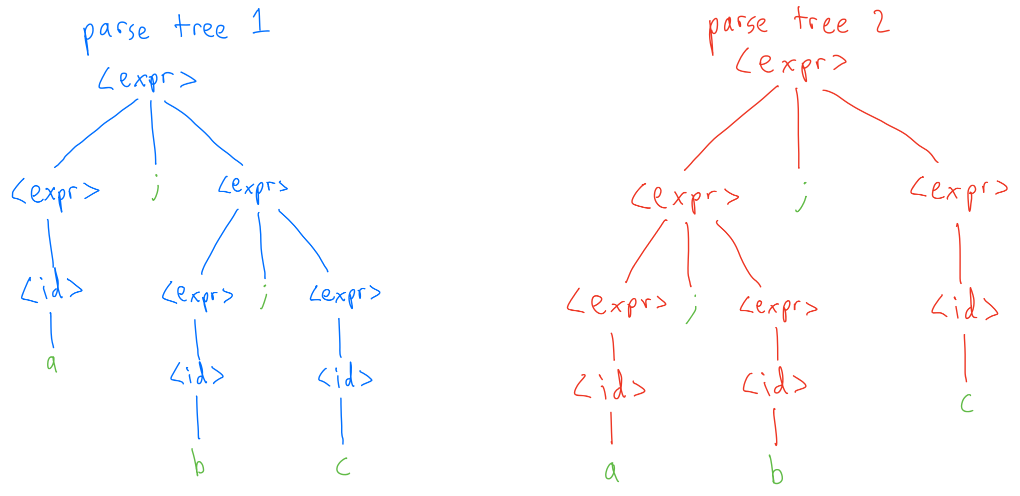
1. Examine the valid sentence: a ; b ; c



Since this grammar can produce a sentence that can be derived by 2 (or more) distinct parse trees, the grammar is ambiguous.

2. modified grammar with changes in red:

```
<id> ::= a | b | c | ... | z
<dig>::= 0 | 1 | 2 | ... | 9
<expr>::= () | <dig> | <id>
          | (let <id> = <expr> in <expr>)
          | ( <expr> ; <expr>)
          | ( begin <expr> end )
```

3. Including parentheses around more complex expressions in the grammar forces any valid sentences to have these disambiguating parentheses.

For example, a ; b ; c is not valid anymore — it would have to be either (a ; (b ; c)) or ((a ; b) ; c)

Now, every <expr> will evaluate to either the trivially unambiguous (), <dig>, or <id> or the one of the other more involved expressions that are wrapped by parentheses which naturally disambiguates an expression because it forces the order of derivation, inherently disallowing for multiple distinct parse trees for a single sentence.

(The parentheses are only really needed around <expr> ; <expr> because that displays symmetry but it does not hurt to include more parentheses in the grammar)