

4predictive-parsing-2-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: Predictive parsing 2

1. To get more experience with parse trees, please redo steps 5, 6, and 8 of the [parsing and syntax lab](#) . However, when you are asked to derive a string, create a parse tree.

2. Here is a BNF grammar for an arithmetic expression with subtraction:

`expr ::= expr - expr | NUM`

Is this grammar ambiguous?

3. If you think the grammar is ambiguous, show two different parse trees that can be derived from the grammar. If you don't think the grammar is ambiguous, go to step 1 (-:-).

4. Define "the language of a grammar" in your own words. Write your definition down.

5. Write a grammar that has the same language as the grammar of 1 above, but that is not ambiguous. Don't worry about removing left recursion for now.

6. Is your unambiguous grammar correct in the sense that if you evaluate operations in the tree from the bottom up, you get the correct value?

If you still have time, continue where you left off in the [predictive parsing 1 lab](#) .

Hints

1. Yes it is.

2. Try the string "4 - 3 - 2".

3. The language of a grammar is the set of all strings that can be derived from the grammar.

4. It is not necessary to introduce a new non-terminal. Don't read on unless you want another hint. Second hint: replace one of the expr non-terminals with NUM.

5. Subtraction is a left-associative operator, so 4 - 3 - 2 should be interpreted as (4 - 3) - 2. Hint 2: this grammar works correctly: `expr ::= expr - NUM | NUM`.

Published by [Google Drive](#) – [Report Abuse](#) – Updated automatically every 5 minutes
