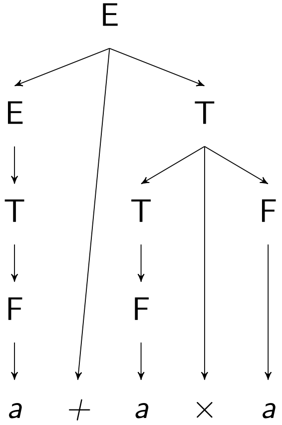
Example 2.4 I

- The following CFG handles mathematical expressions
- $G_4 = (V, \Sigma, R, \langle \text{expr} \rangle)$
  $V = \{\langle \text{expr} \rangle, \langle \text{term} \rangle, \langle \text{factor} \rangle\}$
  $\Sigma = \{a, +, \times, (,)\}$
  R:

$$
\begin{aligned}
\langle \text{expr} \rangle &\rightarrow \langle \text{expr} \rangle + \langle \text{term} \rangle \mid \langle \text{term} \rangle \\
\langle \text{term} \rangle &\rightarrow \langle \text{term} \rangle \times \langle \text{factor} \rangle \mid \langle \text{factor} \rangle \\
\langle \text{factor} \rangle &\rightarrow (\langle \text{expr} \rangle) \mid a
\end{aligned}
$$

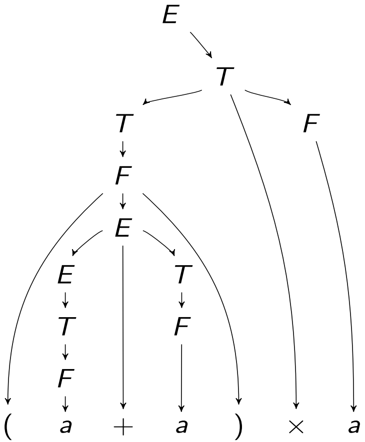
Fig 2.5: check $a + a \times a$

Example 2.4 II



check $(a + a) \times a$

Example 2.4 III



Design Grammars I

- $\{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$

$$S_1 \rightarrow 0S_1 1 \mid \epsilon$$
$$S_2 \rightarrow 1S_2 0 \mid \epsilon$$
$$S \rightarrow S_1 \mid S_2$$

- CFG versus DFA

  For a CFG that is regular, it can be recognized by DFA

Design Grammars II

Rules of CFG can be

$$R_i \rightarrow aR_j \text{ if } \delta(q_i, a) = q_j$$
$$R_i \rightarrow \epsilon \text{ if } q_i \in F$$

- We see the main difference between CFG and DFA
  CFG: a rule can be like

$$R_i \rightarrow aR_jb$$

Design Grammars III

DFA: a rule can only be

$$R_i \to aR_j,$$

where we treat each R_i as a state and let

$$\delta(R_i, a) = R_j$$

Ambiguity I

- The same string but obtained in different ways
- For the example of mathematical expressions discussed earlier, what if we consider the following rules?

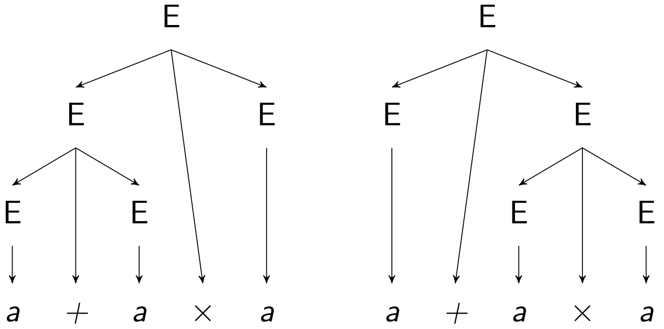
$$\langle \text{expr} \rangle \to \langle \text{expr} \rangle + \langle \text{expr} \rangle \mid$$
$$\langle \text{expr} \rangle \times \langle \text{expr} \rangle \mid (\langle \text{expr} \rangle) | a$$

- We see the following ways to parse

$$a + a \times a$$

Ambiguity II

- Fig 2.6



- This CFG does not give the precedence relation
- We want that $a \times a$ is done first

Ambiguity III

- By the more complicated CFG earlier, the parsing is unambiguous
- Question: how to formally define the ambiguity?
- We need to define "leftmost derivation" first

Leftmost derivation I

- Even for an unambiguous CFG we may have

  the same parse tree, but different derivations
- For the CFG discussed earlier we can do

$$\begin{aligned}\langle\text{expr}\rangle &\Rightarrow \langle\text{expr}\rangle + \langle\text{term}\rangle\\ &\Rightarrow \langle\text{expr}\rangle + \langle\text{term}\rangle \times \langle\text{factor}\rangle\end{aligned}$$

  where the second part is expanded.
- On the other hand, we can handle the first one first.

$$\begin{aligned}\langle\text{expr}\rangle &\Rightarrow \langle\text{expr}\rangle + \langle\text{term}\rangle\\ &\Rightarrow \langle\text{term}\rangle + \langle\text{term}\rangle\end{aligned}$$

Leftmost derivation II

- This is not considered ambiguous
- Definition of leftmost derivation: at every step the leftmost remaining variable is the one replaced.

Formal definition of ambiguity I

- w ambiguous if there exist

  two leftmost derivations
- Some context-free languages can be generated by ambiguous & unambiguous grammars
- We say a CFG is inherently ambiguous if it only has ambiguous grammars
- See prob 2.29 in the textbook. Details not given here.