Universitetet i Oslo
Institutt for Informatikk

Reliable Systems
Martin Steffen, Gianluca Turin

INF 5110: Compiler construction

Spring 2021 **Series 4** 12. 3. 2021

Topic: Chapter 5: LR parsing

Issued: 12. 3. 2021

Exercise 1 (LR(0)-items, SLR(1) parsing) Consider the following grammar for well-balanced parentheses:

$$S \;\rightarrow\; S\,\mathbf{(}\,S\,\mathbf{)} \;\mid\; \boldsymbol{\epsilon}$$

1. Construct the DFA of LR(0) items for the grammar.
2. Construct the SLR(1) parsing table.
3. Show the parsing stack and the actions of an SLR(1) parser for the input string

$$\mathbf{(\,(\,)\,(\,)\,)}\;.$$

4. Is the grammar LR(0)? If not, describe a resulting LR(0) conflict. If yes, construct the LR(0) parsing table and describe how a parse might differ from an SLR(1) parse.

Exercise 2 (LR(1) parsing)

1. Show that the following grammar is not LR(1):

$$A \rightarrow \mathbf{a}A\mathbf{a} \;\mid\; \boldsymbol{\epsilon}$$

2. Is the grammar ambiguous or not?

Exercise 3 (Bottom-up parsing) The following ambiguous grammar generates the same language as the grammar of Exercise 1 in this collection (namely all strings of well-balanced parentheses):

$$A \rightarrow AA \;\mid\; \mathbf{(}\,A\,\mathbf{)} \;\mid\; \boldsymbol{\epsilon} \tag{1}$$

Will a `yacc`-generated parser using this grammar recognize all legal strings? Why or why not?

Extra: Try to change the order: put the production $A \rightarrow AA$ at the end.

Exercise 4 (Priorities & associativity by manual conflict resolution) Take the following variant of the "expression grammar"

$$\begin{array}{rcl} exp' & \to & exp \\ exp & \to & exp + exp \mid exp * exp \mid \mathbf{n} \end{array}$$

and extend it with exponentiation as follows

$$\begin{array}{rcl} exp' & \to & exp \\ exp & \to & exp + exp \mid exp * exp \mid exp \uparrow exp \mid \mathbf{n} \end{array}$$

Assume that the usual associativities and precedences are intended (which includes right-associativity for exponentiation).

Now: indicate how *conflicts* in an LR-parse-table are to be resolved (if possible) to obtain the indicated behavior.

Exercise 5 (Bottom-up parsing routine) [1] Consider the following grammar G, where S is the start symbol, and the terminals as # and **a**

$$\begin{array}{rcl} S & \to & T\,S \\ S & \to & T \\ T & \to & \#\,T \\ T & \to & \mathbf{a} \end{array}$$

Now do:

1. calculate the first and follow sets of S and T. Use, as in the lecture, **$** to stand for the end-of-input.
2. formulate, in your own words, which words of terminals are derivable from S.[2]
3. Decide if you can formulate a regular expression that captures words of # and **a** derivable from S.[3] If the answer is yes, give a regular expression that captures the language.
4. Introduce a new start symbol S' and construct the LR(0)-DFA for G directly from that grammar. Enumerate the states.
5. Give the parsing table for that grammar, and let the type of the grammar should determine the form of the parsing table.
6. Show how

   a # a

   is being parsed; do that in the form presented in the book/lecture, making use of the yet-to-parse input and the stack and indicate the shift and stack operations appropriately during the parsing process.

[1] It corresponds to an exam question from 2006, minus one sub-question.

[2] The "language of S".

[3] Is $\mathcal{L}(G)$ regular?