CS4308 — Concepts of Programming Languages

Assignment #2 (Module 2): Solutions

Given grammar

$$
\begin{aligned}
\langle assign\rangle &\to \langle id\rangle \ = \ \langle expr\rangle \\
\langle id\rangle &\to A \mid B \mid C \\
\langle expr\rangle &\to \langle id\rangle + \langle expr\rangle \ \mid \ \langle id\rangle * \langle expr\rangle \ \mid \ (\langle expr\rangle) \ \mid \ \langle id\rangle
\end{aligned}
$$

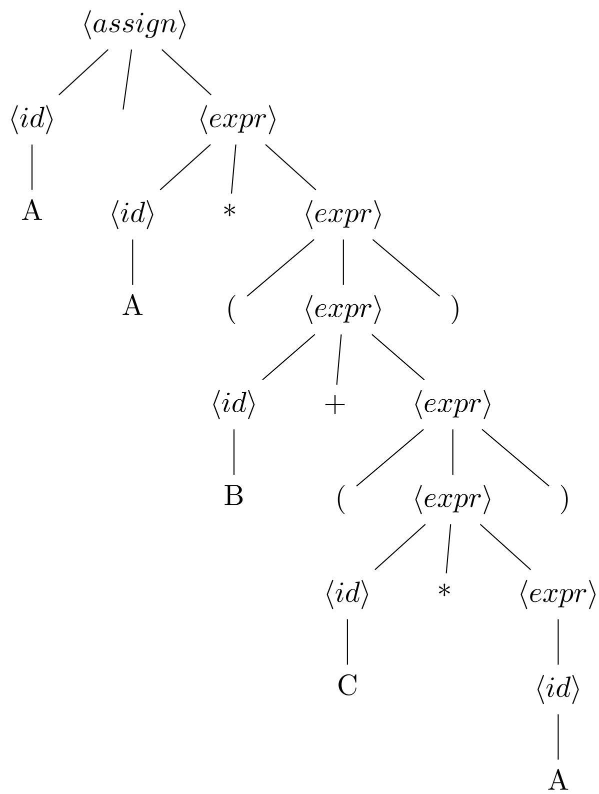
1. Parse trees and leftmost derivations

(a) $A = A * (B + (C * A))$

Leftmost derivation

$$
\begin{aligned}
\langle assign\rangle &\Rightarrow \langle id\rangle = \langle expr\rangle \\
&\Rightarrow A = \langle expr\rangle \\
&\Rightarrow A = \langle id\rangle * \langle expr\rangle \\
&\Rightarrow A = A * \langle expr\rangle \\
&\Rightarrow A = A * (\langle expr\rangle) \\
&\Rightarrow A = A * (\langle id\rangle + \langle expr\rangle) \\
&\Rightarrow A = A * (B + \langle expr\rangle) \\
&\Rightarrow A = A * (B + (\langle expr\rangle)) \\
&\Rightarrow A = A * (B + (\langle id\rangle * \langle expr\rangle)) \\
&\Rightarrow A = A * (B + (C * \langle expr\rangle)) \\
&\Rightarrow A = A * (B + (C * \langle id\rangle)) \\
&\Rightarrow A = A * (B + (C * A))
\end{aligned}
$$

Parse tree

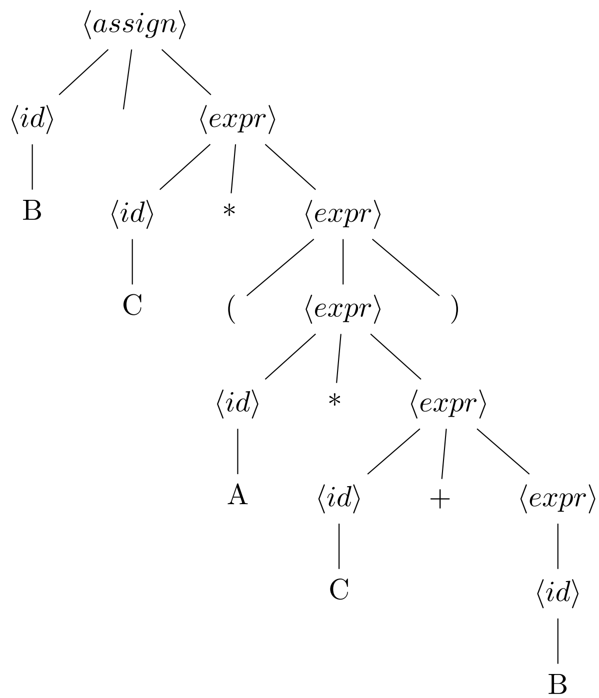


(b) $B = C * (A * C + B)$

Leftmost derivation

$$\begin{aligned}
\langle assign \rangle &\Rightarrow \langle id \rangle = \langle expr \rangle \\
&\Rightarrow B = \langle expr \rangle \\
&\Rightarrow B = \langle id \rangle * \langle expr \rangle \\
&\Rightarrow B = C * \langle expr \rangle \\
&\Rightarrow B = C * (\langle expr \rangle) \\
&\Rightarrow B = C * (\langle id \rangle * \langle expr \rangle) \\
&\Rightarrow B = C * (A * \langle expr \rangle) \\
&\Rightarrow B = C * (A * (\langle id \rangle + \langle expr \rangle)) \\
&\Rightarrow B = C * (A * (C + \langle expr \rangle)) \\
&\Rightarrow B = C * (A * (C + \langle id \rangle)) \\
&\Rightarrow B = C * (A * (C + B))
\end{aligned}$$

Parse tree



2. Convert to BNF

Given EBNF:

$$\langle S \rangle \to \langle A \rangle \; \{ \, b \, \langle A \rangle \, \}$$
$$\langle A \rangle \to a \, [b] \, \langle A \rangle$$

A BNF conversion that preserves the optional and repetition constructs is:

$$\langle S \rangle \to \langle A \rangle \, \langle R \rangle$$
$$\langle R \rangle \to b \, \langle A \rangle \, \langle R \rangle \mid \epsilon$$
$$\langle A \rangle \to a \, \langle O \rangle \, \langle A \rangle \mid a \, \langle O \rangle$$
$$\langle O \rangle \to b \mid \epsilon$$

Here, $\langle R \rangle$ expands the "zero-or-more" repetition of `b<A>`, and $\langle O \rangle$ realizes the optional `[b]`. The second rule for $\langle A \rangle$ provides the terminating (non-recursive) alternative needed in BNF.

3. Grammar for the language $\{ \, a^n b^n \mid n > 0 \, \}$

A concise grammar is

$$\langle S \rangle \to a \, \langle S \rangle \, b \; \mid \; ab$$

This generates one or more matching pairs of a's followed by b's.

4. Legality under the grammar

Given

$$\begin{aligned}\langle S\rangle &\to \langle A\rangle\ a\ \langle B\rangle\ b\\ \langle A\rangle &\to \langle A\rangle\, b \mid b \quad (\text{so } \langle A\rangle \Rightarrow b^k,\ k \geq 1)\\ \langle B\rangle &\to a\,\langle B\rangle \mid a \quad (\text{so } \langle B\rangle \Rightarrow a^m,\ m \geq 1)\end{aligned}$$

Therefore,

$$\langle S\rangle \Rightarrow b^k\ a\ a^m\ b \;=\; b^k\ a^{m+1}\ b \quad \text{with } k \geq 1,\ m \geq 1,$$

i.e., strings of the form *one or more b's, then at least two a's, then a single trailing b.*

Decisions

- (a) `baab` is legal ($k = 1,\ m = 1$).
- (b) `bbbab` is *not* legal (only one a before the final b).
- (c) `bbaaaaaS` is *not* legal (contains the nonterminal `S` as a literal symbol).
- (d) `bbaab` is legal ($k = 2,\ m = 1$).

Notes. The parse trees above follow the given right-recursive expression grammar; no operator precedence between + and $*$ is assumed beyond that imposed by parentheses.