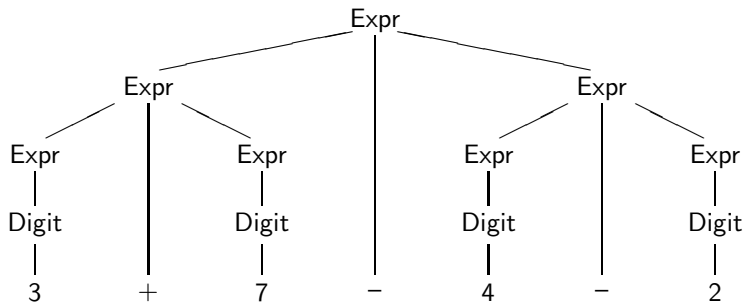
Concurrence et Compilation 2001

(Solutions)

1 Overview and Foundations

Exercise 1

(a) The expression 3 + 7 – 4 – 2 can be derived (among others, see part (c) of this exercise) by the following concrete syntax tree:



(b) The expression – 5 + 2 cannot be derived from the grammar, hence does not belong to the language produced by it. The reason is that every + and – sign has to be embraced by an expression, and there is no way of deriving the *empty word* ϵ as an expression.

More formally (not necessary for the course!), three observations lead to the conclusion that – 5 + 2 cannot be derived.

1. *Context-freeness:* The left-hand sides of the production rules consist of one nonterminal each. This implies that tokens like + and – cannot be erased or modified.

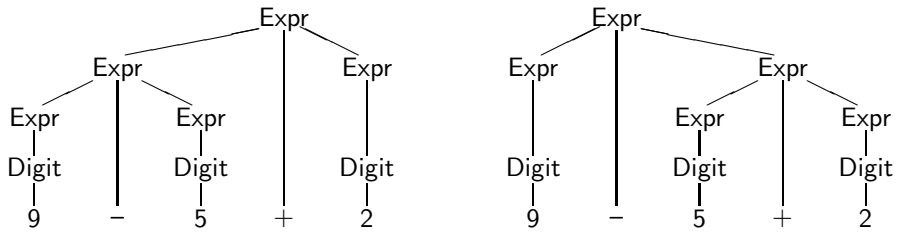
2. There is one production each for deriving a + and a – sign; in these, every + and – sign is enclosed in an expression.

3. *Monotonicity:* In every derivation $v \to w$, the sequence w contains at least as many tokens and nonterminals as v. As a consequence, the empty word ϵ cannot be derived.

The first and second observation imply that in order to derive – 5 + 2, we can either start with the production Expr ::= Expr "+" Expr, or with Expr ::= Expr "–" Expr.

- In the first case, we would then have to derive – 5 for the left-hand expression, for which the production Expr ::= Expr "–" Expr would have to be applied. For the left-hand expression thereof, one would now have to derive ϵ. But for this there is no production, or chain of productions, according to the third observation.
- In the second case, one would immediately have to derive ϵ for the left-hand expression.

(c) The grammar is ambiguous. Already for the example expression 9 – 5 + 2, there exist two ways of deriving it (how many different ways can you find of deriving the expression from part (a) of this exercise?):



The reason is that the productions for Expr contain Expr on both left-hand and right-hand sides of the + and – signs. We can disambiguate the grammar by replacing one of the two occurrences with Digit:

```
Expr   ::=  Expr "+" Digit
         |  Expr "–" Digit
         |  Digit
Digit  ::=  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```