# TDT4205 - PS 3

Øyvind Skaaden (oyvindps@stud.ntnu.no)
March 6, 2022

## 1 Bottom-up parsing

### 1.1 LR(0) automaton

The LR(0) automaton can be seen in fig. 1.1.

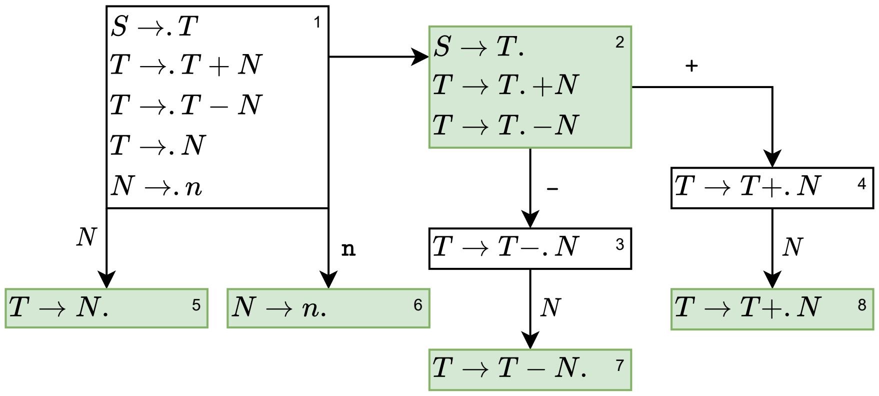

**Figure 1.1:** LR(0) automaton for the grammar.

### 1.2 SLR parsing

The grammar is SLR. By looking at the parsing table in table 1.1, we can see that there is not any shift-reduce conflicts. We have selectively reduced or shifted based on the FOLLOW sets.

**Table 1.1:** The SLR parsing table for the automaton.

| | n | + | - | $ | $T$ | $N$ |
|---|---|---|---|---|---|---|
| 1 | s6 | | | | g2 | g5 |
| 2 | | s4 | s3 | **A** | | |
| 3 | | | | | | g7 |
| 4 | | | | | | g8 |
| 5 | r3 | r3 | r3 | r3 | | |
| 6 | r4 | r4 | r4 | r4 | | |
| 7 | r2 | r2 | r2 | r2 | | |
| 8 | r1 | r1 | r1 | r1 | | |

## 2 Tree simplification

See the attatched code.

### 2.1 Eliminate nodes of purely syntactic value

Removed the nodes with the type `GLOBAL`,`ARGUMENT_LIST`,`PARAMETER_LIST`,`STATEMENT`,`PRINT_ITEM` and `PRINT_STATEMENT`.

If the parent node is `PRINT_STATEMENT`, set the child to `PRINT_STATEMENT` to keep the print structure.

### 2.2 Flatten list structures

If the node is any of `GLOBAL_LIST`, `STATEMENT_LIST`, `PRINT_LIST`, `EXPRESSION_LIST`, `VARIABLE_LIST` or `DECLARATION_LIST` the compiler will try to flatten the list, but keep the top most list node.

### 2.3 Resolve constant expressions

If both children are of the `NUMBER_DATA` type, try to calculate the new value. The same for single children expression, like negative and bit-wise not.