BU CS320 Assignment 6: Context Free Grammars

November 6, 2023

1. Given the following grammar where $\langle expr \rangle$ is the starting symbol:

$$
\begin{array}{lll}
\langle id \rangle & ::= & \text{a} \mid \text{b} \mid \text{c} \mid \ldots \mid \text{z} \\
\langle dig \rangle & ::= & 0 \mid 1 \mid 2 \mid \ldots \mid 9 \\
\langle expr \rangle & ::= & () \mid \langle dig \rangle \mid \langle id \rangle \\
 & \mid & \text{let } \langle id \rangle = \langle expr \rangle \text{ in } \langle expr \rangle \\
 & \mid & \langle expr \rangle \text{ ; } \langle expr \rangle \\
 & \mid & \text{begin } \langle expr \rangle \text{ end}
\end{array}
$$

Demonstrate the grammar above is ambiguous.

Suppose we want to derive the statement let z = 3 in z ; c

Left-associative:

<expr>
<expr> ; <expr>
let <id> = <expr> in <expr> ; <expr>
let z = <expr> in <expr> ; <expr>
let z = <dig> in <expr> ; <expr>
let z = 3 in <expr> ; <expr>
let z = 3 in <id> ; <expr>
let z = 3 in z ; <expr>
let z = 3 in z ; <id>
let z = 3 in z ; c

Right-associative:

<expr>
let <id> = <expr> in <expr>
let <id> = <expr> in <expr> ; <expr>
let <id> = <expr> in <expr> ; <id>
let <id> = <expr> in <expr> ; c
let <id> = <expr> in <id> ; c
let <id> = <expr> in z ; c
let <id> = <dig> in z ; c
let <id> = 3 in z ; c
let z = 3 in z ; c

The grammar above generates at least 2 distinct parse trees for the sentenial form "let z = 3 in z ; c". This can be seen in the left-associatitve and right-associative derivations, which shows that the sentence can be derived from two different definitions of <expr> – <expr> ; <expr> and let <id> = <expr> in <expr>. Therefore, the grammar is ambiguous.

2. Modify the grammar (reproduced below) to be unambiguous. Hint: There is not just one way.

$\langle id \rangle$ ::= a | b | c | ... | z

$\langle dig \rangle$ ::= 0 | 1 | 2 | ... | 9

$\langle expr \rangle$ ::= () | $\langle dig \rangle$ | $\langle id \rangle$
| let $\langle id \rangle$ = $\langle expr \rangle$ in $\langle expr \rangle$
| $\langle expr \rangle$; $\langle expr \rangle$
| begin $\langle expr \rangle$ end

<id> ::= a | b | c | ... | z

<dig> ::= 0 | 1 | 2 | ... | 9

<val> :: = () | <dig> | <id>

<term> ::= <val> | begin <expr> end

<expr> ::= <term>
| let <id> = <expr> in <term>
| <expr> ; <term>

3. Demonstrate your modified grammar fixes the previously shown ambiguity.

Suppose we want to derive the statement let z = 3 in z ; c

Attempt 1:

<expr>
let <id> = <expr> in <term>
let z = <expr> in <term>
let z = <term> in <term>
let z = <val> in <term>
let z = <dig> in <term>
let z = 3 in <term>

Not able to derive the sentence
let z = 3 in z ; c

Attempt 2:

<expr>
<expr> ; <term>
let <id> = <expr> in <term> ; <term>
let z = <expr> in <term> ; <term>
let z = <term> in <term> ; <term>
let z = <val> in <term> ; <term>
let z = <dig> in <term> ; <term>
let z = 3 in <term> ; <term>
let z = 3 in <val> ; <term>
let z = 3 in <id> ; <term>
let z = 3 in z ; <term>
let z = 3 in z ; <val>
let z = 3 in z ; <id>
let z = 3 in z ; c

Only way to derive the sentence "let z = 3 in z ; c".
Therefore, grammar is no longer ambiguous.

<id> ::= a | b | c | ... | z

<dig> ::= 0 | 1 | 2 | ... | 9

<val> :: = () | <dig> | <id>

<term> ::= <val> | begin <expr> end

<expr> ::= <term>
| let <id> = <expr> in <term>
| <expr> ; <term>