AtoCC - Learning environment for teaching theory of automata and formal languages

Michael Hielscher
Dept. of Computer Science
University of Applied Science Zittau/Goerlitz
Goerlitz, Germany
mail@genesis-x7.de

Christian Wagenknecht
Dept. of Computer Science
University of Applied Science Zittau/Goerlitz
Goerlitz, Germany
c.wagenknecht@hs-zigr.de

ABSTRACT
The learning environment AtoCC is presented to be of use in teaching abstract **a**utomata, formal languages, and some of its applications in **c**ompiler **c**onstruction. From a teacher's perspective AtoCC aims to address a broad range of different learning activities forcing the students to actively interact with the subjects being taught.

Categories and Subject Descriptors
F.1.1 [**Models of Computation**]: Automata (e.g., finite, push-down, resource-bounded); F.4.3 [**Formal Languages**]: Classes defined by grammars or automata

General Terms
Languages, Theory

Keywords
Automata publication, formal languages, compiler construction

1. PEDAGOGICAL BACKGROUND
With the help of AtoCC a student is forced to practice various mental activities of learning. Naturally, teaching of these topics needs to provide printed material, presentations onto the Web and slides, respectively, that include the definitions of automata using figures to depict the appropriate state diagrams with high quality of rendering. This is facilitated by AtoCC. By means of AtoCC the students internalize theoretical aspects by actively work on exercises and little projects within a pedagogical designed software environment. The theory of formal languages and abstract automata are the fundamentals of the compiler construction principles. T diagrams are widely used to model compilers composed of individual components. The diagrams can also be used to illustrate the particular steps of source code compilation. Fig. 1 exposes the different teaching aims associated with the individual tools belonging to AtoCC. Furthermore the main kinds of their contributions to the student's mental activities are specified.

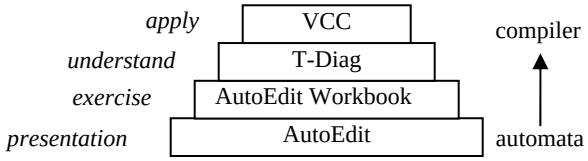

Figure 1: Contributions, tools, and teaching aims

2. LEARNING ENVIROMENT
AtoCC covers a set of four interconnected tools. For example, it is possible to automatically create an automaton from a compiler description which makes the compiler itself comparable to the results of the theory. However, each tool can also be used individually.

AutoEdit [1] is a tool for creation and simulation of automata (similar to the well known JFLAP [2]). AutoEdit allows for high quality graphs of automata to be created for quite different purposes, including JPEG files for Web presentation or vector graphics (for example SVG) for print media, respectively. **AutoEdit Workbook** which is built on the top of AutoEdit is a client/server tool. It provides a large set of exercises. Students can practically test their knowledge. With an administration tool the teacher can easily add new exercises to the server database. **T-Diag** allows the student to assemble so called T diagrams which are used to model compiler construction processes. Compilers like javac or csc (.NET C#) as well as those created by the students can be interactively used to simulate the developed diagrams.
VCC (Visual Compiler Compiler) offers a graphical user interface for compiler construction. VCC produces compiler source code for both C# and Scheme based on adapted versions of YACC. VCC combines the scanner generator (similar to LEX) with the parser generator (similar to YACC) to get one unique tool. Tokens and rules of the underlying formal language can be stored in one single file. All four tools store their respective data in XML files (validated by XML-Schemas) to be used for multiple purposes.

3. REFERENCES
[1] Hielscher, M.; Wagenknecht, Chr.: *AutoEdit - a tool for editing, simulation, transformation, and publication of abstract automata* - In: Pedagogical concepts for CS education. INFOS'05 28.-30.09.05 Dresden – 2nd. vol. (Ed.: H. Rohland), pp. 25-27.

[2] Rodger, S. H.: JFLAP at http://www.jflap.org