Kong, S.C., Ogata, H., Arnseth, H.C., Chan, C.K.K., Hirashima, T., Klett, F., Lee, J.H.M., Liu, C.C., Looi, C.K., Milrad, M., Mitrovic, A., Nakabayashi, K., Wong, S.L., Yang, S.J.H. (eds.) (2009). *Proceedings of the 17th International Conference on Computers in Education [CDROM]*. Hong Kong: Asia-Pacific Society for Computers in Education.

Evaluation of a Constraint-Based Homework Assistance System for Logic Programming

Nguyen-Thinh LE[a], Wolfgang MENZEL[a], Niels PINKWART[b]
[a]*Department of Informatics, University of Hamburg, Germany*
[b]*Department of Informatics, Clausthal University of Technology, Germany*
{le,menzel}@informatik.uni-hamburg.de
niels.pinkwart@tu-clausthal.de

Abstract: In this paper, we report on the learning benefits of the system INCOM, a constraint-based tutoring system which assists students doing their homework assignments in logic programming. The system has been evaluated with 35 students as part of a logic programming course. The evaluation indicated that the students using INCOM have improved their programming skills significantly after using the system ($p<0.01$, $\alpha = 0.05$) and the students who used INCOM outperformed students in a control condition with an effect size of Cohen's d=0.23.

Keywords: Intelligent tutoring systems, evaluation, constraint-based approach.

Introduction

We have developed a constraint-based system (INCOM) which assists students doing their homework assignments in logic programming. The diagnostic capability of the system has been already confirmed in prior research where INCOM was able to identify the intention underlying 87.9% of students' logic programs and diagnosed 92.7% of the errors in these programs correctly [3].

In this paper, we report on the learning benefits of this system. First, we outline briefly the learning scenario and the modeling approach of INCOM. Then, we describe the design of the evaluation. The effectiveness of the system is shown by three types of data: empirical statistics, video data and subjective attitudes of the evaluation participants.

1. INCOM: A Web-based Homework Assistance System For Logic Programming

The system INCOM is intended to help students of a beginner course on logic programming. It is meant to coach students individually as they solve their homework assignments to better prepare them for subsequent classroom activities. Upon request, the system informs the student about possible errors occurring in her solution attempts and provides her with correction hints to improve the solutions.

INCOM follows a two-stage coaching strategy. In the initial analysis phase, the system requests the student to derive an adequate signature for the predicate to be implemented from the problem task description. A predicate signature consists of a predicate name, the required argument positions, the corresponding argument types and the calling modes of the required argument positions. If the signature is not appropriate, that is, the number of declared argument positions is not correct, or the specified argument

types and the calling modes do not correspond to the requirements given implicitly by the task statement, the system provides corrective feedback by highlighting keywords in the task statement and advises the student to fully make use of the available information. As soon as a satisfactory predicate signature has been specified, the second (implementation) phase is entered and the student is invited to compose a solution for the given task using pure Prolog[1]. If the student's implementation violates a principle of logic programming (e.g., a variable of a arithmetic test must be instantiated), or does not fulfill requirements of a given problem task, errors are detected and explained to the student.

The knowledge of the system is modeled using weighted constraints. They represent the principles of a domain (of an area of study or certain problem tasks) [5] and are used to check the student's implementation against the semantic requirements of correct solutions [4]. When the student submits a predicate's implementation to the system for diagnosis, the system collects appropriate constraints to check the semantic correctness of the student's implementation. Constraint weights are used to facilitate a comparison of competing error explanations. This allows the system to hypothesize the solution strategy possibly pursued by the student. Furthermore, constraint weights can be used to determine the order in which feedback is presented to the student [3].

2. Experiment

2.1 Settings

Our study aims at answering the research question whether students improve their programming skills after using the system INCOM. The study was conducted with students who were attending a course in logic programming and took place in the computer pool rooms of the Department of Informatics during regular classroom hours, where normally students are expected to demonstrate their homework in the presence of a human tutor. The students were given credits for participating in the study but had the possibility to opt out. The course offered four groups on two different days. We divided each group into two sections: a control condition and an experimental condition. In order to balance the two conditions in terms of students' performance, we summed up the achievement score of the preceding sessions for each student and balanced the two conditions by these scores (i.e. the difference between the total scores of the two conditions was minimal). Students, who came late, were allowed to join the experiment, but were not considered as a participant of the study. After eliminating these cases, the control condition and the experimental condition were balanced in terms of participants (17 and 18 students, respectively).

2.2 Design

A pre-test was administered to both control and experimental group in order to assure initial comparability on the dependent measures. The students were required to complete the pre-test within 10 minutes. After that, they were given five exercise assignments which were collected from the homework and examinations of former years of the same course. The time for this part of experiment session was limited to 60 minutes.

1 Pure Prolog does not support higher-order predicates, among them cuts, disjunctions, if-then-else operators, assert, retract, abolish and other database-altering predicates.

In the experimental group, the participants were asked to read a short tutorial which explains the user interface (Figure 1 and 2) of the system INCOM. The participants of the control group were provided with the normal environment consisting of editor and interpreter (in our case SWI-Prolog). Both the system INCOM and the Prolog interpreter were started before the experiment session begun.

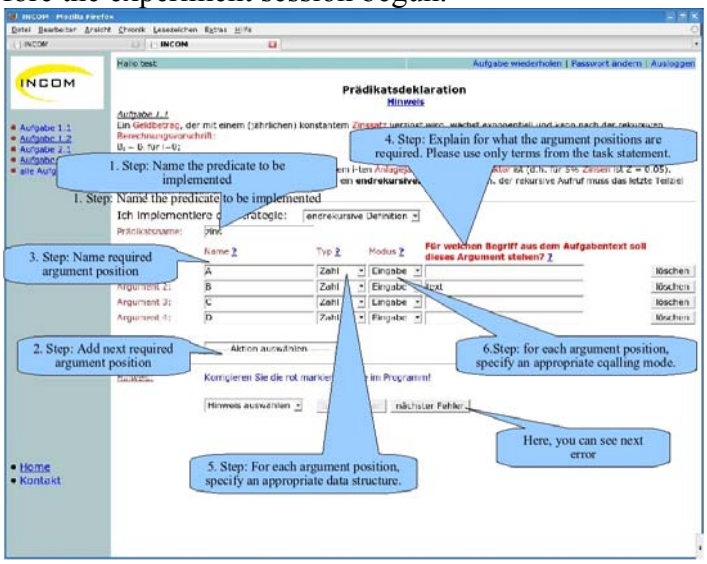

Figure1: Interface for specifying a predicate's signature

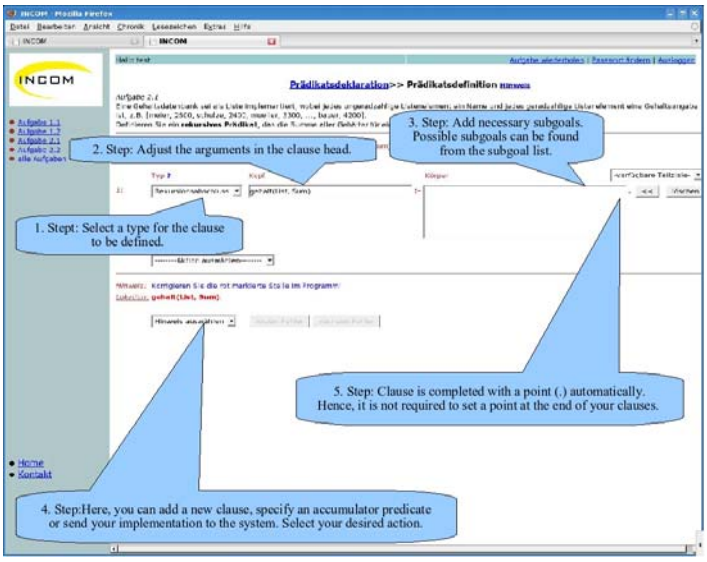

Figure2: Interface for implementing a predicate

A post-test (to be completed within 10 minutes) was given to the participants after completion of the experiment session. Pre- and post-test were made comparable by a counter-balanced design of the test items. Specifically, we developed two test versions: Test A and Test B. We assigned Test A as pre-test and Test B as post-test to 50% of the participants of the control group, and the rest of the control group had Test B as pre-test and Test A as post-test. The same was done for the experimental group. At the end, the students of the experimental group were given a questionnaire in order to express their opinions about the usefulness of the system. The participants of the other group were given questions about how difficult they found the test and experiment exercises.

Students' solutions of tests and of experiment exercises were collected to be used for analysis. In addition, students' responses to the questionnaire were used as subjective data.

The whole process including pre-test, experiment session, post-test and questionnaire was limited to 90 minutes. Samples of test and experiment exercises are shown in the Appendix.

2.3 Results

2.3.1 Learning Gains

Each test (either pre or post) could be scored maximally with nine points. Comparing the results of the pre-tests of the experimental group and the control group, we did not find a statistically significant difference ($p=0.27$ for Day 1, $p=0.14$ for Day 2 with a significance level of 0.05). Therefore, we can assume that the groups were fairly balanced.

In order to determine whether the system is effective in improving the programming skills of the students, learning gains were calculated as the difference between post-test and pre-test scores. Table 1 shows the development of test scores from pre-test to post-test. We distinguish the result of Day 1 from the one of Day 2 in order to identify parameters which might have influenced the learning gains. The third column represents the learning gains of each group and of each experiment day. The last column shows whether the difference between pre- and post-test is statistically significant at the 5% level. The table indicates that only the experimental group of Day 2 made significant learning gains none of other groups did, though clearly the learning gains of the experimental group of Day 1 were higher than for any control group. A possible explanation for this is that during the session on Day 1 a technical problem caused unusually long system's response (more than 60 seconds for each diagnosis). Therefore, a smooth interaction with the system was almost impossible. This problem did not occur during the second session on Day 2 where performance was back to normal. In overall, the experimental group has made an significant improvement ($p<0.01$).

Table 1: Learning gains of two groups

Group	Day	Learning gains (s.d.)	Significant (p value)
Control	1	1.00 (3.05)	No (0.38)
Control	2	0.50 (2.39)	No (0.55)
Control	1+2	0.74 (2.64)	No (0.27)
Experimental	1	1.14 (2.06)	No (0.19)
Experimental	2	1.32 (1.74)	Yes (0.03)
Experimental	1+2	1.25 (1.81)	Yes (0.01)

2.3.2 Difference Between The Experimental Group and The Control Group

The indicator of learning gains clearly shows that the experimental group of the study on Day 2 improved. However, the control group also has made progress on average as well, though this was statistically not significant. In order to compare the improvement between the experimental group and the control group, we computed the effect size as a standardized mean difference between the two groups. Table 2 shows that the effect size of learning gains between the experimental group and the the control group has a medium value ($d=0.3923$) for the Day 2 session. According to [6], this effect size indicates an educational significance, i.e. something was learned due to the use of INCOM as

compared to a standard programming environment. For the Day 1 session, the learning effect was very small (probably, due to the long response time of the system).

Table 2: Effect size of learning gains

Day	Control group Learning gains: means (s.d)	Experimental group Learning gains: means (s.d.)	Effect size Cohen's d[2]
1	1.00 (3.0)	1.14 (2.06)	0.06
2	0.50 (2.39)	1.32 (1.74)	0.39
Overall	0.74 (2.64)	1.25 (1.81)	0.23

2.3.3 Student's Attitude

Based on a questionnaire, we identified the attitude of participants towards the system. It contained the following questions:

1. How precise is the information about the location of the error?
2. How comprehensible are the system hints?
3. Did system hints motivate you to continue working on your solutions?
4. Did the system help you to find a solution for a problem task?
5. Would you be in a position to solve other problem tasks of the same type as the experiment exercises?
6. Would you use this system to do your homework?

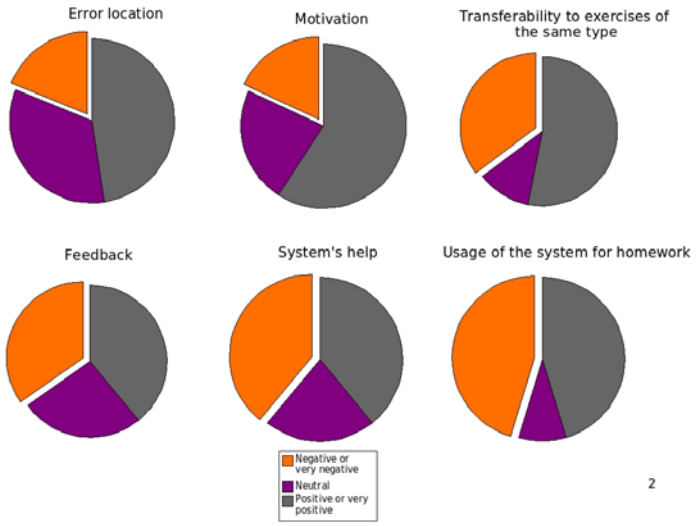

Figure 3: Student's ratings on questionnaire

For each question, participants were asked to provide their opinion on a scale between 1 (very negative) and 5 (very positive). Figure 3 shows the results of the questionnaire. The two diagrams on the most left side shows the ratings for questions 1 and 2 and that most students were pleased with the diagnostic information provided by the system. The diagrams in the middle of the figure represent the opinion of the students on questions 3 and 4 and reveal that the system has motivated most of the students to practice

2 Cohen's d=(gain(experimental)-gain(control))/sqrt((variance(experimental)+variance(control))/2)
Cohen's "Rules-of-Thumb": small effect (d=0.2), medium effect (d=0.5), large effect (d>0.8)

programming but that a remarkable part of participants (39%) did hesitate to confirm that the system could help them to find a solution for a problem task. Despite this cautious self-assessment, the statistic results showed that at least some of them have made moderate learning gains (namely in the experimental group on Day 2). The ratings for questions 5 and 6 on the most right side indicate that more than half of the participants were confident that they would be able to solve similar problem tasks, and almost half of the participants (45%) would like to use the system for receiving help doing homework in logic programming. Some participants commented on the questionnaire that they would have liked to have more time to become more familiar with the user interface of the system and be able to solve sample exercises.

2.3.4 Analysis of Visual Video Data

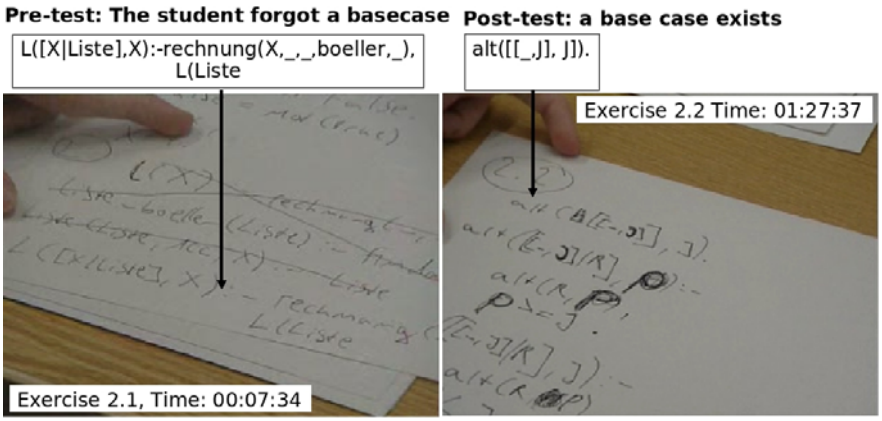

Figure 4: Participant 1 has learned to specify a base case

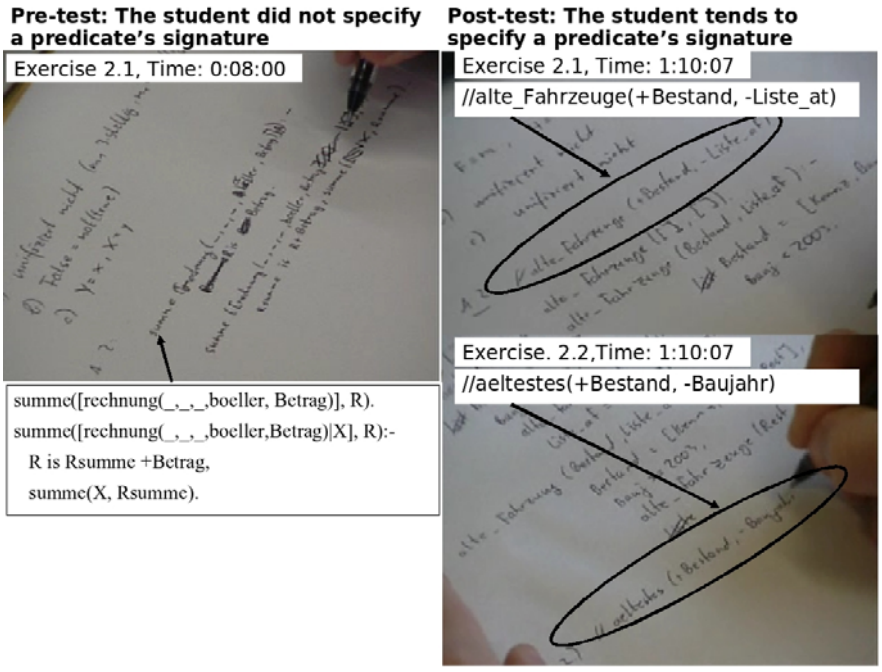

Figure 5: Participant 2 starts specifying a signature

In addition to the benchmark evaluation, where we compared the tutoring system against traditional instruction, we invited students to participate in a separate experiment where the interactions of each participant with the system were recorded by a video camera. This

experiment was carried out to find out the features which lead to the learning gains, a goal which cannot be achieved by a benchmark evaluation [7]. Four students took part in this experiment. They had to go through the same sequence of pre-test, experiment exercises with INCOM and post-test as did the participants of the empirical study. By analyzing the video data we were able to identify the concepts which the participants acquired. For example, participant 1 learned that a recursive implementation requires at least a base case (Figure 4); Participant 2 started to specify the signature before implementing a solution (Figure 5), a behaviour which was not observed during the pre-test. Although we could not find an improvement in the post-test of participant 3, we found evidence in the videos that he was able to correct six of seven shortcomings in his solution attempts after requesting feedback from the system. Figure 6 illustrates two erroneous cases where the system gave feedback and participant 3 corrected his shortcomings. Participant 4 has not had made any improvement because his performance was already very good.

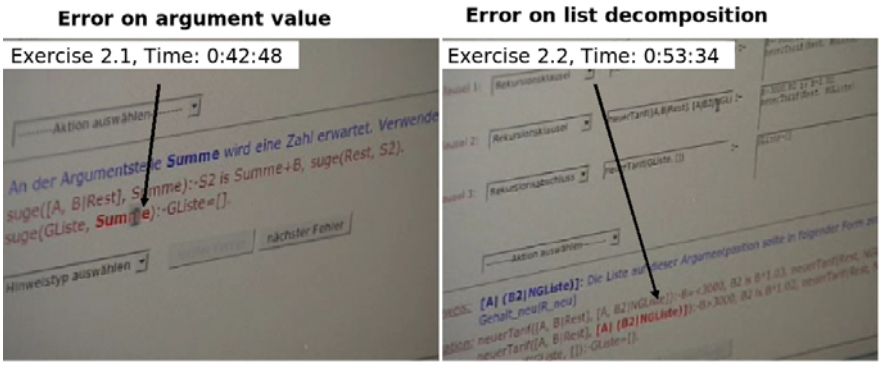

Figure 6: Participant 3 corrected errors identified by the system

3 Discussion and Conclusion

We have evaluated a constraint-based tutoring system which assists students doing their homework in logic programming. In this evaluation, we used three types of data: pre/post-test statistics, subjective students' attitudes towards the system and video sequences which logged the individual problem solving process of four students. The pre/post-test data show that the system has helped students to improve their skills in logic programming. The statistical result (on Day 2) indicates that the students who used the system outperformed the group without using it by 0.39 standard deviation (the mean of the experimental group is at 79% of the control group). Given the degree of difficulty of the problems to be solved and the relatively short exposure time of 60 minutes this is a remarkable result. Other studies with different systems and within different tutoring domains reported higher learning gains after longer tutoring periods (Table 3).

Table 3: Learning benefits of different tutoring systems

System	Period	Effect size
INCOM	60 minutes	0.39 s.d.
LISP tutor [2]	1 semester	1.00 s.d.
SQL tutor [1]	120 minutes	0.65 s.d.

We did, however, decide deliberately not to carry out a truly long-term study, because it makes extremly difficult to factor out properly the various contributions which might have influenced the learning process.

In addition to the statistical results, the responses to a questionnaire indicate that the system has provided useful diagnostic information, and that students were motivated by this feedback to continue working on their individual solution. By analyzing video sequences we identified specific features which have been learned by each subject, e.g. constructing a base case, or a predicate's signature.

Similar to a classroom session where a human tutor would expect that students achieve learning gains within 90 minutes, the study has also demonstrated that students who used INCOM for the same period did improve their programming skills. The study, however, was not able to identify the most relevant factors contributing to this improvement. It could be accounted to the feedback of the system, to the guidance of the user interface which forces students to work systematically or even to the recall of previously learned concepts by being exposed to the terminology of the user interface and the system feedback. Since many students expressed their desire to use such a system for regular homework assignments, we started to deploy more problem tasks into the system in order to be able to fully integrate it into the logic programming courses.

Acknowledgements

We would like to thank our colleagues Matthias Kerzel and Lidia Khmylko for scoring the pre and post test.

References

[1] Barrow, Devon et al (2008). Assessing the impact of positive feedback in constraint-based tutors. In Woolf, B. et al, *ITS 2008*, LNCS 5091, 250-259, Springer Berlin/Heidelberg.

[2] Koedinger, K.R. et al (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8, 30-43, 1997.

[3] Le, Nguyen-Thinh & Menzel, Wolfgang (2009). Using weighted constraints to diagnose errors in logic programming - The case of an ill-defined domain. *International Journal of AI in Education - Special Issue on "ill-defined domains",* in press.

[4] Mitrovic, Antonija et al (2001). Constraint-based tutors: a success story. In L. Monostori and J. Vancza, *Proc. of the 14th Int. Conf. on Industrial Eng. App. of AI and Expert Systems,* 931-940, Budapest.

[5] Ohlsson, Stellan (1994). Constraint-based student modelling. In J. E. Greer, G.I. McCalla, *Student Modelling: The Key to Indivi-dualized Knowledge-based Instruction,* 167-189. Berlin.

[6] Wolf, Fredderic Marc (1986). *Meta-analysis: Quantitative Methods for Research Synthesis*. Beverly Hills, CA: Sage.

[7] Woolf, Beverly Park (2009). *Building intelligent interactive tutors.* p. 191. Morgan Kaufmann.

Appendix

An exercise of pre/post-test

A list represents the size of the audience for a series of TV programs. Each list element contains a sublist with information about the TV station, the program title and the size of the audience (in thousands). The list is ordered in descending order according to the size of the audience and is implemented as an argument of the predicate audience/1 in the database of the Prolog system: audience([[TV1, Pro1, 5300], [TV2, Pro2, 4200],...,[TVn, ProN, 3000]]).

1) Define a predicate which builds a new list of programs for a particular TV station. Please notice, that the original order should be kept and the operator *not/1* is provided to negate a subgoal.
2) Define a predicate which builds a new list which contains the N best programs. In the new list, again, each element should keep all information: TV station name, program title and size of the audience.

An exercise from the experiment session (Time limit 35 minutes)

Given a constant yearly interest rate, a sum of money increases exponentially and can be calculated according to: $Sum_i = Sum_i$ if i=0; $Sum_i = (1+Z)*Sum_i - 1$ otherwise. Define three different predicates to calculate the available amount of money, after N years of investment using a) naive recursion, b) tail recursion and c) no recursion at all.