COMP2012/G52LAC
Languages and Computation
Lecture 11
Disambiguating Context-Free Grammars

Henrik Nilsson

University of Nottingham

Recap: Derivation Trees (1)

A tree is a ***derivation tree*** for a CFG $G = (N, T, P, S)$ iff

1. Every node has a label from $N \cup T \cup \{\epsilon\}$.
2. The label of the root node is S.
3. Labels of interior nodes belong to N.
4. If a node n has label A and nodes $n_1, n_2, \ldots, n_k$ are children of n, from left to right, with labels $X_1, X_2, \ldots X_k$, respectively, then $A \to X_1 X_2 \ldots X_k$ is a production in P.
5. If a node n has label ϵ, then n is a leaf and the only child of its parent.

Recap: Derivation Trees (2)

- The string of ***leaf labels*** read from left to right, eliding any ϵ, constitute the ***yield*** of the tree.
- For a CFG $G = (N, T, P, S)$, a string $\alpha \in (N \cup T)^*$ is the yield of some derivation tree iff $S \underset{G}{\overset{*}{\Rightarrow}} \alpha$.

Recap: Ambiguity (1)

A CFG $G = (N, T, P, S)$ is ***ambiguous*** is there is at least one word $w \in L(G)$ such that there are

- two different ***derivation trees***, or
- two different ***left-most derivations***, or
- two different ***right-most derivations***

for w.

Recap: Ambiguity (2)

Ambiguity can be problematic for a number of reasons, including that the structure of a derivation tree often is used to suggest a ***meaning*** for the word.

Example: Arithmetic Expressions

Another reason is that many (especially efficient) parsing methods are not applicable if the grammar is ambiguous.

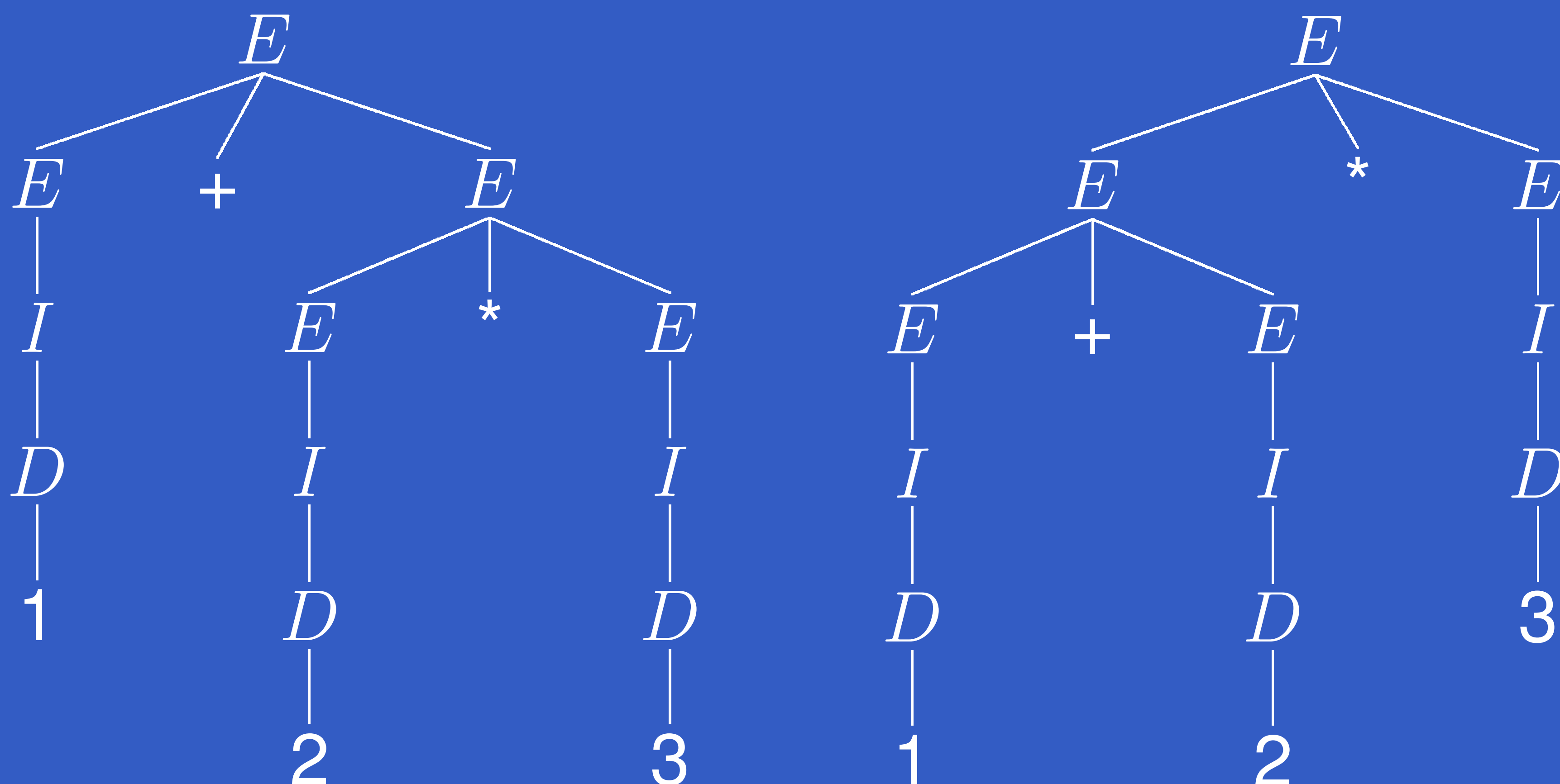
Recap: Ambiguity (3)

$SAE = (N = \{E, I, D\}, T = \{+, *, (,), 0, 1, \ldots 9\}, P, E)$
where P is given by:

$$
\begin{array}{rcl}
E & \rightarrow & E + E \\
 & | & E * E \\
 & | & (E) \\
 & | & I \\
I & \rightarrow & DI \mid D \\
D & \rightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

Recap: Ambiguity (4)

Consider: $1 + 2 * 3$. Two derivation trees:



Disambiguating Grammars

Given an ambiguous grammar G, it is often possible to construct an ***equivalent*** grammar G' (i.e., $L(G) = L(G')$), such that G' is ***not*** ambiguous.

Disambiguating Grammars

Given an ambiguous grammar G, it is often possible to construct an ***equivalent*** grammar G' (i.e., $L(G) = L(G')$), such that G' is ***not*** ambiguous.

Some languages are ***inherently ambiguous*** CFLs, meaning that every CFG generating the language necessarily is ambiguous.

Disambiguating Grammars

Given an ambiguous grammar G, it is often possible to construct an ***equivalent*** grammar G' (i.e., $L(G) = L(G')$), such that G' is ***not*** ambiguous.

Some languages are ***inherently ambiguous*** CFLs, meaning that every CFG generating the language necessarily is ambiguous.

We will consider exploiting

- Operator Precedence
- Associativity

to disambiguate expression grammars as an example.