The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

Improving Formative Feedback on Argument Graphs

Nancy L. Green, Kevin Walker, Somya Agarwal

Department of Computer Science
University of North Carolina Greensboro
Greensboro, NC 27402 USA
nlgreen@uncg.edu

Abstract

A prototype educational argument modeling system was previously developed for an undergraduate genetics course. The system provides formative feedback by comparing a student's argument graph structure to solutions generated internally by the system. This paper describes improvements to the generation and delivery of feedback and results of small study of effectiveness.

Introduction

Science educators have long recognized the need to improve students' argumentation skills. The National Science Teachers Association suggests that one way to do so in biology courses is "to engage students in scientific argumentation as part of the teaching and learning of biology" (Sampson and Schleigh, 2013). In previous work, we described a prototype educational argument modeling system for teaching undergraduate students to construct graphical representations of arguments about genetic disorders (Green 2017). The system's drag-and-drop interface for constructing argument graphs provides two types of information: data about a clinical case (possible premises of arguments) and general information about genetic disorders (possible warrants (Toulmin 1998)). Atomic arguments, consisting of one or more premises, a warrant, and conclusion, can be composed into a tree structure. A unique feature of the system is that it does not require instructors to provide pre-made arguments. Instead, the system generates arguments from an underlying causal domain model and abstract reasoning patterns, such as *Inference to the Best Explanation* (IBE), for this domain. These arguments are used as a knowledge-source by the system for generating feedback.

Our system provides *formative feedback*, "information communicated to the learner that is intended to modify his or her thinking or behavior for the purpose of improved learning" (Shute, 2008, p. 154). A survey of automatic feedback techniques used in educational argument modeling systems is given in (Scheuer et al. 2012). The approach implemented in our system gives feedback not just on the structure of the argument graph, but also on its content. Unlike other approaches surveyed in that article that deliver content feedback, our approach does not require an expert to manually encode arguments.

A limitation of the previous version of the system was that when the system generated more than one argument as a solution, very simple heuristics were used to decide which of the system's arguments was most similar to the student's argument and then that one was used as the basis for giving feedback. As a result, the student might be led to believe mistakenly that certain correct subarguments in her solution were in error if they did not match corresponding parts of the system's selected argument. Another limitation was that only one feedback message was displayed after each attempt, although multiple errors might have been detected by the system. On the other hand, if feedback on all detected errors had been given at once, the student might be overwhelmed. Another problem was that the feedback was presented as text apart from the student's argument graph, requiring the student to mentally connect the feedback to the relevant part of her argument graph.

The rest of this paper describes our work to address these limitations. The next section describes an improved method for generation of feedback. After that, we describe how the delivery of feedback was changed. The last section discusses results of a small study of the effectiveness of the system.

Feedback Generation

To generate feedback, the system must compare the student's attempted solution to the system's solution(s). (The system may have generated multiple arguments since different data may be used to generate different arguments for the same conclusion.) For example, to argue for the claim that patient JB has cystic fibrosis, the

system might generate the following series of four "chained" IBE arguments internally (i.e., premise B of Argument 1 is supported by Subargument 2, and premise C of Subargument 2 is supported by Subargument 3, which is based on data D).

Argument 1:
Conclusion: (A) JB has cystic fibrosis, i.e., 2 variant CFTR alleles.
Premise: (B) JB has abnormal CFTR protein.
Warrant: (WAB) Having 2 variant CFTR alleles leads to abnormal CFTR protein.

Subargument 2:
Conclusion: (B) JB has abnormal CFTR protein.
Premise: (C) JB has viscous lung secretions.
Warrant: (WBC) Having abnormal CFTR protein leads to abnormal lung secretions.

Subargument 3:
Conclusion: (C) JB has viscous lung secretions.
Premise: (D) JB has frequent lung infections.
Warrant: (WCD) Having viscous lung secretions leads to frequent lung infections.

Note that the above is a paraphrase of the content, which is not stored as English text, but is derived from nodes and arcs of the causal domain model and a set of accepted reasoning patterns. The causal domain model is constructed by an instructor using an authoring tool. For more information see (Green 2017).

The student's attempted solution is translated by the system into an internal representation based upon information provided by the authoring tool. The internal representation of structure and content of the student's attempted solution is then comparable to the solution(s) generated by the system. As an example, Figure 1 shows outlines of the student's argument (on the left) and the system's two solutions (on the right), where System solution 1 is an outline of the argument shown above. (Warrants appear at right angles to premise-conclusion links.) In the previous version of the system, the feedback generator tried to determine which system solution best matched the student's solution, and then it was used as the sole basis for providing feedback. Due to limitations of that approach we have implemented the following new approach.

First, a metric of the similarity to the student's solution S is computed for each of the system's solutions. We describe the internal representation of an argument graph as a directed hypergraph H = {X, E}, where X is a set of vertices and E is a set of edges. Each edge in E is a subargument of the graph, consisting of

$\{v_{conclusion}, \{v_{w1}..v_{.wn}\}, \{v_{p1}..v_{pm}\}\}$

where $v_{conclusion}$ is the conclusion, $\{v_{w1}..v_{.wn}\}$ is the warrant, and $\{v_{p1}..v_{.pm}\}$ are the premises. (Vertices come from the causal domain model.) For each of the system's solutions, a similarity score (N1 + N2) – N3 is computed. N1 is the number of subarguments in the student's solution that are identical to the content of the conclusion and premises of a subargument in the system's solution. N2 is the number of vertices in the student's solution that are identical to vertices anywhere in the system's solution, and N3 is the number of missing or incorrect vertices in the student's solution compared to the system's solution. This metric was motivated by the observations that often (N1) a student's solution may contain a subargument that is correct except for the warrant, or (N2) at least may contain elements of the system's solution, but (N3) should be penalized for irrelevant or missing elements. For example, in Figure 1 the similarity scores for system's solutions 1 and 2, respectively, are (1 + 5) - 2 = 4 and (1 + 2) - 6 = -3. Thus, solution 1 will be preferred when generating feedback, although solution 2 is structurally identical to the student's.

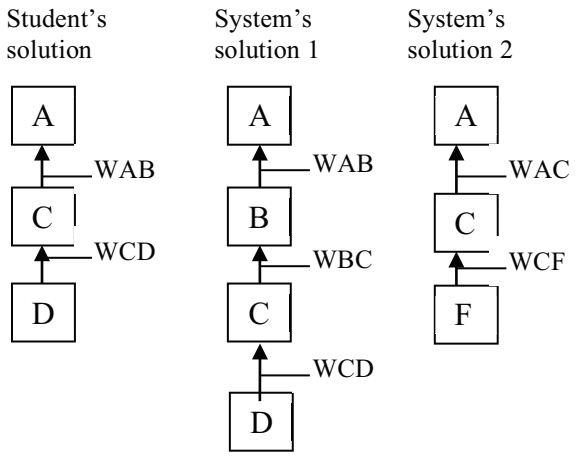

Figure 1: Student's solution compared to system's.

The goal of the next phase of the algorithm (Figure 2) is to detect and annotate errors in each subargument e(S) of the student's solution. The system's solutions A_1 .. A_n are ordered from the most to least similar solution according to the above metric. If e(S) does not match a subargument in the best matching system argument, it still could match an acceptable subargument in one of the other system solutions. Note that the annotations in the algorithm are for internal use and are more concise than the wording used to give feedback to the student.

Feedback Presentation

In the new version of the system, a non-editable copy of the student's solution appears in an adjacent window vertically aligned with the student's solution (Figure 3). Vertices of the argument graph in which errors have been detected are highlighted in color. The student can "mouse

over" any of the vertices to see the feedback message. This approach addresses the previous limitations of providing feedback on only one error at a time, and of not linking the feedback to the relevant part of the student's solution. In the new design, the student can control the timing and quantity of feedback. In addition, the student can change her solution at any time and see the system's analysis of her latest solution.

```
For each e(S) {
For each A_i in A_1 .. A_n {
   Search in A_i for a subargument e(A_i) with the same
   conclusion as the conclusion of e(S) and at least one
   premise matching a premise of e(S).
   If found then {
      compare(e(S), e(A_i)).
      exit inner loop.}
   } % end for each A_i
}

% compare two subarguments S and A
compare(S, A) {
   Mark the premises in S that match premises in A as
        OK and mark the others as NOT OK.
   If there are premises in A that are not in S, mark
        S as HAS MISSING PREMISES.
   Mark the parts of the warrant in S that match parts of
        the warrant in A as OK and mark the others as
        NOT OK.
   If there are parts of the warrant in A that are not in S,
        mark S as HAS MISSING WARRANTS.
   For all premises p of S that are marked NOT OK {
        If p matches a premise of a subargument of A,
           change mark to INDIRECTLY SUPPORTS.
        Elseif p belongs somewhere else in A, change
           mark to IN WRONG PLACE.
        Elseif p belongs somewhere in another argument
           A', change mark to IN WRONG ARGUMENT.
        }
   }
```

Figure 2: Detecting errors in subarguments.

Study and Conclusion

A small study was done to see if the new version of the system was delivering the correct feedback and whether full feedback was more beneficial than just marking an argument component as incorrect. Eleven computer science undergraduate students took part individually. However, data from the first two participants is not reported below since the procedure was changed after their sessions.

After a brief demonstration on how to use the system to create arguments and being given information on the genetic condition used in the problems (cystic fibrosis), the students were given five problems asking them to create arguments for certain hypotheses using the data and warrants presented on the user interface, e.g., an argument for the hypothesis that the patient has cystic fibrosis. The students were told that they could submit their answer to a problem as many times as they wanted, and that they could use the system's feedback to help them arrive at a correct solution. Also they were told that they were free to move on to the next problem at any time. The students were shown the correct solution after they decided to quit working on a problem. Each student was given up to 30 minutes to attempt to answer all the problems.

The students were randomly divided into two groups. In one group (N=5), participants saw full feedback messages; in the other group (N=4) only the messages "Correct" or "Incorrect" were shown. In both cases, incorrect nodes of the argument graph were highlighted and feedback messages could be viewed by "mousing over" a node.

First, to ensure that the system was actually giving correct feedback, system logs of each participant's sessions were used to manually recreate the sessions and verify the correctness of the feedback delivered by the system by "mousing over" each node. Feedback on all nodes of the solution was checked even if the participant had not actually looked at it. It was found that 385 of an expected 401 messages (96%) were correct. The 16 feedback error messages that were expected but not given by the system were due to failure to detect a missing warrant. Also, to check if the similarity metric was being applied as designed, the system's solutions were ordered by the experimenter according to their similarity to the participant's solutions for each problem. The result was compared to the ordering given by the similarity metric, which was found to match.

To evaluate the effectiveness of the feedback, the number of correct solutions found by the participants in each group was compared. The group that received reduced feedback on average answered 1.4 problems correctly, compared to 1.5 for the group that received full feedback. Since that measure failed to show any clear benefit from full feedback, we explored whether full feedback was beneficial in other respects.

We noted that on average, the full feedback group made 10.2 attempts, while the reduced feedback group made only 6.75 attempts. Participants in the reduced group may have made fewer attempts since they were not receiving guidance on why their solution was incorrect. Thus, a possible effect of full feedback was that it was better at guiding the student towards a correct solution. This could be indirectly measured by considering for each problem, the change in the number of error messages over time, and the change in size of the student's solution over time.

To determine if the feedback helped steer the participants towards a correct solution the average change in error messages from the first submission to the final submission for each problem was calculated. Participants who received full feedback were more likely to have a decrease in the number of error messages per problem.

Also, participants who received full feedback had a larger average increase in size of solutions for Questions 1 through 4 than participants who did not. (Question 5 could not be counted since it was answered by only one participant in the group with full feedback due to timing out.) The difference could indicate that participants who did not receive full feedback might see that a specific node in their argument was wrong, and without more detailed feedback, swapped that node out for another keeping the size of their attempt the same.

While we would have liked to perform a larger-scale and more rigorous study with genetics students to evaluate benefits of our approach, it should be noted that the implementation of the improvements and the study were performed without external funding as two Master's student projects. In conclusion, one contribution of this paper is an approach to evaluating argument correctness that takes into account that a student's argument may contain correct subarguments (possibly from multiple different system solutions) even when there are other errors in the graph. Another is our demonstration of what is likely to be a more effective way of presenting and controlling presentation of feedback on complex argument graphs. Finally our results show some promise that such intelligent feedback may motivate a student to work harder to construct an acceptable argument.

References

Green, N.L. 2017. Argumentation Scheme-Based Argument Generation to Support Feedback in Educational Argument Modeling Systems. *International Journal of AI in Education*. September 2017, 27(3): 515–533.

Narciss, S. 2008. Feedback strategies for interactive learning tasks. In J.M. Spector, M.D. Merrill, J. van Merrienboer, and M.P. Driscoll eds. 2012. *Handbook of research on educational communications and technology* (3rd ed., 125-143). New York: Lawrence Erlbaum Associates.

Sampson, V., and S. Schleigh. 2013. *Scientific Argumentation in Biology: 30 Classroom Activities.* National Science Teachers Association Press.

Scheuer, O.; Loll, F.; Pinkwart, N.; and McLaren, B.M. 2012. Automated analysis and feedback techniques to support and teach argumentation: A survey. In N. Pinkwart, B.M. McLaren eds. *Educational Technologies for Teaching Argumentation Skills*. Bentham Science Publishers.

Toulmin, S.E. 1998. *The uses of argument*. Cambridge Cambridge University Press.

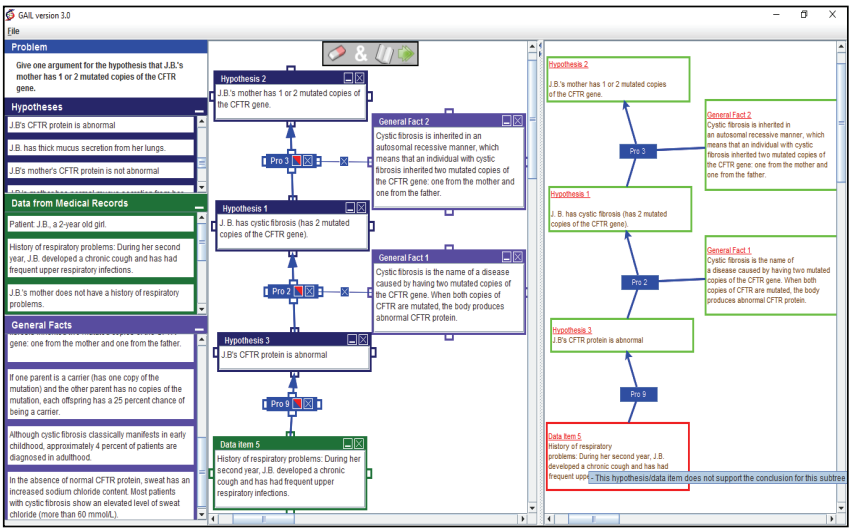

Figure 3: User interface. The far left panel contains a problem, possible hypotheses, data and general facts (warrants) that might be used in the argument. A student has constructed an argument by dragging and dropping some elements into the middle panel and connecting them. The far right panel shows the system's copy of the student's solution with red highlighting to indicate an error. A feedback message is on display after the student "moused over" the highlighted area. The feedback message indicates that Data Item 5 selected by the student does not support the intermediate conclusion (Hypothesis 3) directly above it. The argument structure from Hypothesis 3 upwards is correct.