Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education. India: Asia-Pacific Society for Computers in Education

Hybrid ITS for DFA Construction Problems

Darshan K. M.[a] **& Viraj KUMAR**[b*]
[a]*Department of Information Science and Engineering, PES Institute of Technology, India*
[b]*Department of Computer Science and Engineering, PES University, India*
*viraj.kumar@pes.edu

Abstract: Intelligent Tutoring Systems (ITS) that match or exceed human tutoring are often the product of expensive research efforts in specific domains, and the lack of high-quality ITS in other domains is partly explained by this high creation cost. In this paper, we propose a *hybrid* ITS, where human instructors perform two critical tasks within an otherwise automated system: (1) giving learners specific types of feedback, and (2) scaffolding students' reasoning. We argue that such a hybrid ITS can be cost-effective when the pool of learners is too small to justify the cost of creating an ITS. To illustrate these arguments concretely, we have implemented a hybrid ITS for a component of the undergraduate Computer Science curriculum: the construction of DFA (deterministic finite automata).

Keywords: ITS, scaffolding, semantic feedback, deterministic finite automata (DFA), JFLAP.

1. Introduction and Related Work

Intelligent Tutoring Systems (ITS) typically observe students executing selected tasks, compare their performance against a "gold standard", and provide them with personalized feedback that "brings their performance closer to the gold standard" (VanLehn, 2015). The intelligence of an ITS lies in its ability to meaningfully compare learners' attempts against the gold standard and to provide feedback that helps learners make adjustments. Effective ITS are the outcome of expensive research efforts, and have been developed for domains such as mathematics (Koedinger *et al.*, 1997; Arroyo *et al.*, 2004; Ritter *et al.*, 2007), SQL (Mitrovic, 1998), and physics (Gertner and VanLehn, 2000), where large student numbers justify these costs. In this paper, we consider a relatively esoteric domain in the Computer Science undergraduate curriculum (ACM/IEEECS, 2013): the construction of deterministic finite automata (DFA). In this paper, we demonstrate a low-cost *hybrid* ITS that "amplifies" the skills of instructors (Toyama, 2010) by identifying two specific tasks for faculty/teaching assistants to perform: (1) provide *scaffolding* (defined by Chi *et al.* (2001) as "cooperative execution or coordination by the tutor and the student […] in a way that allows the student to take an increasingly larger burden in performing the skill"; and (2) provide *semantic feedback* (Le, 2016).

For DFA construction problems, JFLAP (Rodger and Finley, 2006) provides *syntax feedback* and indicates whether the DFA accepts or rejects given inputs. D'Antoni *et al.* (2015) have investigated the efficacy of automatically generated semantic feedback, and their findings are promising: learners report that these hints are useful, and providing hints lowers task completion times. However, learners occasionally find these hints "confusing" and cannot translate them into specific corrective actions.

2. Specific roles of human instructors in a hybrid ITS

We illustrate key roles that human instructors can play in helping learners construct DFA using two examples that were posed to a set of 70 undergraduates. Let P_1 be the problem: Construct a DFA accepting binary strings with an even number of 1's, whose integer value is divisible by 3. Let P_2 be the problem: Construct a DFA accepting binary strings where every 1 is followed by two consecutive 0's.

Role 1: Provide scaffolding. Several learners were unable to recognize that the condition in P_1 is the conjunction of two simpler properties: "binary strings with an even number of 1's" and "binary strings whose integer value is divisible by 3". (Poor language skills could contribute to this struggle.) In our hybrid ITS, the problem is represented internally using a functional representation (Shenoy *et al.*, 2016) as shown in Figure 1. In this tree representation, each internal node is a function representing a property, and leaves are either constants or the input string *x*. In this case, the two sub-problems correspond to the left and right sub-trees below the root node. For such problems, our system can provide automatic scaffolding by splitting the problem into two reasonable sub-problems. For a problem such as P_2, no such reasonabl ding.

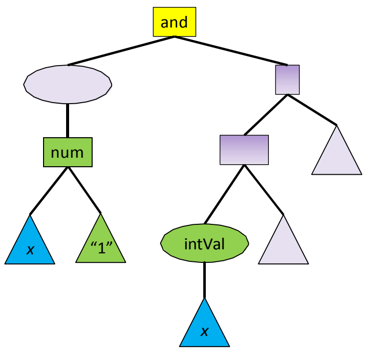


Figure 1. A functional representation of problem P_1.

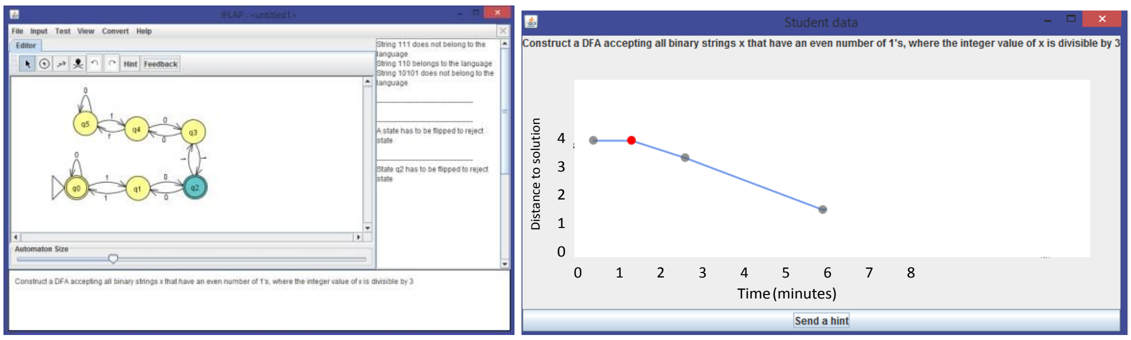
Role 2: Provide semantic feedback. Some learners found the expression "every 1 is followed by two consecutive 0's" in P_2 ambiguous. When probed, a few stated that they were unsure whether every 1 was *immediately* or *eventually* followed by two consecutive zeroes. Others were uncertain whether each 1 was followed by *at least* or *exactly* two zeroes. Clearly, the question *is* ambiguous and it should have been phrased better. However, bearing in mind that this investigation was conducted in a linguistically diverse country where neither learners nor instructors are necessarily fluent in English (which, in any case, can be ambiguous), there is always potential for such confusion to arise. Learners with poor English skills may be embarrassed to seek help if they assume that the question is unambiguous, and they may blame themselves for failing to parse it accurately.

In our hybrid ITS, learners can seek automated answers to queries of the form: "Should the DFA accept the string *x*?", where *x* is specified by the learner. For P_2, learners could resolve these ambiguities by testing the strings 1100 (the first 1 is eventually but not immediately followed by 00) and 1000 (the 1 is followed by at least but not exactly two 0's). Learners can state that they disagree with (or don't understand) answers to certain queries, and these are flagged for (human) responses, which are necessarily delayed but can still be useful. Our hybrid ITS helps instructors in formulating responses as follows: if a query string *x* is *not* accepted by the DFA, we identify the problem that is syntactically closest (Alur *et al.*, 2013) to the given question for which *x is* accepted, and suggest this problem as a potential misunderstanding to the instructor. (Such feedback can also be given to the learner.) If the question is truly ambiguous, the instructor should naturally issue appropriate clarifications. Otherwise, instructors may need to engage learners in dialog to help them interpret the results accurately.

Our system also allows learners to ask the question "Is my DFA correct?" and receive two types of automated replies. D'Antoni *et al*. (2015) suggest that some learners prefer replies in the form of counter-examples (which are perhaps easy to translate into corrective action), whereas others prefer more sophisticated textual hints. Our hybrid ITS only uses counter-examples at present. These can be of two types: strings that the solution DFA accepts but the learner DFA does not, and strings that the solution DFA does not accept but the learner DFA does. We generate shortest possible counter-examples of both types, if possible. Once again, it may be necessary for human instructors to help learners interpret such responses. For instance, one learner constructed a DFA for P_2 that rejected the empty string, and hence received this string as a counter-example. The empty string satisfies P_2's condition vacuously, and the learner needed help in understanding this subtle point.

Following D'Antoni *et al*. (2015), we also automatically find the smallest *edit set* of changes that will transform the learner's DFA into a DFA that is equivalent to the solution. If a learner requests additional help, one of these corrective steps is reported. In the first request for feedback, we withhold some details. For the example shown in Figure 2 (left), we merely state that "a state" needs to be made a rejecting state. Only if additional feedback is requested do we identify this target state as q_2. If a learner asks for further feedback, a request is sent to instructors accompanied by data capturing the learner's prior interaction with the system for this task. This is presented to instructors as a *timeline* (Figure 2, right) and provides *context* for the learner's request for help. The *y*-axis represents the edit

distance to the solution (if known). Each feedback request by the learner is represented as a clickable dot on this graph (a red dot indicates a request for manual feedback). Instructors can click on individual dots ar
usefu
learne
be equ


JFLAP : <untitled1>

String 111 does not belong to the language
String 110 belongs to the language
String 10101 does not belong to the language

A state has to be flipped to reject state

State q2 has to be flipped to reject state

Construct a DFA accepting all binary strings x that have an even number of 1's, where the integer value of x is divisible by 3

Student data

Construct a DFA accepting all binary strings x that have an even number of 1's, where the integer value of x is divisible by 3

Figure 2. (Left) A screenshot of our hybrid ITS, implemented as a JFLAP extension. Auto-generated and instructor assistance appears in the scrollable pane to the right of the drawing area. (Right) A learner's timeline for the problem P_1.

3. Conclusions

Our hybrid ITS uses existing techniques to provide learners with automated assistance as a "first line of defense", and also gives instructors automated suggestions and visualizations that can help identify why the learner is struggling. Our system can recycle manual feedback in limited instances, and we are exploring ways to enhance its capabilities. This hybrid ITS will be evaluated in the upcoming semester.

References

ACM/IEEECS (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, ACM, New York, NY, 2013.

Alur, R., D'Antoni, L., Gulwani, S., Kini, D. and Viswanathan, M. (2013). Automated Grading of DFA Constructions, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 1976-1982.

Arroyo, I., Beal, C., Murray, T., Walles, R., and Woolf, B. P. (2004). Web-based intelligent multimedia tutoring for high stakes achievement tests. In International Conference on Intelligent Tutoring Systems, 468–477.

Chi, M. T. H., Siler, S., Jeong, H., Yamauchi, T., and Hausmann, R. G. (2001). Learning from human tutoring. Cognitive Science, 25, 471–533.

D'Antoni, L., Kini, D., Alur, R., Gulwani, S., Viswanathan, M. and Hartmann, B. (2015). How Can Automatic Feedback Help Students Construct Automata? ACM Transactions on Computer-Human Interaction, 22(2).

Gertner, A. S., and VanLehn, K. (2000). Andes: A coached problem solving environment for physics. In International conference on intelligent tutoring systems, 133–142.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education, 8(1), 30–43.

Le, Nguyen-Thinh (2016). A Classification of Adaptive Feedback in Educational Systems for Programming. Systems, 4(2), 22.

Mitrovic, A. (1998). A knowledge-based teaching system for SQL. In Proc. of ED-MEDIA, vol. 98, 1027–1032.

Ritter, S., Anderson, J. R., Koedinger, K. R., and Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. Psychonomic bulletin & review, 14(2), 249-255.

Rodger, S. H. and Finley, T. W. (2006). JFLAP – An Interactive Formal Languages and Automata Package, Jones and Bartlett, Sudbury, MA, 2006.

Shenoy, V., Aparanji, U., K., Sripradha and Kumar, V. (2016). Generating DFA Construction Problems Automatically, Proc. of the 4th Intl. Conf. on Learning and Teaching in Computing and Engineering, 32-37.

Toyama, K. (2010). Can technology end poverty? Boston Review, 36(5), 12-18, 28-29, www.bostonreview.net/forum/can-technology-end-poverty.

VanLehn, K. (2015). Regulative Loops, Step Loops and Task Loops. International Journal of Artificial Intelligence in Education, 26, 107–112.