JOHANNES KEPLER
UNIVERSITY LINZ

Bachelor's Thesis
LALR(1) Bottom-up Parser Generator

Student: Alexander Voglsperger
Advisor: Dipl.-Ing. Dr. Markus Weninger, BSc
Start date: February 2023

Dipl.-Ing. Dr. Markus Weninger, BSc
Institute for System Software
T +43-732-2468-4361
markus.weninger@jku.at

In the lecture "Compilerbau" (Compiler Construction) at the Johannes Kepler University, students are tasked to develop a top-down LL(1) parser throughout the semester. This parser always targets the language "MicroJava", with only slight deviations from year to year. Thus, the amount of work that has to be put in by the lecturers to prepare the material and homeworks is reasonable.

Yet, at the end of the semester, the course also teaches how bottom-up LALR(1) parsers work (using so-called *shift* and *reduce* operations stored in a *state table)*. If the input has no syntax errors, the parser repeatedly performs these steps until the whole input has been consumed and the complete parsing tree has been built. In the homework targeted at LALR(1), it is the students' task to derive the *state table*, i.e., which shift and reduce operations have to be performed in which situations. The same work has to performed by the lecturers that prepare the sample solution for the homework - most of the time even multiple times due to reworkings and restructuring of the assignment. This is error-prone and time consuming.

The goal of this thesis is to develop a generator for bottom-up LALR(1) state tables given a user-specified grammar. The grammar is given in BNF such as

```
S = A B.
A = x x.
A = x y z.
B = .
B = z.
```

Based on this, the generator should simulate which situations (“states”) the bottom-up parser might encounter, as taught in the lectures. Data structures such as `Production`, `NTS` (non-terminal symbol), `TS` (terminal symbol), `Item`, `State` and `StateTable` might proof useful to keep track of any information about the grammar under test as well the generator's current state. In the end, a .csv file that lists the various shift and reduce operations in the different states should be produced that can be handed out as a sample solution.

As a last part of this thesis, a system should be developed that can “run” the parser and determine whether a certain sentence corresponds to the underlying grammar. For example, it should be able to simulate whether the sentence "xxy" will be accepted by the grammar mentioned above. If this is not the case, the system should be able to “recover” from the parsing error as taught in the lecture, i.e., it should calculate the “escape path” based on the states’ guide symbols and use this information to synchronize the parser’s current state with the remaining input.

Modalities:
The progress of the project should be discussed at least every four weeks with the advisor. A time schedule and a milestone plan must be set up within the first 3 weeks and discussed with the advisor. It should be continuously refined and monitored to make sure that the thesis will be completed in time. The final version of the thesis must be submitted not later than 31.07.2023.