

Knowledge communication and CBR

Frode Sørmo¹ and Agnar Aamodt²

Department of Computer and Information Science, NTNU
NO-7491 Trondheim
Norway

¹frodeso@idi.ntnu.no

²agnar@idi.ntnu.no

Abstract. In this paper, we describe work in progress to further develop methods from knowledge-intensive Case-Based Reasoning (CBR) in general and the Creek system in particular to address the requirements of Intelligent Tutoring Systems (ITS). ITS systems are often required to solve problems in the domain, but unlike decision support systems, the ability to communicate the process of solving the problem to the student, is as important as finding the right solution.

1 Introduction

One way of describing the difference between Intelligent Tutoring Systems (ITS) and more traditional educational software is the system's ability to solve problems in the domain of discourse on its own without relying on pre-solved problems and explanations. This allows the system to tailor the learning experience to the individual student. Case-Based Reasoning (CBR) has been used in a number of roles in ITS systems, for instance as part of the student modeling process [7, 8] and to assist the student in exercising some kind of operational skill [3, 4, 12].

A student model represents an estimation of what the student knows about the domain, and it forms the basis for tailoring the learning session to the student. Typically, a student model consists of an overlay of an expert model, where it is recorded what parts of the expert model the student is familiar with. In addition, many systems contain a library of common misconceptions in the domain (bug libraries), used to identify the student's likely misconceptions about the domain. Given a student model, it is expected that an ITS system is able to find learning tasks for the student that will challenge his skills without being too difficult for the student to solve. This task could be a specific learning goal, or an exercise designed to train some operational skill, e.g. as in [5] where exercises are formed to train Air Traffic Control operators.

When a task is identified, the system should be able to go through the task with the student, identifying problem areas and provide help when required. In existing systems this is done in different ways. The CATO system [3, 4], for instance, helps

the student form legal arguments by using a domain model to find appropriate earlier cases to draw analogies from. The ELM system [12] uses general knowledge about the domain (LISP programming) with episodic knowledge about past student programs to identify errors and suggest corrections. These systems are not only able to help the student find a solution to the exercise she is trying to solve, but also help the student to identify particular problems she may have in solving the problem. While a non-knowledge-intensive case-based reasoner also is able to find a good solution to a problem, the addition of a general domain model (e.g. the “Factor Hierarchy” in the CATO system) increases the system’s ability to help the learner understand why a past case is relevant. These systems have a knowledge communication strategy [11] that allows the student to learn not only the particular solution to one problem, but is also able to ground it in the theory of the domain.

One way of doing this is to build a cognitive model of how humans solve problems in the domain and use this model in attempting to solve the problem, both from the point of view of the current student (using the student model) and an expert (represented by an expert model). By doing this, the system can identify areas where the student makes mistakes, and suggest steps from the expert’s solution if the student requires help. It is important for the student not only to learn the episodic knowledge represented in the case presented by the expert, but also to learn why her suggestion will not work as well as the expert’s. The case-based reasoner will therefore have to be able to evaluate the student’s solution and explain why it does or does not follow from the observed features of the problem. It is our hypothesis that a knowledge-intensive case-based reasoner that is able to explain to a user why it thinks a solution is a good solution, will be able to communicate not only the episodic knowledge of the case, but also identify and fill holes in the student’s general knowledge.

In this paper, we suggest a design for an ITS system based on the CREEK system [1, 2] where the case-based reasoner is expected to reproduce results and problems from the point of view of both the student and an expert, and be able to communicate the differences. In the next section we set the scene by identifying a set of assumptions and specifying a set of goals for this research. This is followed in section 3 by an outline of the overall framework and ITS-CREEK system architecture. In section 4 we show how the mechanisms of the CREEK CBR system can work in the framework from section 3, and identify some challenges and questions. Section 5 summarizes and concludes the paper.

2 Assumptions and goals

This paper reports on research in progress. Hence, while particular methods and foci may change somewhat as the research progresses, the underlying questions we would like to answer should not.

Our approach to case-based tutoring and learning systems is based on the following three assumptions:

- In intelligent tutoring systems, there is a change in focus from traditional case-based reasoning from being able to solve a problem to being able to communicate how to solve a problem.
- The above task is easier to accomplish if the cognitive problem-solving methods of the student have some similarity to the automated problem-solving methods of the system.
- Although humans use episodic knowledge to solve problems, we also tend to be able to generalize better from fewer cases than a case-based reasoner, and if we are asked, we are often able to provide underlying principles or methods to explain and justify a solution.

Based on these assumptions, and our high-level research goal of studying the integration of case-based and model-based reasoning, our ITS research has three specific goals.

Our first goal is to achieve improved understanding of how knowledge-intensive case-based reasoning methods can provide user-targeted explanatory support in an ITS context. While this has been a part of CBR for a long time both in the roots of case-based reasoning [6] as well as knowledge-intensive CBR systems such as CREEK, we believe that because this is critical in an ITS system. As such, ITS may also prove an excellent area to further this goal for user-interactive systems in a more general sense.

As our second goal, we want to study how well a diagnostic case-based reasoner will be able to use a student model explicitly formed by the student through a modelling tool. This means that the student model may take quite a qualitatively different form than the model created by an expert. A student may even come up with an alternative model explaining a subset of the exercises, and although the student model is not our primary point of interest in this study, it should prove interesting in itself to see how robust the reasoner will prove to different expressions of the same knowledge by expert and student.

Finally, we are interested in seeing how other machine-learning and statistical techniques can be used to generate the necessary domain knowledge required by the reasoner in domains where there might be a large number of cases but little available general domain knowledge. This is often the case in case-based reasoning systems in use today. Because of the similarities between a case-based reasoner in a decision support role and as a diagnostic component in an ITS system, it might be possible to produce systems that are used for both, or to extend existing specialized case-based reasoning systems to have an ITS component. Using statistical or machine learning methods to express the knowledge present in cases in a generalized form more easily communicated to human users, might allow this to be done with a smaller investment in time and money required to create the additional knowledge required by an ITS system.

3 The ITS-CREEK architecture

The focus of the system is to train the student in applying a particular intellectual or operational skill. It might be to diagnose patient cases in some limited medical domain, predict unwanted events during an industrial process, solve specific math problems, or interpret the essential contents of an image. The architecture is, in other words, general, and the domain-dependent part is represented by the knowledge base – combining a general domain model with a set of cases.

In order to be able to focus on the student-diagnostic module of the ITS system, we will have a system where the student himself explicitly models his knowledge about the domain through a modeling tool. This model built by the student is used directly by the system to find appropriate exercises to the student. When an exercise is selected, the system will try to solve this exercise using both the student model created explicitly by the student and the expert model, thus attempting to predict possible problem areas for the student. It is also very important that the system is able to communicate to the student how the expert reasoning is different from the student's and if necessary go through the exercise itself, demonstrating to the student how the problem effectively can be solved. Figure 1 contains an overview of the system architecture.

To realize this architecture, we are using the CreekL knowledge representation [2], which is a frame-based system designed to store general domain knowledge and cases in a common structure. The strengths of the representation is currently in classification structures, causal and statistical models and default reasoning.

3.1 Target model

The target (or expert) model contains the episodic and general domain knowledge of the system. It can be seen as what we would like the student to approach in knowledge. If he learns all that the system can teach, he should end up with a student model close to the target model. The target model contains both general domain knowledge and a case-base of exercises.

As an example, we can consider the domain of the ELM system – a computer science programming class. In this case, the general domain knowledge might contain grammar rules, templates and general techniques for solving a type of problem, while the exercise-base contains problems and programs solving the particular problems. Another example, from the medical domain, is the training of medical students in interpreting ECG graphs. Here, the domain model would contain procedural information on how to read different variables (e.g. the time between heartbeats), and how these variables alone or together may indicate different types of problems.

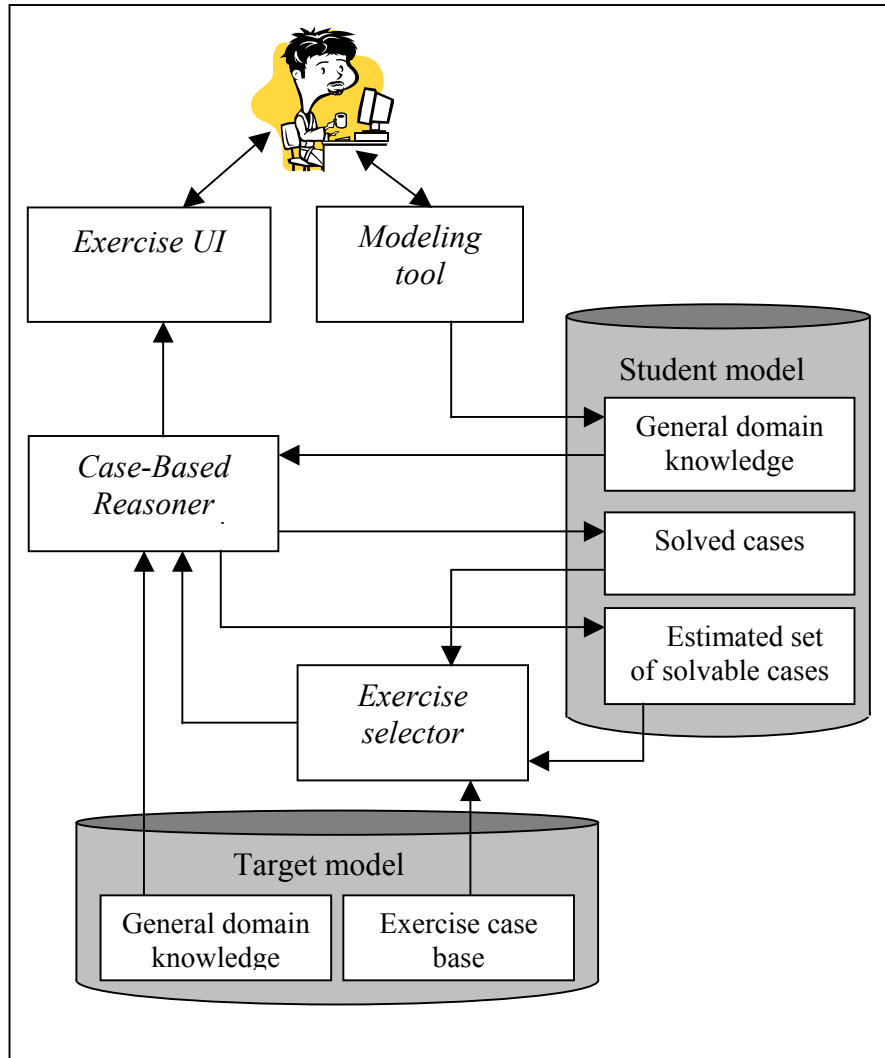


Fig. 1. The system architecture

3.2 Student Model

The student model has the same components as the target model. In this system, the general domain model component of the student model is formed directly and explicitly by the student through a modeling tool where she can describe to the system what she knows. This opens up the possibility that the student will express the same knowledge as the expert but in a different way. At this point we are not sure

how big an impact this will have on the system, as the student model is not directly compared to the expert model. Rather, the result of the reasoning processes for each model is compared. It is an open question how different the expert and student model can be before it becomes a problem for the reasoner, but the modeling tool will guide the student to express his knowledge in a way that is not too far away from the target model.

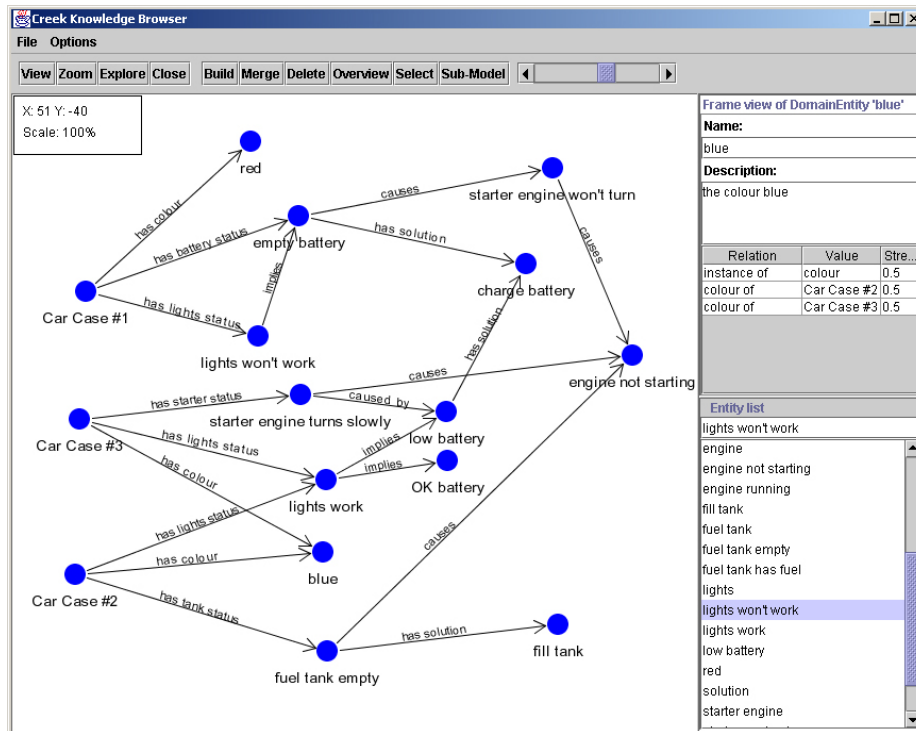


Fig. 2. The modeling tool interface has three parts:

1. The main graphical model editor to the left, where the modeler can create entities in the domain and connect them with relations – e.g. saying that *lights won't work* implies *empty battery*.
2. The list of entities available in the current model (bottom right pane)
3. In the top right pane, the description of the currently chosen entity, *blue*, with direct and inherited (in cyan) relations displayed in a table.

The other component of the student model is the exercises solved by the student. In addition to keeping track of the exercises the student has actually solved (by using an overlay of the cases in the target model), the student model also keeps track of what exercises the student should be able to solve based on analogies from solved exercises and general domain knowledge. This information is provided by the case-based reasoner, which can use the student model to attempt to solve all the exercises in the target model.

3.3. Modeling tool

As mentioned earlier, we have chosen to let the student explicitly model her domain knowledge with the help of a modeling tool in order to be able to focus on how to use this model and compare it to an expert model. We are going to base this tool on the modeling tools already developed for building models in the Creek system (see figure 2).

We will use this modeling tool to create an environment where the student is asked to use entities and relations to describe his understanding of, for instance what may cause a car not to start. In order to limit the orders of freedom, we may limit the student to use only the entities and relation-types used in the expert model – in other words, the student must play an advanced version of “connect the dots”.

3.4 Exercise selector

The task of the exercise selector is to suggest appropriate exercises for the student. There are several possible strategies to follow here. A policy could be to always suggest an exercise that challenge the student but that is possible for her to solve. This could mean suggesting exercises that require some small amount of knowledge in addition to what is present in the student model. Another policy could be to select exercises the student is able to complete without additional knowledge in order to solidify the knowledge, or to sometimes suggest exercises using parts of the knowledge that hasn't been used in a long time. Potentially, this module can use methods like in [4], where custom exercises for the student is designed by adapting cases and combining several sub-problems to form a more complex task. However, this is not considered a research focus at the moment.

3.5 Case-based reasoner

In our design, the task of the Case-based reasoner is to understand the difference in how the student attempts to solve a problem compared to an expert. We hypothesize that it is possible for a knowledge-intensive CBR system like Creek to attempt to solve the problem from the point of view of the student by using the student model, and from the point of view of the expert by using the pre-stored expert model. By comparing the traces of these problems-solving processes, the system will be able to see where the student (eventually) departs from the reasoning of the expert, and may suggest solutions to the student. More details about how we want to do this are found in section 4.

3.6 Exercise user interface

The exercise must be well presented to the student. Ideally, she should be presented with an interface that will let her through the steps of finding a solution inside the

system so that the system can give hints and help if the student is stuck. We consider this mainly a user-interface problem at this point, as the case-based reasoner is able to provide expert and student solution traces, and we will not dwell on it here.

4 The ITS-CREEK problem-solver

To illustrate how we think a knowledge-intensive case-based reasoner like Creek can be used to emulate the problem-solving process in a student and an expert, we have created an example in a “car fault” domain (as shown in the modelling tool figure 2). In this small domain, we are trying to find the reasons why a car will not start, and it basically recognizes two different solutions: The car will not start because it is out of fuel, or because the battery power is too low. If we imagine a situation where *Case #1* (a battery empty case) and *Case #2* (an out of fuel case) is already stored in the case-base and also known to the student, we have a third case (*Case #3*) presented as an exercise to the student. This case will on first glance match better to *Case #2* than *Case #1*, as it has a few direct matches with features in *Case #2* (see figure 2 - both are blue cars, and in both cases the headlights work). Even if we look more carefully, it seems that the fact that the headlights work in *Case #3* indicate that the battery power is OK. However, there is a finding that is unmatched in both *Case #1* and *#2* – the starter engine is turning very slowly. This is usually caused by there being some battery power left, but not enough to start the car. Given the full domain model above, Creek is able to match *Case #3* to *Case #1* because it could match the *empty battery* finding of *Case #1* to the (implied) *low battery* finding in *Case #3* and see that they fill the same role in explaining that the engine did not start. When running this model through Creek, it came up with an explanation supporting the *charge battery* solution that was judged to be two to three times as strong as the *fill fuel tank* explanation. Figure 3 shows the justifications Creek found in the first case.

If we imagine that a student training in this domain have yet to learn that a slow-turning starter engine may not be able to start the engine, and that this is also a symptom that the battery power is low, the outcome would have been quite different. In order to simulate this, we removed the *causes/caused by* relations from the *engine turns slowly* entity in the model in figure 2, and let Creek try to solve the same problem with the reduced domain model. The explanations Creek found from the point of view of this student are shown in figure 4. As we can see, the number of explanations is much smaller, and the total strength of the explanation is judged to be less than half as strong as the expert explanation in figure 4. In fact, if we ask Creek to display explanations in support of the *empty fuel tank* solution, it will produce an explanation of almost the same strength (figure 5), and with the direct matches between *Case #3* and *Case #2*, it chooses *Case #2* as the strongest match, in effect suggesting to refill the fuel tank.

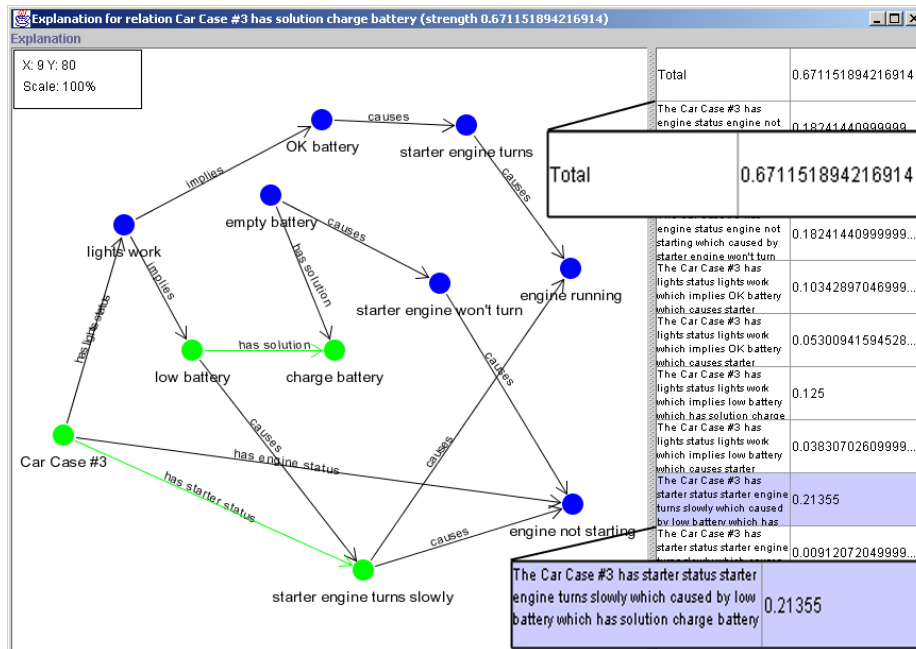


Fig.3 Explanations in support of *charge battery* being the solution to *Case Case #3*. On the left hand side of the screen, Creek has formed a submodel of the expert model containing all entities and relations it deems relevant, and on the right hand side a list of justifications for inferring that the *has solution charge battery* relation applies to *Car Case #3*. The strongest of these are highlighted, and represents the link from *starter engine turns slowly* implying *low battery*, which has the solution *charge battery*. Other explanations strengthen the *low battery* implication by showing that *lights work* is consistent with *low battery*, and that the *charge battery* is a possible solution to the *engine not starting* problem. While none of the individual justifications are particularly strong (the strongest is 0.21 on a scale from 0 to 1), the total sum of the justifications is judged to be 0.67 .

In the above example, we assume the only differences between the expert and student model are in parts of the general domain model, but it may also be in the number of cases known. It may be that the expert system knows of cases that match the exercise *Case #3* much better, and it can then use this case as an example to explain the situation to the student.

The model-based reasoning engine used by Creek is based on a form of default reasoning by rules, which we call plausible inheritance [9, 10]. This method allows the knowledge modeler to specify rules like “The *causes* relations are allowed to be inherited over other *causes* relations,” effectively defining *causes* as transitive. Another example from the above domain is that the *has solution* relation may be inherited over *implies*, *causes* and *has ... status* relations. These rules are used to propagate the *has solution charge battery* relationship from the *low battery* entity via the *starter engine turns slowly* entity to the *Case #3* case. By storing the propagation

path, it can be used as an explanation for why *Case #3* have an implied *has solution charge battery*. Currently, this explanation engine is able to strengthen an explanation for one such inference by considering multiple propagation paths, but it is not able to weaken it by considering negative evidence or checking coherence. An example of this last weakness is found in figure 4, where one of the explanations assume an implied *empty battery* in order to justify the *charge battery* solution, while the battery level in *Case #3* is actually found to be *low battery*. Obviously, the battery can't be both low on power and completely dead. In an ITS context we suspect that the capability to produce explanations that weakens evidence will also be necessary, as well as explanations on a higher level, for instance why the solution suggested by a student is not as good as the one suggested by the expert.

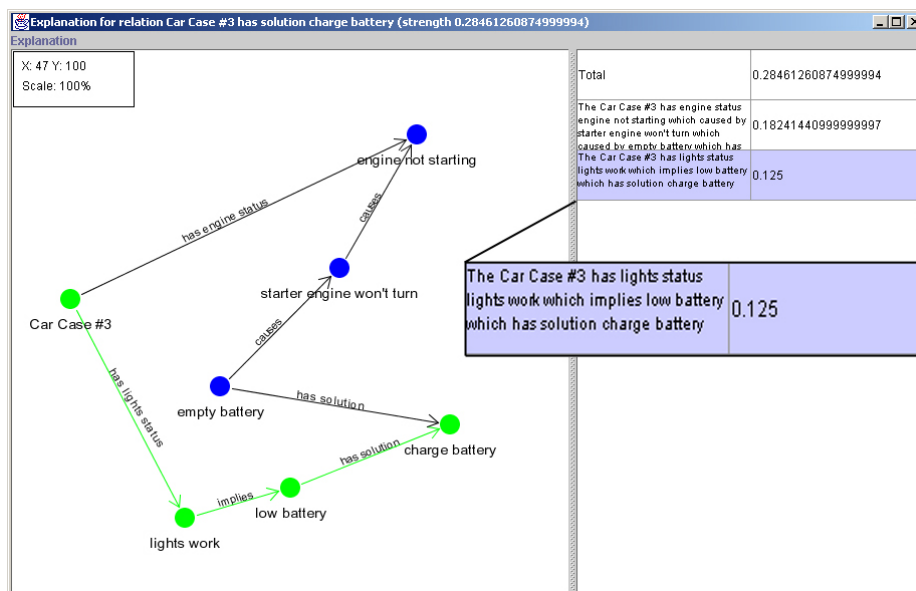


Fig.4 Explanations from running the reduced general knowledge model through the Creek explanation engine. All the justifications involving the *starter engine turns slowly* is gone, and the explanations focus on saying that the *charge battery* solution is consistent with both the *lights work* finding and the *engine not starting* problem of the case. The total strength of this justification is a low 0.28 on a scale from 0 to 1.

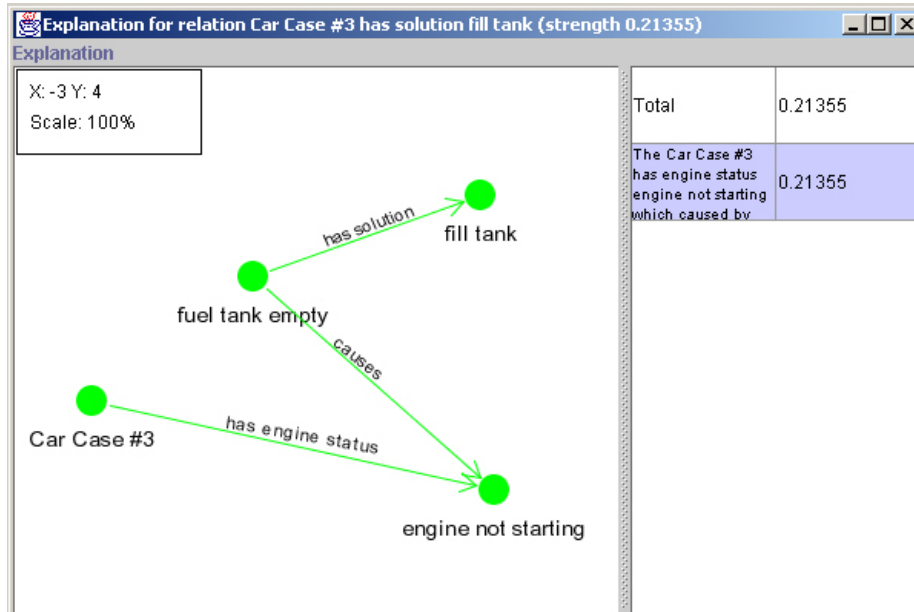


Fig. 5. The only justification supporting the *fuel tank empty* solution is one saying that the *engine not starting* problem may be caused by a *fuel tank empty* situation, although no other finding support this conclusion, so the strength of the explanation is judged to be low (0.21 on a scale from 0 to 1).

CREEK is best at producing explanations in domains with causal or statistical models. Other areas that are often important in ITS systems, such as representing different problem-solving strategies and procedural knowledge in general is currently not supported and may be something we will look into in this work. Currently, the CREEK problem-solving strategies are created for diagnostic domains (such as identifying a disease from symptoms), and we expect this system to be most applicable in these domains as well. Another limitation of the system described, is the reliance of students to be able to model their own knowledge. This requires that the student have some level of sophistication. The ITS-CREEK approach can potentially be used in combination with techniques that is able to build a student model by analyzing student behavior, thus avoiding this limitation.

5 Conclusion

The main goal of the work presented in this paper is to study how case-based reasoning can be extended to provide the type of explanation and justification required by a problem-solving emulation module in an ITS system. We are excited over this possibility not only because ITS is an exciting field to apply techniques

from case-based reasoning, but also because the techniques developed in this field could very well prove to be quite useful in case-based reasoning in general. The ability for a case-based reasoner to communicate justifications and reasoning steps should be appreciated in many domains. In some situations, such a justification can help the user evaluate the solution suggested by the system, and in others it might even serve to teach the user about the domain even when used primarily as a decision support tool. It is our hope and belief that working on improving the ability for CBR methods to communicate explanations, justifications and information about the reasoning process will improve CBR as a method both in ITS and in general.

References

1. Aamodt A.: A Knowledge-Intensive Integrated Approach to Problem Solving and Sustained Learning. PhD. Dissertation. University of Trondheim, Department of Electrical Engineering and Computer Science, Trondheim (1991)
2. Aamodt A.: A Knowledge Representation System for Integration of General and Case-Specific Knowledge. Proceedings from IEEE TAI-94, International Conference on Tools with Artificial Intelligence (1994). New Orleans, November 5-12.
3. Aleven V., Ashley K.D.: How different is different? Arguing about the significance of similarities and differences. Advances in Case Based Reasoning Third European Workshop, EWCBR 96 Proceedings. Springer Verlag, Berlin, Germany (1996) 1-15
4. Ashley K.D., Aleven V.: Reasoning symbolically about partially matched cases. IJCAI 97 Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers, San Francisco, CA, USA (1997) vol 1 335-41
5. Dong Mei Zhang, Alem L.: Using case-based reasoning for exercise design in simulation-based training. Intelligent Tutoring Systems Third International Conference, ITS '96 Proceedings. Springer Verlag, Berlin, Germany (1996) 560-8
6. Schank R.: Dynamic memory; a theory of reminding and learning in computers and people. Cambridge University Press. (1982)
7. Seitz A.: A case-based methodology for planning individualized case oriented tutoring. Case Based Reasoning Research and Development Third International Conference on Case Based Reasoning, ICCBR 99 Proceedings (Lecture Notes in Artificial Intelligence Vol 1650). Springer Verlag, Berlin, Germany (1999) 318-28
8. Shiri A., Aimeur E., Frasson C.: SARA: a case-based student modeling system. Advances in Case Based Reasoning 4th European Workshop, EWCBR 98 Proceedings. Springer Verlag, Berlin, Germany (1998) 394-403
9. Soeromo F., Aamodt A.: Knowledge Elaboration for Improved CBR. IJCAI 97 Workshop on Automating the Construction of Case Based Reasoners. Morgan Kaufmann, San Francisco, CA (1999) 39-43

10. Soeremo F.: Plausible Inheritance; Semantic Network Inference for Case-Based Reasoning. MSc thesis, NTNU, Department of Computer and Information Science (2000)
11. Wenger, E.: Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. Morgan Kaufmann, Los Altos, CA (1987)
12. Weber G.: Episodic learner modeling. Cognitive Science. Ablex Publ. Corp., Norwood, N.J. (1996) vol 20, no 2 195-236