COMS3261: Computer Science Theory

Fall 2013

Mihalis Yannakakis

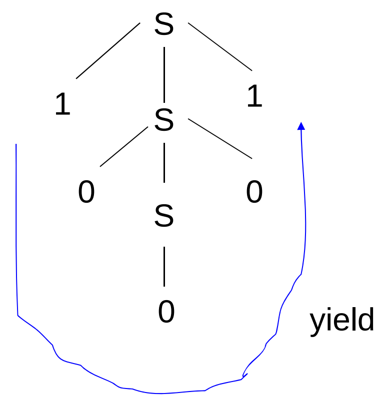
Lecture 10, 10/7/13

Parse tree (Derivation tree)

- Rooted ordered tree: has root, children of every node are ordered
- Internal nodes labeled with variables
- Leaves labeled with variables, terminals or ε
- If internal node labeled A and children $X_1,..X_k$ in left-to-right order then production $A \rightarrow X_1..X_k \in P$
- Yield of parse tree=string of labels of leaves from left to right
- Complete parse tree: all leaves labeled by terminals or ε and root labeled by S (start symbol) – we'll see the set of yields of such trees = L(G)

- Parse tree shows structure of a string, eg. structure of a sentence in a language, of an expression (eg. an arithmetic expression), of a program

Parse tree example

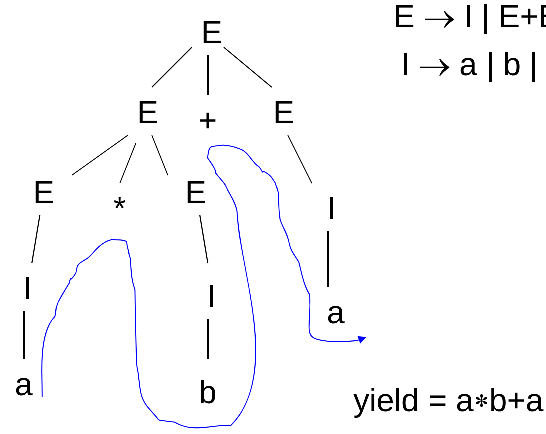


$S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$

Parse tree example

$E \rightarrow I \mid E{+}E \mid E{*}E \mid (E)$

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$



same parse tree for leftmost and rightmost derivation

Recursive Inference

- Recursive inference: use productions from body to head to deduce that string is of type S
- If $A \to w \in P$ then terminal string w is of type A
- If $A \to X_1X_2 \ldots X_k \in P$ and w_1 is either $=X_1$ if X_1 is a terminal or is of type X_1 if X_1 is a variable, …,w_k is equal to X_k or of type X_k then $w_1w_2 \ldots w_k$ is of type A

Example:

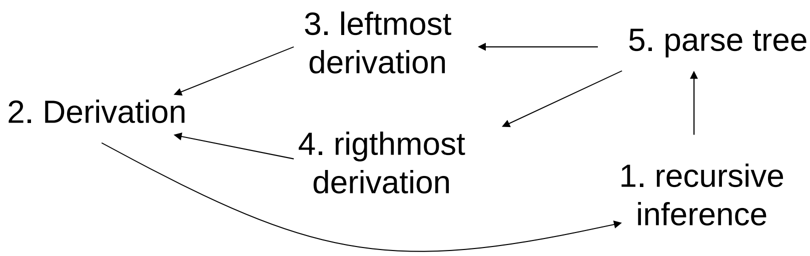
- 0 palindrome (by rule $S \to 0$)
- 000 palindrome (by rule $S \to 0S0$)
- 10001 palindrome (by rule $S \to 1S1$)

Derivations $\equiv$ Parse trees $\equiv$ Rec. Inference

Theorem: Let G = (V,T,P,S) be a cfg and $A \in V$. The following are equivalent for any string w in T*.

1. Recursive inference implies that w is in the language with start symbol A.
2. $A \Rightarrow^* w$ (by any derivation)
3. $A \Rightarrow^*_{lm} w$ (by a leftmost derivation)
4. $A \Rightarrow^*_{rm} w$ (by a rightmost derivation)
5. There is a parse tree with root labeled A and yield w

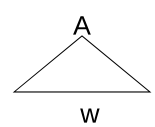
Structure of proof



- 3,4 $\Rightarrow$ 2: obvious

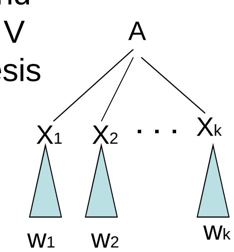
Recursive Inference $\rightarrow$ Parse tree

- If w can be inferred then there is parse tree with root labeled A and yield w
- Induction on # steps in inference
- Basis: 1 step. Production $A \rightarrow w$



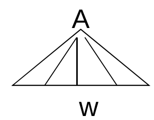
- Induction: w is inferred in n>1 steps using $A \rightarrow X_1 \ldots X_k$ where $w = w_1 \ldots w_k$ and we inferred each w_i is of type X_i if $X_i \in V$ or $w_i = X_i$ if $X_i \in T$. By induction hypothesis have parse trees for the w_i's.

  - Combine in one tree.



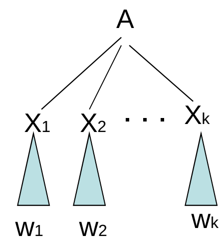
Parse Tree $\rightarrow$ LM / RM Derivation

- Given parse tree, obtain a lm and a rm derivation of yield
- Induction on height of tree.
- Basis: height 1. Tree is :



  Then production $A \rightarrow w$, so

  $A \Rightarrow_{lm} w$ and $A \Rightarrow_{rm} w$



- Induction: height >1. Tree is :

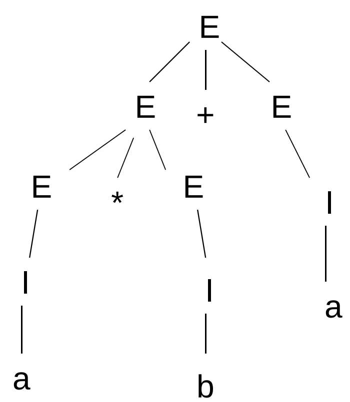
  By i.h. $X_i \Rightarrow_{lm} w_i$ or $X_i = w_i \in T$ for all i

  So $A \Rightarrow_{lm} X_1 \ldots X_k \Rightarrow^*_{lm} w_1 X_2 \ldots X_k$

  $\Rightarrow^*_{lm} w_1 w_2 \ldots X_k \ \ldots \Rightarrow^*_{lm} w_1 w_2 \ldots w_k$

Similarly for RM derivation

Example



Leftmost derivation:

$E \Rightarrow E+E \Rightarrow E*E+E \Rightarrow I*E+E \Rightarrow a*E+E \Rightarrow a*I+E \Rightarrow a*b+E \Rightarrow a*b+I \Rightarrow a*b+a$

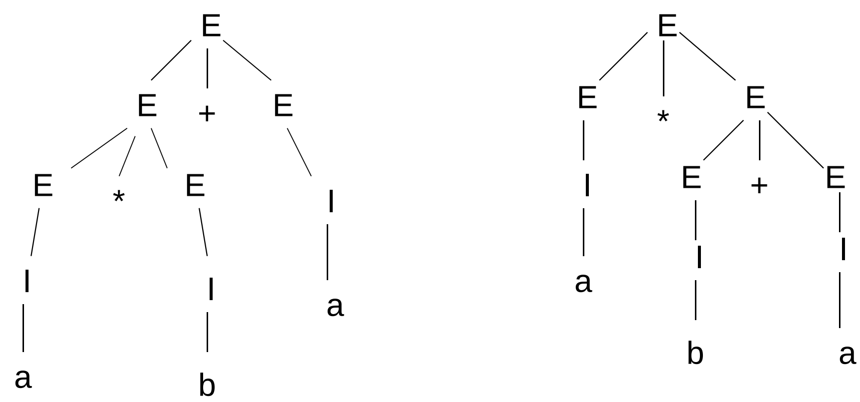
Derivation → Inference

- Suppose $A \Rightarrow^* w$. Then can infer w of type A
- Induction on length of derivation.
- Basis. Length 1. Then production $A \rightarrow w$
- Induction: Length n>1.
- Derivation is $A \Rightarrow X_1 \ldots X_k \Rightarrow^* w_1 w_2 \ldots w_k = w$, where each w_i is = X_i (if in T) or is derived from X_i in fewer than n steps.
- In the latter case can infer w_i is of type X_i by i.h.
- Follows that w is of type A

Ambiguity in Grammars

- A string can have multiple derivations that correspond to the same parse tree.
- There is 1-1 correspondence between parse trees of a terminal string, leftmost derivations and rightmost derivations
- A grammar G is ambiguous if there is a string w in L(G) that has more than one parse trees (equivalently lm or rm derivations)
- Otherwise G is unambiguous.
- Ambiguity can be a problem: different parse trees → different structure → different meaning

Example



a∗b+a: Different meaning: (a∗b)+a versus a∗(b+a)

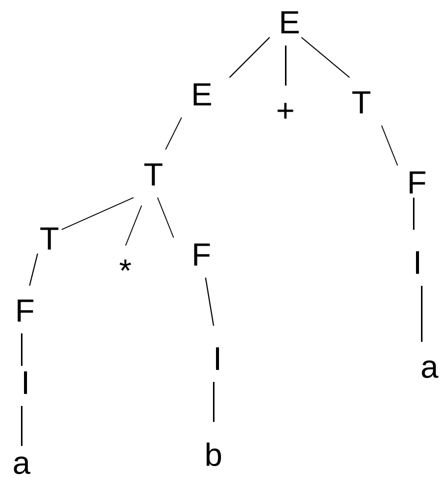
For example if a=b=2, then left=6, right=8

Removing ambiguity

- Want to remove ambiguity, if possible (not always possible)
- In this case we can by introducing more variables: T for term, F for factor, to enforce priority of ∗ over +
- $E \rightarrow T \mid E+T$ (so E is a sum of terms)
- $T \rightarrow F \mid T*F$ (so T is a product of factors)
- $F \rightarrow I \mid (E)$ (F is an identifier or a parenthesized expression)
- $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

- Generates same language.
- Only one parse tree for every string in the language

Example



Only parse tree for $a*b+a$

Inherently Ambiguous Language

- A CFL L is inherently ambiguous if all cfg's for L are ambiguous
- There are inherently ambiguous languages
- for example $\{ a^n b^n c^m d^m \mid n,m \geq 1\} \cup \{ a^n b^m c^m d^n \mid n,m \geq 1\}$
- Strings of the form $a^n b^n c^n d^n$ must have 2 parse trees

- There is no algorithm that can determine whether a given CFG is ambiguous. (Undecidable problem)
- Also whether the language of a given CFG is inherently ambiguous