Program Optimisations and RTS SS 2019

Assignment 2: AST for ANSI C99 programs

Now the abstract syntax tree (AST) for executable statements shall be constructed – not for declarations. Declarations are usually not stored as abstract syntax tree but saved in a symbol table and identifiers used in statements point to the corresponding symbol table entry. In this assignment identifiers which are part of statements shall be stored as terminal symbols, i.e. no symbol table is required.

The following tasks have to be done:

1) Define the AST data structure for saving executable statements of functions.
2) Extend the parser to generate the AST based on synthesized attributes.
3) Visualize the AST in an easy to read fashion for verifying correctness. This shall be done based on the open-source Graphviz package which provides a command-line tool `dot` which reads a DOT file and produces a directed graph in png format (several output formats are supported). The DOT language is a rather simple graph description language.
   The Graphviz package is installed on VM scws15.cs.univie.ac.at and can be installed by yourself elsewhere. If you want to use another visualization tool, please let me know in advance.
4) Modify the makefile to support `dot` calls to generate the png-files.

Command line interface:

```
$ cparser test_function.c
```

generates output file: `test_function.dot`

Build a file assignment2.zip containing the following files and upload it to Moodle:

1) Documentation.pdf: containing a description of the data structures and design decisions
2) Makefile: for creating an executable, analyzing `test_function.c` and creating `test_function.dot` and `test_function.png`, and cleaning files
3) parser.y
4) scanner.l
5) intermediate.c: C functions for constructing and visualizing AST
6) header files (if any)
7) test_function.c: one (non-trivial) test function for which your program works correctly

Due date: **4.5.2019**