BU CS320 Assignment 6: Context Free Grammars

November 6, 2023

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

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

Demonstrate the grammar above is ambiguous.

Suppose we want to derive the statement let z = 3 in z ; begin () end

Left-associative:

```
<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 ; begin <expr> end
let z = 3 in z ; begin () end
```

Right-associative:

```
<expr> ; <expr>
<expr> ; begin <expr> end
<expr> ; begin () end
let <id> = <expr> in <expr> ; begin () end
let <id> = <expr> in <id> ; begin () end
let <id> = <expr> in z ; begin () end
let <id> = <dig> in z ; begin () end
let <id> = 3 in z ; begin () end
let z = 3 in z ; begin () end
```

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

```
⟨id⟩   ::= a | b | c | ... | z

⟨dig⟩  ::= 0 | 1 | 2 | ... | 9

⟨expr⟩ ::= () | ⟨dig⟩ | ⟨id⟩
         |  let ⟨id⟩ = ⟨expr⟩ in ⟨expr⟩
         |  ⟨expr⟩ ; ⟨expr⟩
         |  begin ⟨expr⟩ end
```

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

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

<exprs> ::= () | <dig> | <id>
          |  let <id> = <expr> in <expr>
          |  begin <expr> end

<expr> ::= <exprs> | <exprs> ; <expr>
```

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

Left-associative:

```
<exprs> ; <expr>
let <id> = <expr> in <expr> ; <expr>
let z = <expr> in <expr> ; <expr>
let z = <exprs> in <expr> ; <expr>
let z = <dig> in <expr> ; <expr>
let z = 3 in <expr> ; <expr>
let z = 3 in <exprs> ; <expr>
let z = 3 in <id> ; <expr>
let z = 3 in z ; <expr>
let z = 3 in z ; <exprs>
let z = 3 in z ; begin <expr> end
let z = 3 in z ; begin <exprs> end
let z = 3 in z ; begin () end
```

Right-associative:

```
<exprs> ; <expr>
<exprs> ; <exprs>
<exprs> ; begin <expr> end
<exprs> ; begin <exprs> end
<exprs> ; begin () end
let <id> = <expr> in <expr> ; begin () end
let <id> = <expr> in <exprs> ; begin () end
let <id> = <expr> in <id> ; begin () end
let <id> = <expr> in z ; begin () end
let <id> = <exprs> in z ; begin () end
let <id> = <dig> in z ; begin () end
let <id> = 3 in z ; begin () end
let z = 3 in z ; begin () end
```