

ONLINE EXERCISE SYSTEM

A Web-Based Tool for Administration and Automatic Correction of Exercises

Daniel Baudisch, Manuel Gesell and Klaus Schneider

Embedded Systems Group, University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern, Germany

baudisch@cs.uni-kl.de, gesell@cs.uni-kl.de, schneider@cs.uni-kl.de

Keywords: CAT, CAL, computer aided teaching, computer aided learning, Online Exercise System (OnExSy), web-based exercise tools, submission, automatic correction.

Abstract: We describe OnExSy, our online exercise system that allows our students in computer science to submit their solutions to exercises via a web form. The system is not only able to automate the administration of the student accounts as done by many other comparable systems. In addition, our system is even able to automatically check the correctness of the submitted solutions and is able to provide a counterexample in case the solution is wrong. This feature is provided by modern formal verification tools that have also been developed in our research group. As a result, our tutors do not waste time for checking the correctness of the solutions of many students, and instead, can spend the saved time for an individual support of students. Moreover, the tool is available not only during the time when lectures are held, but also during holidays which is very important for the final preparation of the exam.

1 MOTIVATION

In the Bachelor program in Computer Science at the University of Kaiserslautern, ‘*Computer Systems 1& 2*’ is one of the required courses of the first two semesters. This course covers the basic principles of computer systems, especially the design of digital hardware circuits, the design of microprocessors including computer arithmetic and assembler programming, and the overall architecture of a computer system like the memory hierarchy and peripherals. The overall number of students in this course is about 200 students each semester including also students of other disciplines like mechanical engineering.

As most courses of our Bachelor programs, ‘*Computer Systems 1& 2*’ also offers exercises to provide a deeper understanding of the topics. The classical exercises arrogate the students to provide their solutions on paper that have to be submitted each week. After the correction of the students’ solutions by a tutor, the tutor presents the correct solution in the lesson and may also talk about the students’ solutions and their mistakes. Usually, the tutor himself is a student in higher semester.

In the past, we recognized an increasing failure rate of our students as a growing problem in our department. This may be traced back to the following reasons: The Bachelor program has a tight schedule of courses and the exams have to be passed within given deadlines. Although exceptions from these rules are allowed, most students try to meet these requirements. This leads to the fact that less students have time for jobs as tutors, so that the *number of engaged tutors decreases while the number of students grows*. This leads to a reduced individual support of our students, since the *available time per week of the tutors is consumed by the correction* of the submitted solutions.

A further problem is that a minimal amount of exercises have to be solved correctly to be able to apply for the exam in this course. Due to limited time during the week, it seems that students *cheat by copying the results of other students*, which however harms their preparation of the exam at the end. Detecting copies of submitted solutions of 200 students every week that are distributed over many tutors can not be done with reasonable effort.

In this paper, we describe how we support our stu-

dents by improving their individual discussion with their tutors. Besides providing special courses for the preparation of the tutors in the future, we have to make sure that they can spend their time in the really important tasks, i.e., tasks that cannot be solved by a computer: *The submission of the students' solutions, the distribution to their tutors, and the maintenance of the students' accounts* are typical tasks that can be done even better with a computer, and clearly, our tool OnExSy provides this functionality.

In addition to this, our tool OnExSy is also able to *automatically check the correctness of the submitted solutions and to immediately provide a feedback to the students in case their submission was wrong*. In many cases, checking the correctness of a solution is trivial, e.g., if the solution is a simple number. In other cases, e.g., if the solution is a digital circuit or a computer program, many correct solutions are possible, so that we cannot simply check for equality of a sample solution. Instead, the semantics has to be checked, which amounts to the *automatic verification of programs and circuits*. Fortunately, the research done by our group already considered this problem in great detail, so that we are happy to use the tools we originally developed for research also for teaching.

As a result, our tutors now have much more time to discuss potential problems with the students, since for most exercises, there is no longer a need for correcting the submitted solutions. This leads to an *improved individual support of each student with even less tutors*. Tutors can now even prepare the exercises of the next week instead of only explaining why a submitted solution was not correct. In many cases, the feedback of OnExSy was sufficient to solve this problem.

To get rid of copied solutions, we furthermore added the feature to OnExSy that *each student is given a different exercise*. This is either done by randomly generating an instance of an exercise, e.g. in computer arithmetic where two binary numbers of a certain bitwidth are randomly chosen, or by selecting an exercise from a data base. Copying a solution made by another student is therefore no longer possible (unless the rare case should occur that two students were given the same exercise).

The only problem that remains is that we do not yet check the *authentication of the students*, i.e., it is possible that another student provides the solutions. We do not believe that this is currently the case among our own students (at least not to a intolerable degree) as already reported by other Universities (Ross, 2005), where even students of other countries have been paid for submitting online solutions.

Automatic correction of exercises is also done by

some other tools used in other areas for teaching like CalMæth (CalMæth, 2008). Clearly, there already exist a lot of tutoring tools starting with small tools for specific problems, e. g. JADE (Java Decision Diagram Package) (JADE, 2003), and ending with complex user interfaces for online learning that intent to provide flexible support of questionnaires, e. g. Lecturnity and Dynamic Power Trainer from IMC (imc.de, 2008). Inspired by these tools and our goal mentioned above, we decided to implement a compact and modular exercise system that is capable of

- creating individual exercises for each student to get rid of copied solutions
- online (web-based) submission of the students' solutions
- automatically checking the correctness of submitted solutions with immediate feedback, e.g. by counterexamples
- training tools to solve further exercises for preparing the final exam.

Concerning the latter point, we specify a maximal number of trials for each exercise. If this number of trials is exceeded, the sample solution is presented.

Note, that these tools are not intended to replace the course or even the lecturer. Instead, these tools are used to complement and guide the preparation for the final exam by providing 24h a day the opportunity for our students to train themselves on the required exercises.

The remainder of this paper is structured as follows: Our group developed two independent tool sets to achieve the mentioned goals. The first tool set is named Online Exercise System (OnExSy) and is described in the next chapter. This section is followed by the description of the second tool set, which is called Online Training Tools. Finally, we discuss the results and experiences of these tools and close this paper with a discussion of future enhancements.

2 ONLINE EXERCISE SYSTEM

The Online Exercise System (OnExSy) is a modular system that consists of a user interface and a collection of programs, which are necessary to create randomized exercises and to check provided solutions. OnExSy is built of independent modules (single subsequent tools). The modularity of OnExSy is one of the main features, keeping the system manageable and simplifying its extensibility for the future.

In the following, we describe OnExSy's modules, i.e., its user interface followed by a short description of the underlying programs.

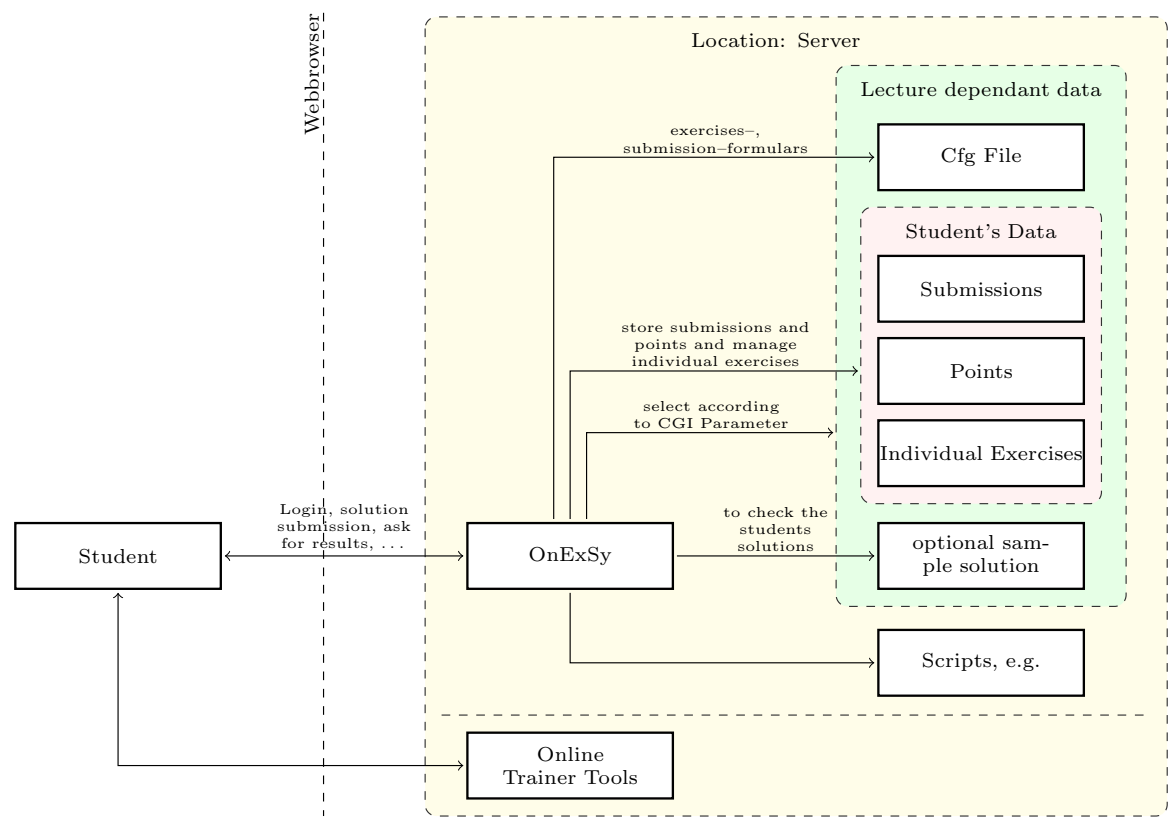


Figure 1: An overview of our Online Exercise System.

2.1 USER INTERFACE

The user interface is the main part of OnExSy that provides the fundamental functions to run the system and to connect it to the graphical user interface, i. e. the web browser, and to the server. The basic functions implemented in the user interface are listed below and will be explained later in more detail.

- The *registration of students* including a fair *distribution to tutorials* (if more than one date is available) with an external program.
- The system also *publishes the group assignments on a web page*.
- It assigns new (individual) exercises to the students.
- Furthermore, the user interface checks the correctness of submitted solutions by external programs and the stores the achieved points of the student in a data base.
- Finally, it provides statistics of submitted solutions and achieved points of the students which

can be seen by the tutors (for his class) or by the student (for his account).

The user interface is based on an Apache web-server (APA08, 2008). The system was written in Moscow ML¹, so that the system can be ported to different platforms like Linux, Windows and MacOS.

The registration is necessary to get the students' personal information like names and matriculation numbers. If the lecture offers more than one date for exercise lessons, the student has to choose three different dates with three priorities during the registration.

To give all students the same chance to get into their preferred group, the final classification to the exercise lessons is done after the registration deadline by an external program. This program tries to find a satisfying classification, i.e. usually, the students can be classified to their first or second selected exercise lessons. The result of this classification is also displayed by the system.

¹<http://www.itu.dk/sestoft/mosml.html>

Each lecture that offers exercises has a configuration file that contains a description of its exercises. This includes descriptions for all exercise sheets and all exercises. A description for an exercise sheet can have a link, e. g. to a PDF file, a description in HTML or/and an arbitrary number of exercise descriptions. Each exercise description consists of an optional link or/and an HTML description and the description of the solution checking script. The latter is simply given as a command line. We will describe its functionality later.

To access the correct data and the available functions mentioned above, the user interface receives via a CGI interface the parameters defining the lecture, the semester and the function that has to be executed. Additionally, one can define additional CGI parameter, e. g. the language. However, in our implementation this parameter affects only the user interface, but basically it is possible to select an exercise with the specified language.

Of course, our system supports common exercises that have to be made by all students. These exercises can be put in a familiar way like publishing a link to an electronic version of the exercise sheet or distributing a printed version. The students calculate the results of the exercises as usual with paper and pencil or specific tools like Hades (Hades, 2007) (a design tool for digital circuits). Having calculated the results, the students can submit them via a web page. They just have to open the link on the corresponding page of the exercise lesson. In particular, after the student has opened the exercise site, he is asked for his name and his matriculation number to login. These informations are necessary to be entered at the beginning of the session to enable the assignment of individual exercises. After the login, the student sees an overview over the exercise sheets and the exercises (see Figure 2). After having selected an exercise, the submission form is opened (see Figure 3). The next step is to enter the calculated result (see Figure 4). After pressing the button for submission, the result is sent to our server and is automatically checked. The next screen shows whether the submitted result was correct (see Figure 5). The result window also shows the output of the solution checker, e. g. messages like 'syntax error', i.e. the submitted solution was given in a wrong format, or as can be seen, a counterexample if a submitted solution was not correct. The students do not have to submit all exercises at once. They can even submit the solutions to parts of an exercise one after the other (at different points of time). If a solution is not correct, the student can submit a modified solution. The number of trials can be limited by the configuration program, but can also be set to infinity.

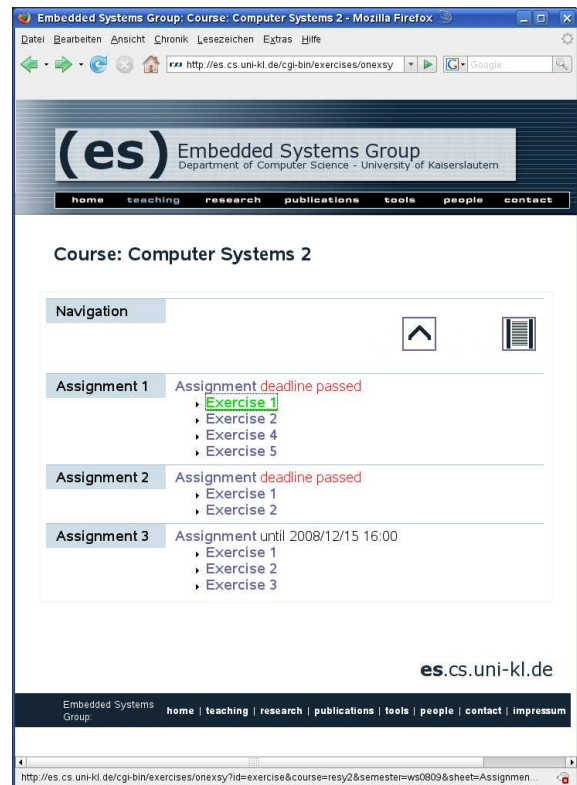


Figure 2: After the login, OnExSy presents an overview of all assignments.

Furthermore, it is important to know that all given solutions are stored and will not be overwritten by subsequent ones. In particular, correct submissions will not be invalidated by incorrect or empty ones.

However, in the last semester, we experienced that quite a lot of students reached nearly 100% of the points of the online exercises, but nevertheless failed in the final written exam. We believe that they copied the solutions of other students, which is very simple via email and other means of communication.

We reacted in two ways to this experience: First, we reorganized the classical procedure of exercise lessons. Instead of working out the sample solution of the previous exercise sheet, we focus more on preparing the solution of the next exercise sheet. In particular, the tutors are instructed to give useful hints and - most important - to clarify the exercises to avoid misunderstandings.

Second, OnExSy supports individual exercises for every student that are randomly generated. To this end, we can assign a particular program for each submission form that generates an exercise. When the user interface of OnExSy is instructed to dynamically generate the submission form for the selected exercise, it checks the configuration file for an exercise generator program. If no task has been created for

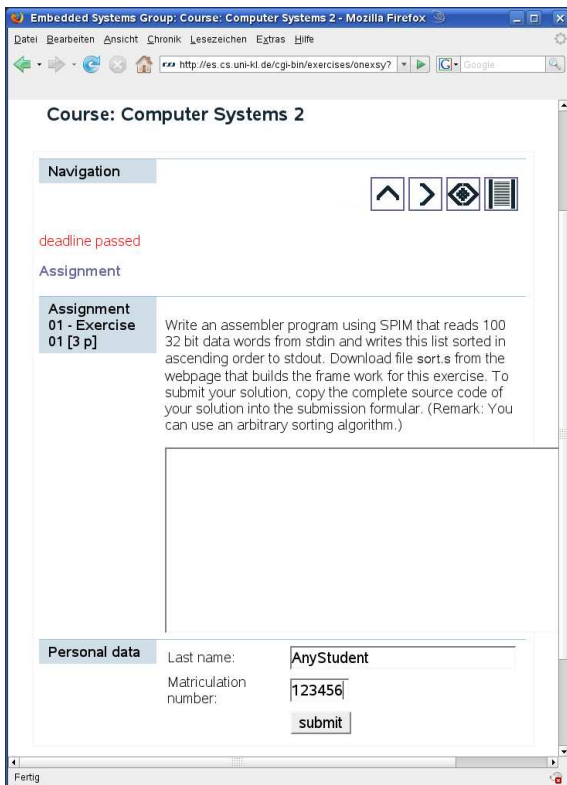


Figure 3: The students select an exercise and get the assignment. After having solved the exercise, the student enter their solutions into a submission form.

the selected exercise, yet, the program is executed and a task is created. The information about the task is stored in the student's directory that is located on the server (see Figure 1). So he can go offline, work out the solution for the given task and whenever he logs in again, he can submit the solution of the previously stored exercise. After the correct solution has been submitted, or if the number of trials has been exceeded and no correct solution has been submitted, the generator program is called again and a new task is created. Independent of the result, the sample solution for the old task is given. Hence, we give the students the opportunity to check and to compare their solutions with a sample solution. Furthermore, we offer to train a specific type of exercise arbitrarily often without having any effort for the correction of these solutions.

2.2 PROGRAMS

The solution programs and the generator programs are independent of OnExSy's user interface. Hence, OnExSy is a flexible system with a solid user interface that can be extended by arbitrary solution checker and generator programs. The whole system can be ex-

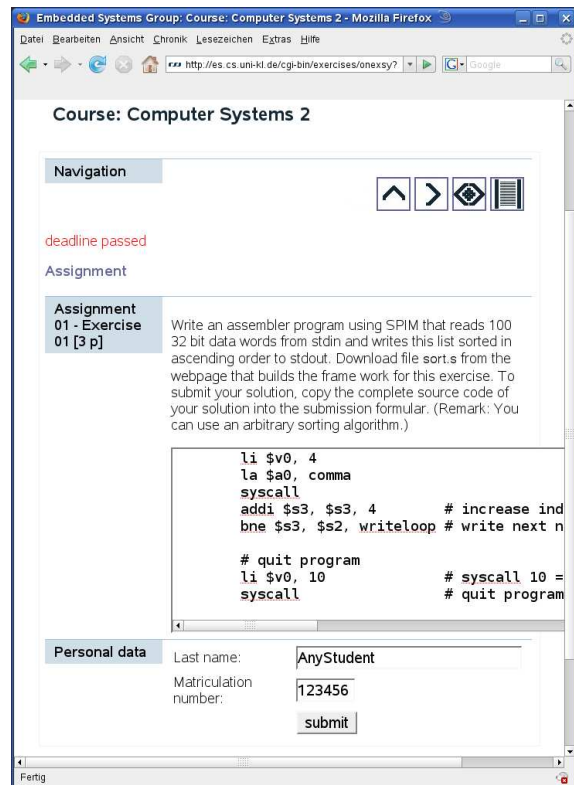


Figure 4: Students can also copy complete source code into the submission form.

tended to any kind of exercise that can be checked by a computer. Clearly, every exercise that has a canonical solution can be automatically checked. However, even though many correct solutions could exist, it is still possible to check the correctness of these exercises by means of formal verification tools. As an example, we verified digital hardware circuits for computer arithmetic that have been submitted by the students.

Note that 'verification' does not mean 'testing'. Instead, verification of a program checks that the program behaves according to the specification (given in the exercise) for all possible input traces. Using state-of-the-art verification tools based on symbolic model checking (Burch et al., 1990), it is possible to explore very large state spaces within some seconds, so that an immediate feedback to the student can be given (the circuits and programs required by the exercises are small enough to allow this immediate response). In our own research, we developed a synthesis and verification system (AVEREST, 2008) whose core techniques are now also used for teaching. The use of formal verification is a key feature of OnExSy that enables both automatically checking the correctness of solutions and the generation of counterexamples in case a solution should be wrong.

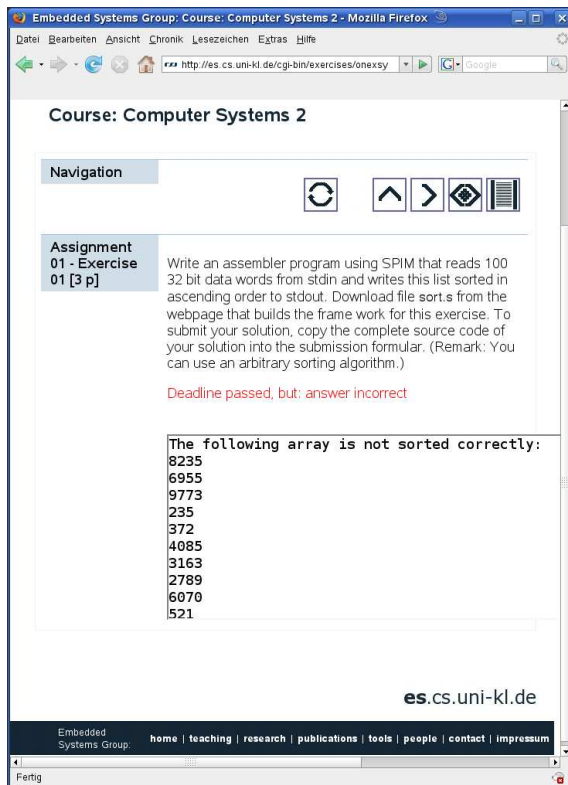


Figure 5: The solution is checked and the result is returned. In this case, the submitted solution was not correct and the student gets a counterexample. Remark: if the answer is correct and given within the determined deadline, the achieved points are granted.

A positive side effect of applying solution checking programs to the students' solutions is the equal quality level for each student. While the correction by persons may naturally differ between different tutors, OnExSy provides a fair treatment for all students.

2.3 ONLINE TRAINING TOOLS

OnExSy is complemented with further training tools that are not used to correct solutions to exercises, but to allow our students to do some complex calculations on their own. For example, these tools can be used to calculate some normal forms like BDDs of a given propositional formula (see Figure 6), to compute minimal disjunctive normal forms, to minimize finite state machines (see Figure 7), etc.

The purpose of these tools is that the students can explore themselves many special cases that can not all be discussed within the course or the tutorial. Moreover, these tools are an important means for the final preparations for the exams when the lecture time is over.

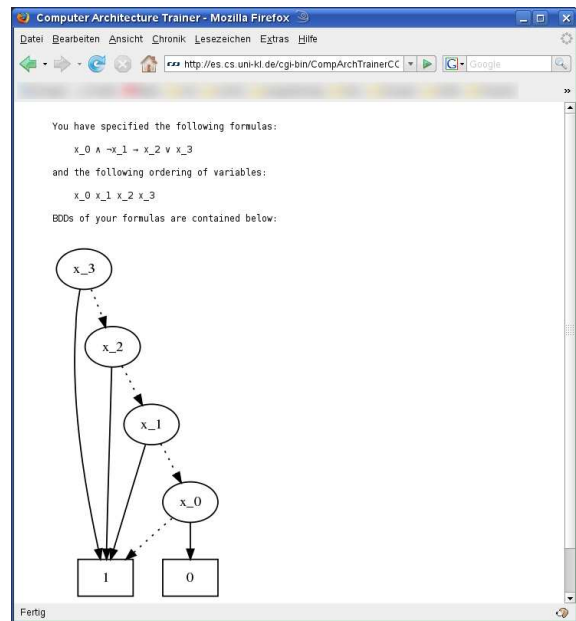


Figure 6: The Online Training Tool generated a BDD out of an user-defined propositional formula.

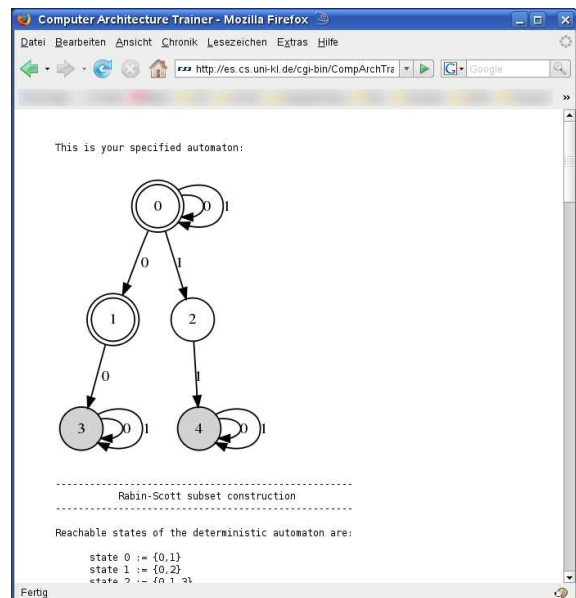


Figure 7: The Online Training Tool generated an automaton out of an user-defined description.

3 CONCLUSIONS

We applied our OnExSy to the complete exercises of 'Computer Systems 1 & 2'. The correctness of all submitted solutions have been automatically checked by the OnExSy alone without any further effort of our tutors. After eliminating some initial software prob-

lems, the feedback of the students was mainly positive. Currently, we are using the OnExSy for two further lectures that are also offered by our group. Due to the topics of these lectures, more complex programs are required for checking the correctness of the solutions.

We experienced another interesting effect: First, we did not limit the number of trials for an exercise. Due to this fact, some students tried to *guess a solution* instead of trying to solve the problem. One of our students even tried 417 solutions for one exercise. Obviously, he wrote a program that simply enumerated all possibilities. The short time intervals of his submissions noted by our system confirmed this suspicion. For this reason, a limitation of the number of trials became necessary. It forces students to think about a solution instead of starting a brute force submission of guesses.

The development of the programs for generating individual exercises and for checking the correctness of the solutions required a lot of work. The initial effort of creating these programs often reaches and sometimes exceeds the effort that would be necessary to correct the exercises manually. However, this effort will be clearly amortized in the near future, and also leads to a better service for our students.

We intend to improve OnExSy by further features like tools to generate more statistic analysis of the submitted solutions. This gives our students individually information about their success, and moreover helps us and our tutors to identify particular problems more quickly than before. This allows us to immediately react in the lecture and the tutorials to clarify potential misunderstandings.

REFERENCES

- APA08 (2008). Apache http server project. <http://www.apache.org/index.html>.
- AVEREST (2008). Averest - a framework for the specification, verification, and implementation of reactive systems. <http://www.averest.org/>.
- Burch, J., Clarke, E., Mcmillan, K., Dill, D., and Hwang, L. (1990). Symbolic model checking: 10^{20} states and beyond. In *Symposium on Logic in Computer Science*, pages 1–33. IEEE Computer Society Press.
- CalMaëth (2008). Calmaëth. <http://CalMaëth.maths.uwa.edu.au>.
- Hades (2007). Hades - interactive simulation framework. <http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/index.html>.
- im-c.de (2008). IMC AG: Learning management system. <http://www.im-c.de/>.

JADE (2003). JADE: Java decision diagram package. <http://www.informatik.uni-bremen.de/agra/doc/software/manual/index.html>.

Ross, K. (2005). Academic dishonesty and the internet. *Communications of the ACM*, 48(10):29–31.