BU CS320 Assignment 6: Context Free Grammars

November 6, 2023

1. Given the following grammar where ⟨*expr*⟩ is the starting symbol:

```
⟨id⟩   ::= a | b | c | ... | z

⟨dig⟩  ::= 0 | 1 | 2 | ... | 9

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

Demonstrate the grammar above is ambiguous.

To demonstrate the ambiguity, we consider the string 'begin let a = b in c ; d end". The string can be parsed to at least two distinct ways.

1. "let" expression is limited to the first "<expr>" following the "in" keyword.

<expr>
→ begin <expr> end
→ begin <expr> ; <expr> end
→ begin let <id> = <expr> in <expr> ; <expr> end.
→ begin let a = b in c ; d end.

In this one, 'let a = b in c" is treated as one expression, and "d" is treated as a separate expression.

2. the scope of "let" expression encompasses both expressions following the "in" keyword.

<expr>
→ begin <expr> end
→ begin let <id> = <expr> in <expr> end
→ begin let a = b in <expr> ; <expr> end
→ begin let a = b in c ; d end.

In this one, 'let a = b in c ; d" is treated as a single expression, with 'd' is executed in the context where 'a' is bound to 'b'.

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

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

<let_expr> ::= let <id> = <expr> in <expr>

<seq_expr> ::= <expr>; <seq- expr> | <atom>

< block_expr> ::= begin <expr> end | <let_expr>|<seq-expr>

<expr> ::= <block_expr>| <atom>

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

This modified version clearly defines the precedence of 'let' binding within '<let_expr>'. separating them from sequential expressions. It ensures that sequential expressions are evaluated in order from left to right by '<seq_expr>'. And it enforces the expressions within 'begin... end" blocks are treated as single units with '<block_expr>'.