Syntax Analysis

Sudakshina Dutta

IIT Goa

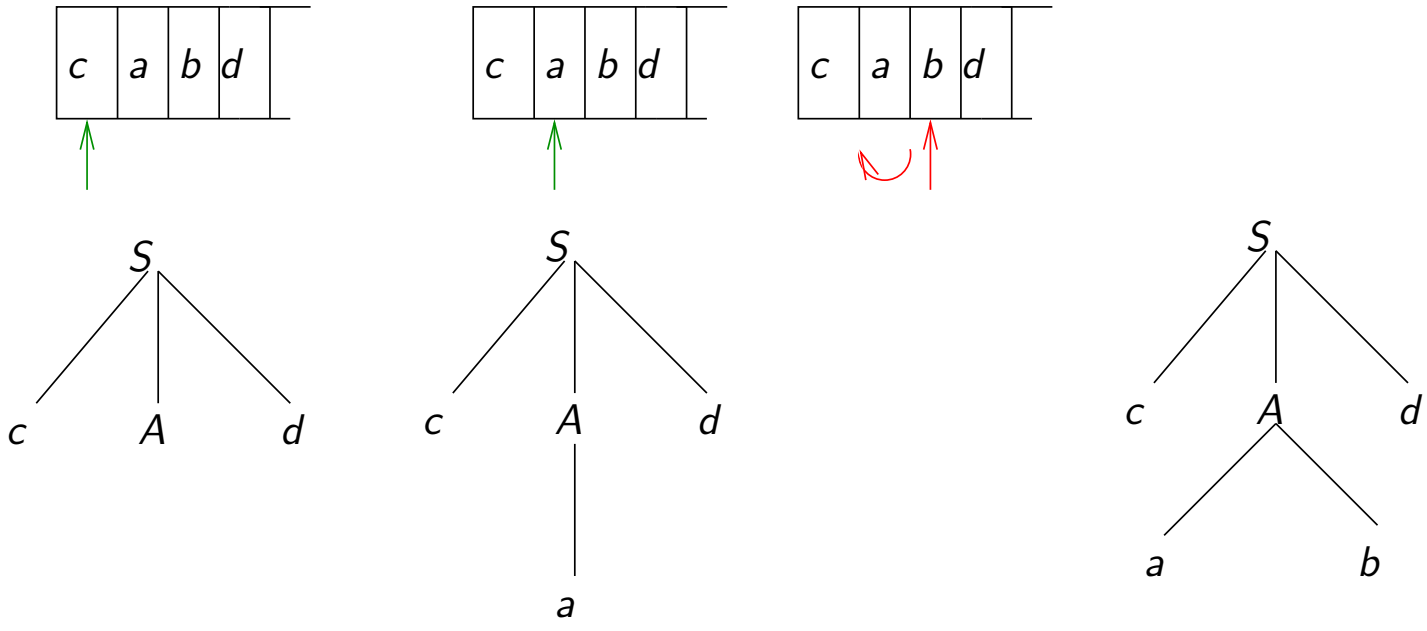
11^{th} February, 2022

Problems faced in designing a parser

Consider the grammar and a string *cabd*

- $S \rightarrow cAd$
- $A \rightarrow ab|a$

Problems faced in designing a parser



The parser might have to backtrack

1. It may have to choose new production rule
2. Rewind the input stream

Problem with backtracking

- The approach of backtracking systematically chooses the alternative for the most recently chosen production rules
- If it exhausts those alternatives, it moves back up the parse tree and reconsiders choices at a higher level in the parse tree
- If this process fails to match the input, the parser reports a syntax error

A costly approach of parsing

Ambiguity

There are two ways to resolve ambiguity

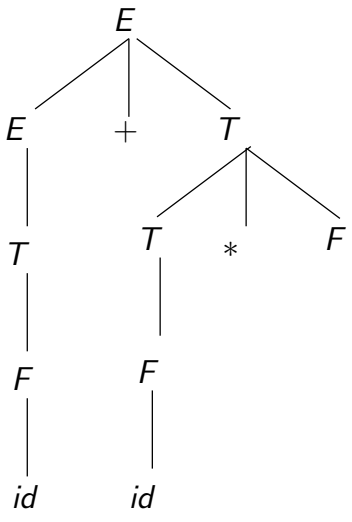
- Developer's intervention
- Other ways

Problem faced in designing parser

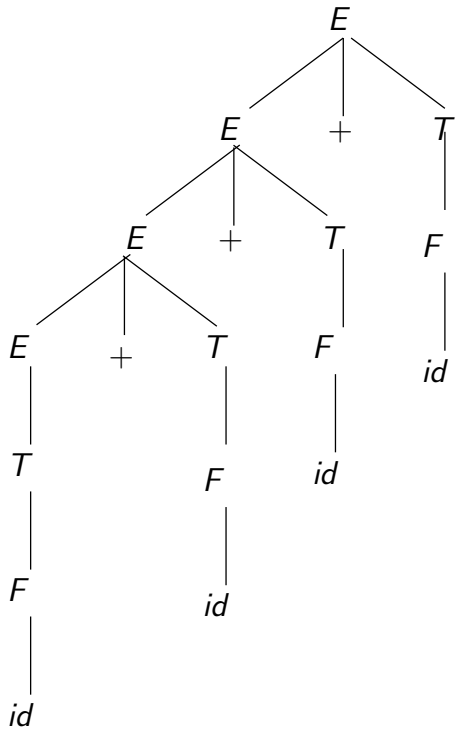
- One problem is that we are unable to choose production rules
- We apply left factoring for this

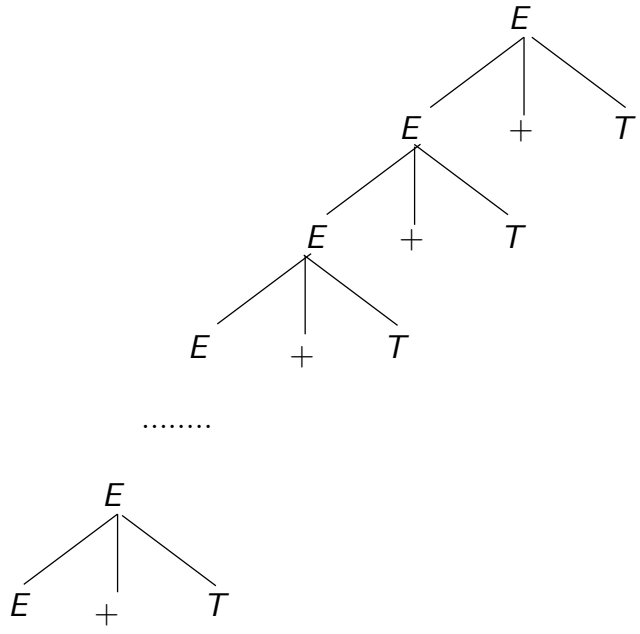
Left recursion

Parser stops as soon as it finds a mismatch of the character of input sentence with the leftmost unmatched symbol (focus)



$id + id + id\$$





$id + id + id\$$

Here once the mismatch happens with *, The compiler starts backtracking. But It does not know when to stop since it is LEFT RECURSIVE.

$id + id + id\ \$$

Had it been Right Recursive It can stop once $ symbol comes in (Intuitively.)

The compiler may continue with construction of parse trees forever

Left recursion

- Parser does not know how many times a particular rule has to be applied
- In one of the situations, it may loop forever

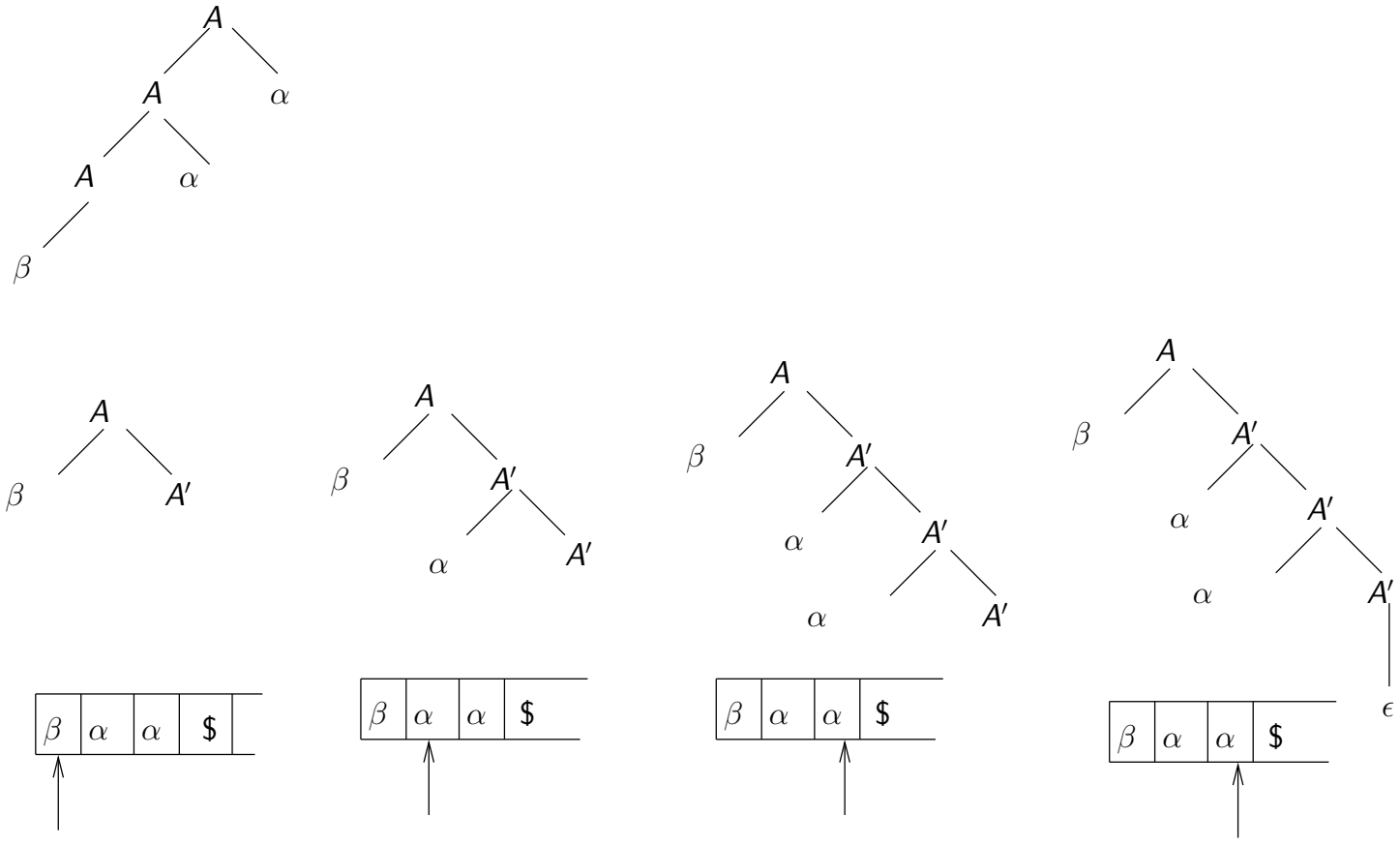
Disambiguation by Left Recursion Elimination

Suppose the production is $A \rightarrow A\alpha|\beta$

- It can be replaced with the following
  - $A \rightarrow \beta A'$
  - $A' \rightarrow \alpha A'|\epsilon$

To check whether a Grammar is suitable for Top Down parsing (Also known as Predictive Parsing or LL(1) Parsing),
We first do
- LEFT FACTORING
- LEFT RECURSION ELIMINATION.
- Now the Grammar is checked whether it is LL(1) grammar OR not. If Yes, We can say it is suitable.

Left recursion



Left recursion is converted to right recursion