Parsing

Professor: Gilberto Huesca Juárez

Top-down parsing

- ■ Input:
  - □ Grammar $G=(V,\Sigma,S,P)$ (numbered rules)
  - □ String $p \in \Sigma^*$
  - □ Queue Q to perform a breadth-first search

```
1.  Put S as root of T
        Q.enqueue(S)
2.  Repeat
    2.1.    q:= Q.dequeue()                 (node to analyze)
    2.2.    i:=0                            (used rule)
    2.3.    done:=false
    Let be q=uAv where A is the leftmost variable in q.
    2.4.    Repeat
        2.4.1.  If there are no more rules with head A, then done: = true
        2.4.2.  Else, then
            Let be $A \to w$ the next rule with higher number than i and let j be the number
            of this rule.
            2.4.2.1.    If $uwv \notin \Sigma^*$ and the terminal prefix of uwv matches a prefix in p,
                    then
                2.4.2.1.1.      Q.enqueue(uwv)
                2.4.2.1.2.      Add the node uwv to T as a child of q.
        2.4.3.  i:=j
    until done or p=uwv
until Q.isEmpty() or p=uwv
3.  If p=uvw then
            Accept
        else
            Reject
```

Bottom-up parsing

- ■ Input:
    - ❑ Grammar G=(V,∑,S,P)
    - ❑ Strig p Є ∑*
    - ❑ Queue Q

```
1. Put p as root of T
      Q.enqueue(p)
2. Repeat
   q:= Q.dequeue()          (node to analyze)
   2.1.   For each rule A→w do
      2.1.1.  Por each decomposition uwv of q where vЄ∑* do
         2.1.1.1.   Q.enqueue(uAv)
         2.1.1.2.   Add the node uAv to T as child of q
   until Q.isEmpty() or q=S
3. If q=S then
            Accept
      else
            Reject
```