Top Down Parser

Top-Down Parsing

- The parse tree is created top to bottom.
- Top-down parser
  - Recursive-Descent Parsing
  - Predictive Parsing

Difficulties with Top Down Parsing

- Left Recursion
- Backtracking
- Selection of Alternatives
- Error Reporting

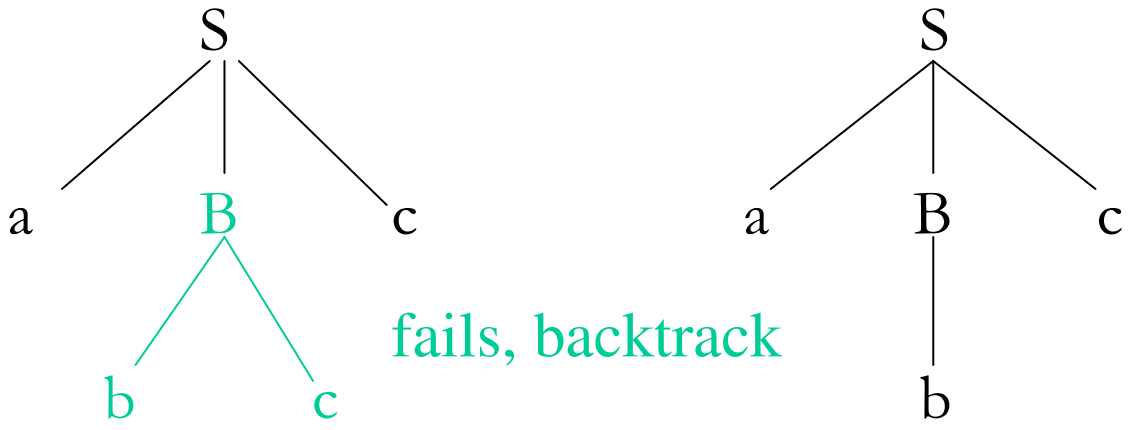
Backtracking

- Backtracking is needed.
- It tries to find the left-most derivation.

$S \rightarrow aBc$

$B \rightarrow bc \mid b$

input: abc



Top-Down Parsing

- Recursive-Descent Parsing
  - Backtracking is needed (If a choice of a production rule does not work, we backtrack to try other alternatives.)
  - It is a general parsing technique, but not widely used.
  - Not efficient
- Predictive Parsing
  - no backtracking
  - efficient
  - needs a special form of grammars (LL(1) grammars).
  - Recursive Predictive Parsing is a special form of Recursive Descent parsing without backtracking.
  - Non-Recursive (Table Driven) Predictive Parser is also known as LL(1) parser.

Recursive Descent Parser

- Uses set of recursive procedures to recognize its input with no backtracking.
- Consider the grammar:

$$E \rightarrow TE'$$
$$E' \rightarrow +TE' \mid \varepsilon$$
$$T \rightarrow FT'$$
$$T' \rightarrow *FT' \mid \varepsilon$$
$$F \rightarrow (E) \mid id$$

Recursive Descent Parser Cont…

```
Procedure E()
Begin
    T();
    EPrime();
end
```

```
Procedure EPrime()
If input symbol = '+' then
Begin
    Advance();
    T();
    EPrime();
End
```

Recursive Descent Parser Cont…

```
Procedure T()
Begin
    F();
    TPrime();
end
```

```
Procedure TPrime()
If input symbol = '*' then
Begin
    Advance();
    F();
    TPrime();
End
```

Recursive Descent Parser Cont…

```
Procedure F()
If input symbol =‘id’ then
    Advance();
Else if input symbol = ‘(‘ then
Begin
    Advance();
    E();
    if input symbol=‘)’ then
            Advance();
    else Error();
End
Else Error();
```