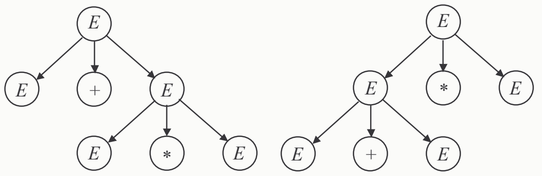
- *Exp* is a CFG
  - $G_{exp} = (\{E\}, \{+, *, (,), 1, \ldots, 9\}, E, P)$ where $P = \{E \rightarrow E + E \mid E * E \mid (E) \mid 1 \mid \ldots \mid 9\}$
- An ambiguous expressions:
  - $E + E * E$
- Two derivations:
  - $E \Rightarrow E + E \Rightarrow E + E * E$
  - $E \Rightarrow E * E \Rightarrow E + E * E$
- They look the same!

An ambiguous grammar

- The final expression looks the same:
  - $E \Rightarrow^{*} E + E * E$
  - $E \Rightarrow^{*} E + E * E$
- But the derivations are different:
  - $E \Rightarrow E + E \Rightarrow E + E * E$
  - $E \Rightarrow E * E \Rightarrow E + E * E$

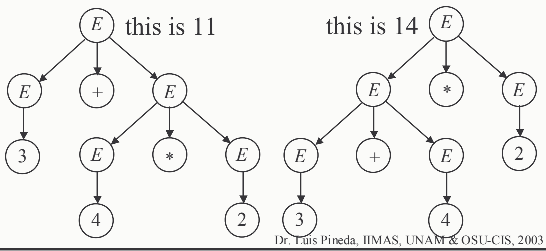
An ambiguous grammar

- They final expression looks the same:
  - $E \Rightarrow^{*} E + E * E$
  - $E \Rightarrow^{*} E + E * E$
- The corresponding syntactic structures are also different!



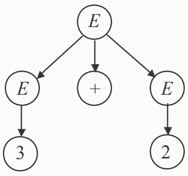
An ambiguous grammar

- The difference is significant
  - $E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow^{*} 3 + 4 * 2$
  - $E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow^{*} 3 + 4 * 2$



The defect may not be in the grammar

- But different derivations may have the same structure
  - $E \Rightarrow E + E \Rightarrow 3 + E \Rightarrow 3 + 2$
  - $E \Rightarrow E + E \Rightarrow E + 2 \Rightarrow 3 + 2$



- Ambiguity arises when there is more than one structure for the same expression!

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

An ambiguous grammar

- A CFG $G = (V, \Sigma, S, P)$ is *ambiguous* if there is at least one string w in Σ^* for which there is more than one parse tree or syntactic structure, each with root S and yield w
- If every string in the grammar has at most one parse tree, the grammar is *unambiguous*
- If G is an ambiguous CFG such that $L = L(G)$, and there is an unambiguous G_i such that $L = L(G_i)$, we can remove the ambiguity by replacing G by G_i

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Removing ambiguity

- In general, there is no algorithm for removing ambiguity
- There are CFG that have only ambiguous grammars!
- In practice, and for some applications like defining CFG for programming languages, we can remove the ambiguity
- For this, we need to study the causes for the ambiguity of the grammar under study, and then provide a particular solution!

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

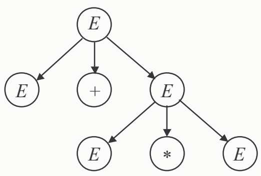
Removing ambiguity

- Consider the language G_{exp} again: where the ambiguity comes from?
- Source 1:
  - Precedence of operators is not respected!
- Source 2:
  - A sequence of identical operators can be grouped either from left to right or from right to left
  - This does not matter if operators are associative

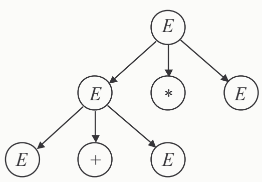
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

An ambiguous grammar

- Precedence of operators is not respected :
  - $E \Rightarrow^* E + E * E$
  - $E \Rightarrow^* E + E * E$



"*" has higher precedence

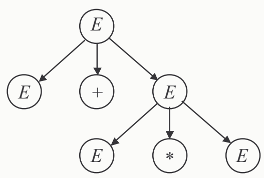


"+" has higher precedence

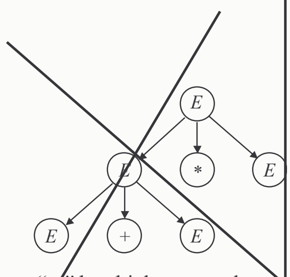
Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

An ambiguous grammar

- Reinforcing precedence: we are left with just one tree:
  - $E \Rightarrow^* E + E * E$



"*" has higher precedence

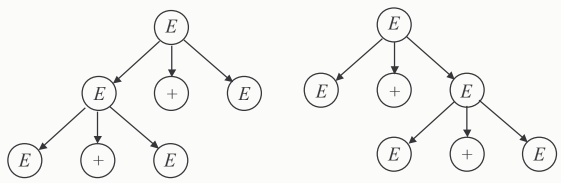


"+" has higher precedence

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

An ambiguous grammar

- Arbitrary groupings of operators with equal precedence:
  – $E \Rightarrow^* E + E + E$

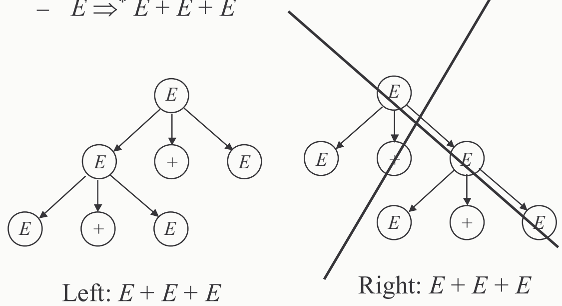


Left: $E + E + E$ Right: $E + E + E$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

An ambiguous grammar

- Adopting a grouping convention (e.g. by the left)
  – $E \Rightarrow^* E + E + E$



Left: $E + E + E$ Right: $E + E + E$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Removing ambiguity

- Enforcing precedence
  – Introduce different kinds of variables representing expressions that share a level of "binding strength": factors, terms and expressions
- Extending G_{exp} with identifiers:
  – $G_{exp} = (\{E, I\}, \{+, *, (,), a, b, 0, 1\}, E, P)$
  where $P = \{E \rightarrow E + E \mid E * E \mid (E) \mid I,$
  $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1\}$
- What are the factors, terms and expressions in G_{exp}?

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Binding strength

- A *factor* (in G_{exp})
  – An expression that cannot be broken apart by any adjacent operator:
    a) Identifiers: $a1+b00*aa10 \Rightarrow (a1+b0)(0*aa10)$?
    b) Expression between parenthesis: parenthesis make expressions within them coherent units! They are syntactic devices for creating factors!
- A *term* (in G_{exp})
  – An expression that cannot be broken by a + operator:
    $a1*a*b \Rightarrow (a1*a)*b$ is ok. (assoc. by the left)
    $a1+a*b \Rightarrow (a1+a)*b$? $a*b$ is a term!
- An *expression* (G_{exp})
  – Any well-formed string that *can* be broken either by an adjacent + or a *: an expression is the sum of two terms

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Removing ambiguity

- Enforcing precedence
  – With these precedence variables we can find an alternative *unambiguous* grammar for G_{exp} (i.e. one that generates the same language):
  $G_{exp\text{-}1} = (\{E, T, F, I\}, \{+, *, (,), a, b, 0, 1\}, E, P)$
  where $P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \rightarrow I \mid (E)$
  $T \rightarrow F \mid T * F$
  $E \rightarrow T \mid E + T\}$
- The rules are designed in a way that variables with lower binding strength dominate variables with a higher binding strength.

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Generate: $a + a * a$

- The productions:
  $P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \rightarrow I \mid (E)$
  $T \rightarrow F \mid (T * F)$
  $E \rightarrow T \mid E + T\}$

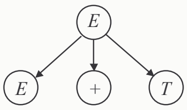
- The derivation
  $E \Rightarrow E + T$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Generate: $a + a * a$

- The productions:

$$P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1, \quad F \rightarrow I \mid (E) \quad T \rightarrow F \mid (T * F) \quad E \rightarrow T \mid E + T\}$$



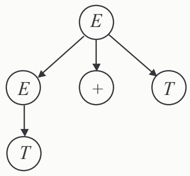
- The derivation

$E \Rightarrow E + T$

Generate: $a + a * a$

- The productions:

$$P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1, \quad F \rightarrow I \mid (E) \quad T \rightarrow F \mid (T * F) \quad E \rightarrow T \mid E + T\}$$



- The derivation

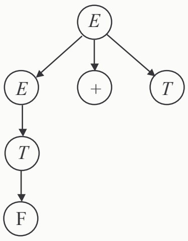
$E \Rightarrow E + T$

$\Rightarrow T + T$

Generate: $a + a * a$

- The productions:

$$P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1, \quad F \rightarrow I \mid (E) \quad T \rightarrow F \mid (T * F) \quad E \rightarrow T \mid E + T\}$$



- The derivation

$E \Rightarrow E + T$

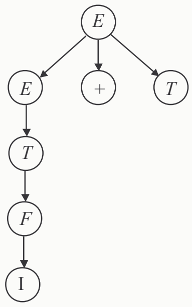
$\Rightarrow T + T$

$\Rightarrow F + T$

Generate: $a + a * a$

- The productions:

$$P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1, \quad F \rightarrow I \mid (E) \quad T \rightarrow F \mid (T * F) \quad E \rightarrow T \mid E + T\}$$



- The derivation

$E \Rightarrow E + T$

$\Rightarrow T + T$

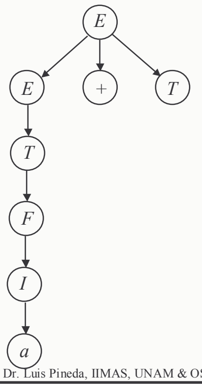
$\Rightarrow F + T$

$\Rightarrow I + T$

Generate: $a + a * a$

- The productions:

$$P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1, \quad F \rightarrow I \mid (E) \quad T \rightarrow F \mid (T * F) \quad E \rightarrow T \mid E + T\}$$



- The derivation

$E \Rightarrow E + T$

$\Rightarrow T + T$

$\Rightarrow F + T$

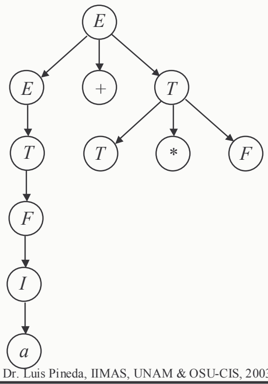
$\Rightarrow I + T$

$\Rightarrow a + T$

Generate: $a + a * a$

- The productions:

$$P = \{I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1, \quad F \rightarrow I \mid (E) \quad T \rightarrow F \mid (T * F) \quad E \rightarrow T \mid E + T\}$$



- The derivation

$E \Rightarrow E + T$

$\Rightarrow T + T$

$\Rightarrow F + T$

$\Rightarrow I + T$

$\Rightarrow a + T$

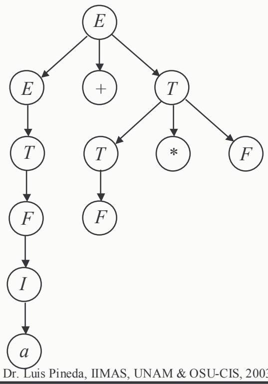
$\Rightarrow a + (T * F)$

Generate: $a + a * a$

- The productions:

  $P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \to I \mid (E)$
  $T \to F \mid (T * F)$
  $E \to T \mid E + T\ \}$
- The derivation

  $E \Rightarrow^* a + (F * F)$

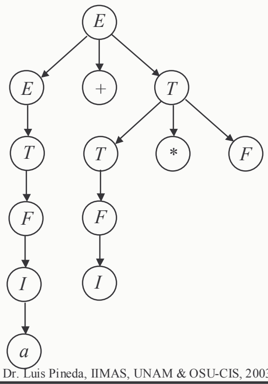


Generate: $a + a * a$

- The productions:

  $P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \to I \mid (E)$
  $T \to F \mid (T * F)$
  $E \to T \mid E + T\ \}$
- The derivation

  $E \Rightarrow a + (F * F)$
  $\Rightarrow a + (I * F)$

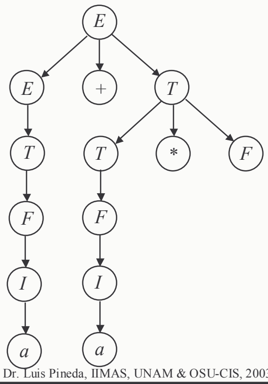


Generate: $a + a * a$

- The productions:

  $P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \to I \mid (E)$
  $T \to F \mid (T * F)$
  $E \to T \mid E + T\ \}$
- The derivation

  $E \Rightarrow a + (F * F)$
  $\Rightarrow a + (I * F)$
  $\Rightarrow a + (a * F)$

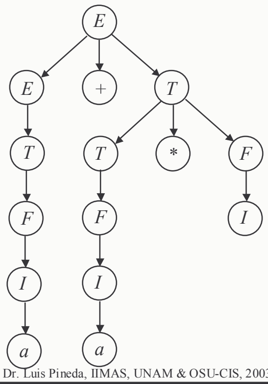


Generate: $a + a * a$

- The productions:

  $P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \to I \mid (E)$
  $T \to F \mid (T * F)$
  $E \to T \mid E + T\ \}$
- The derivation

  $E \Rightarrow a + (F * F)$
  $\Rightarrow a + (I * F)$
  $\Rightarrow a + (a * F)$
  $\Rightarrow a + (a * I)$

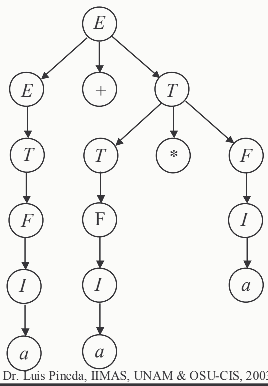


Generate: $a + a * a$

- The productions:

  $P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \to I \mid (E)$
  $T \to F \mid (T * F)$
  $E \to T \mid E + T\ \}$
- The derivation

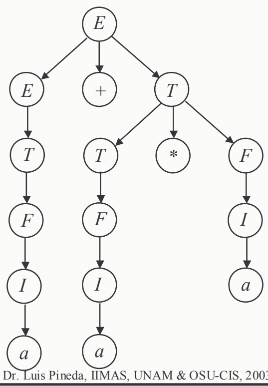
  $E \Rightarrow a + (F * F)$
  $\Rightarrow a + (I * F)$
  $\Rightarrow a + (a * F)$
  $\Rightarrow a + (a * I)$
  $\Rightarrow a + (a * a)$



Generate: $a + a * a$

- The productions:

  $P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$
  $F \to I \mid (E)$
  $T \to F \mid (T * F)$
  $E \to T \mid E + T\ \}$
- The grammar is unambiguous: Variables of lower precedence are introduced before, and variables of higher precedence are units that cannot be broken by variables of lower precedence, which are already in the tree!

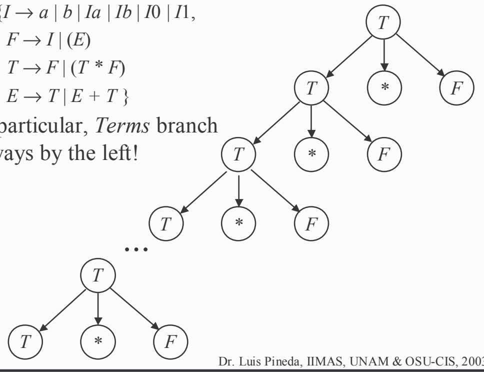


Generate: $a + a * a$

- The productions:

$$P = \{I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1,$$
$$F \to I \mid (E)$$
$$T \to F \mid (T * F)$$
$$E \to T \mid E + T\}$$

- In particular, *Terms* branch always by the left!



Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Unambiguous grammar

If a grammar is unambiguous leftmost and rightmost derivations are unique!

- Leftmost derivation:

$E \Rightarrow E + T$
$\Rightarrow T + T$
$\Rightarrow F + T$
$\Rightarrow I + T$
$\Rightarrow a + T$
$\Rightarrow a + (T * F)$
$\Rightarrow a + (F * F)$
$\Rightarrow a + (I * F)$
$\Rightarrow a + (a * F)$
$\Rightarrow a + (a * I)$
$\Rightarrow a + (a * a)$

- Rightmost derivation:

$E \Rightarrow E + T$
$\Rightarrow E + (T * F)$
$\Rightarrow E + (T * I)$
$\Rightarrow E + (T * a)$
$\Rightarrow E + (F * a)$
$\Rightarrow E + (I * a)$
$\Rightarrow E + (a * a)$
$\Rightarrow T + (a * a)$
$\Rightarrow F + (a * a)$
$\Rightarrow I + (a * a)$
$\Rightarrow a + (a * a)$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Ambiguity and leftmost derivations

- Leftmost derivation:

$E \Rightarrow E + T$
$\Rightarrow T + T$
$\Rightarrow F + T$
$\Rightarrow I + T$
$\Rightarrow a + T$
$\Rightarrow a + (T * F)$
$\Rightarrow a + (F * F)$
$\Rightarrow a + (I * F)$
$\Rightarrow a + (a * F)$
$\Rightarrow a + (a * I)$
$\Rightarrow a + (a * a)$

- Theorem: For each grammar $G = (V, T, P, S)$ and string w in T^*, w has two distinct parse trees iff w has two leftmost derivations from S
- Proof: if it were not the case, a left variable in a leftmost derivation should expand in more than one way!

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Inherent ambiguity

- A language L is said inherently ambiguous if all its grammars are ambiguous; if there is at least one unambiguous grammar for L, L is unambiguous.
  - The language of Expressions is unambiguous
  - Regular expressions are unambiguous
- An example of an inherently ambiguous language:

$$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

  - L is context free:

$S \to AB \mid C$

$A \to aAb \mid ab$ $\quad C \to aCd \mid aDd$

$B \to cCd \mid cd$ $\quad D \to bCc \mid bc$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Inherent ambiguity

- L is context free:

$S \to AB \mid C$

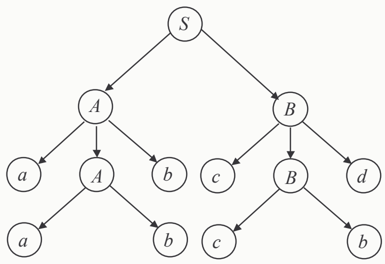
$A \to aAb \mid ab$ $\quad C \to aCd \mid aDd$

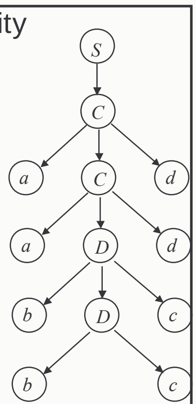
$B \to cBd \mid cd$ $\quad D \to bDc \mid bc$

- The grammar is ambiguous: there are strings with more than two leftmost derivations:
  - Consider: *aabbccdd* $(m = n = 2)$
  - $S \Rightarrow AB \Rightarrow aAbB \Rightarrow aabbB \Rightarrow aabbcBd \Rightarrow aabbccdd$
  - $S \Rightarrow C \Rightarrow aCd \Rightarrow aaDdd \Rightarrow aabDcdd \Rightarrow aabbccdd$

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Inherent ambiguity





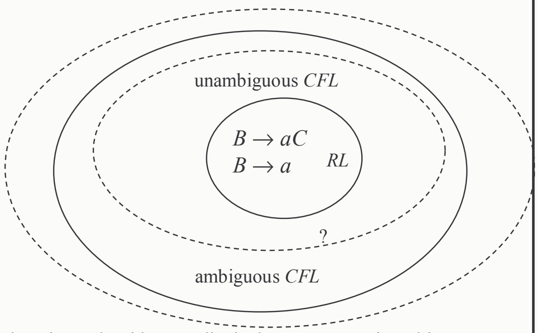
Two parse trees for *aabbccdd*

Dr. Luis Pineda, IIMAS, UNAM & OSU-CIS, 2003

Inherent ambiguity

- The language:
  - $L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$
- The grammar
  - $S \rightarrow AB \mid C \quad A \rightarrow aAb \mid ab \quad C \rightarrow aCd \mid aDd$
  - $\quad B \rightarrow cCd \mid cd \quad D \rightarrow bCc \mid bc$
- Why are all the grammars for this language ambiguous?
  - Consider any string such that $m = n$
  - There two leftmost derivations for all these strings
- What changes in the grammar can we try?
- The problem: The disjunction!
  - There is no way to avoid a mechanism to match the same number of a's and b's, and at the same time, a mechanism for matching the number of a's and d's
  - Similarly for matching c's and d's and, at the same time, b's and c's

Is there a class of ambiguous CFL



- There is no algorithm to tell whether a grammar is ambiguous
- There is no way to tell when a language is inherently ambiguous!