Nondeterministic Top-Down Parsing

Parsing

- We have seen the relationship between CFGs and syntax trees.
- Now, we have to find an efficient algorithm to obtain the syntax tree for a given input $w \in \Sigma^*$ and the grammar G
  - We have to handle ambiguity in the grammar
  - We have to decide which rule to apply for $\alpha \Rightarrow \beta$ if α contains multiple non-terminal symbols
- Like for REs, we will use automata to analyze and understand the parsing process

Nondeterministic Top-Down Automaton

- The nondetermistic Top-Down parsing automaton $NTA(G)$ of the CFG $G =< \Sigma, N, P, S >$ is defined by
  - Input alphabet Σ
  - Pushdown alphabet $X = N \cup \Sigma$
  - Output alphabet $U =$ the rule numbers 1,2,3,...
  - States $\Sigma^* \times X^* \times U^*$
  - Two types of transitions for $w \in \Sigma^*, \alpha \in X, z \in U$:
    - **Expansion** of non-terminal symbol A using rule $A \rightarrow \beta$ with number i:
    $$(w, A\alpha, z) \rightarrow (w, \beta\alpha, z\ i)$$
    - **Matching** of terminal symbol $a \in \Sigma$:
    $$(aw, a\alpha, z) \rightarrow (w, \alpha, z)$$
  - Initial state (w, S, ε) for $w \in \Sigma^*$
  - Final state $(\varepsilon, \varepsilon, u)$ where $u \in U^*$

Example

$E \to E + T \mid T$ (1, 2)
$T \to T * F \mid F$ (3, 4)
$F \to (E) \mid a \mid b$ (5, 6, 7)

- Running the NTA on $w = (a) * b$ gives the leftmost analysis of w:
  - Initial state $\big((a) * b,\ E,\ \varepsilon\big)$
  - Rule 2 to expand E $\big((a) * b,\ T,\ 2\big)$
  - Rule 3 to expand T $\big((a) * b,\ T * F,\ 23\big)$
  - Rule 4 to expand T $\big((a) * b,\ F * F,\ 234\big)$
  - Rule 5 to expand E $\big((a) * b,\ (E) * F,\ 2345\big)$
  - Match the "(" $(\ a) * b,\ E) * F,\ 2345)$
  - Rule 2 to expand E $(\ a) * b,\ T) * F,\ 23452)$
  - Rule 4 to expand T $(\ a) * b,\ F) * F,\ 234524)$
  - Rule 6 to expand F $(\ a) * b,\ a) * F,\ 2345246)$
  - Match "a" $(\ ) * b,\ ) * F,\ 2345246)$
  - Match ")" $(\ * b,\ * F,\ 2345246)$
  - Match "*" $(\ b,\ F,\ 2345246)$
  - Rule 7 to expand F $(\ b,\ b,\ 23452467)$
  - Match "b" $(\ \varepsilon,\ \varepsilon,\ 23452467)$

Nondeterminism

- In the example on the previous slide, we "magically" knew which rule to apply, even in complicated situations
- For a CFG like

$$
\begin{aligned}
S &\rightarrow A \mid B && (1, 2)\\
A &\rightarrow b && (3)\\
B &\rightarrow Cd && (4)\\
C &\rightarrow a && (5)
\end{aligned}
$$

  how should the NTA know that in the initial state (ad, S, ε) the non-terminal S should be expanded to B and not A ?
- Possible implementation: Backtracking
  1. The parser first tries rule 1.
  2. After rule 3, it sees that it cannot match the input "a"
  3. It goes back to step 1 and tries rule 2.
- Backtracking works, but it requires:
  - Remember the places where a choice was made (here: at rule 1)
  - "Unread" the tokens already read when backtracking