

Generating Tailored Examples to Support Learning via Self-explanation

Cristina Conati and Giuseppe Carenini

Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
{conati, carenini}@cs.ubc.ca

Abstract

We describe a framework that helps students learn from examples by generating example problem solutions whose level of detail is tailored to the students' domain knowledge. The framework uses natural language generation techniques and a probabilistic student model to selectively introduce gaps in the example solution, so that the student can practice applying rules learned from previous examples in problem solving episodes of difficulty adequate to her knowledge. Filling in solution gaps is part of the meta-cognitive skill known as self-explanation (generate explanations to oneself to clarify an example solution), which is crucial to effectively learn from examples. In this paper, we describe how examples with tailored solution gaps are generated and how they are used to support students in learning through gap-filling self-explanation.

1 Introduction

Studying examples is one of the most natural ways of learning a new skill. Thus, substantial research in the field of Intelligent Tutoring Systems (ITS) has been devoted to understand how to use examples to enhance learning. Most of this research has focused on how to select examples that can help a student during problem solving [e.g., Burrow and Weber 1996; Alevan and Ashley 1997]. In this paper, we focus on how to describe an example solution so that a student can learn the most by studying it previous to problem solving. In particular, we address the issue of how to vary the level of detail of the presented example solution, so that the same example can be equally stimulating for learners with different degrees of domain knowledge.

This problem is novel in ITS, as it requires sophisticated natural language generation (NLG) techniques. While the NLG field has extensively studied the process of producing text tailored to a model of the user's inferential capabilities [e.g., Horacek 1997; Korb, McConachy et al. 1997; Young 1999], the application of NLG techniques in ITS are few and mainly focused on managing and structuring the tutorial dialogue [e.g., Moore 1996; Freedman 2000], rather than on tailoring the presentation of instructional material to a detailed student model.

The rationale behind varying the level of detail of an example solution lies on cognitive science studies showing that those students who self-explain examples (i.e., generate explanations to themselves to clarify an example solution) learn better than those students who read the examples without elaborating them [Chi 2000]. One kind of self-explanation that these studies showed to be correlated with learning involves filling in the gaps commonly found in textbook example solutions (*gap filling* self-explanation). However, the same studies also showed that most students tend not to self-explain spontaneously. In the case of gap filling, this phenomenon could be due to the fact that gap filling virtually requires performing problem solving steps while studying an example. And, because problem solving can be highly cognitively and motivationally demanding [Sweller 1988], if the gaps in an example solution are too many or too difficult for a given student, they may hinder self-explanations aimed at filling them.

We argue that, by monitoring how a student's knowledge changes when studying a sequence of examples, it is possible to introduce in the examples solution gaps that are not too cognitively demanding, thus facilitating gap filling self-explanation and providing a smooth transition from example study to problem solving. We are testing our hypothesis by extending the SE-Coach, a framework to support self-explanation of physics examples [Conati and VanLehn 2000].

The SE-Coach already effectively guides two other kinds of self-explanations that have been shown to trigger learning [Chi 2000]: (i) justify a solution step in terms of the domain theory (*step correctness*); (ii) map a solution step into the high-level plan underlying the example solution (*step utility*). The internal representation of an example solution used by the SE-Coach to monitor students' self-explanation is generated automatically. However, because the SE-Coach does not include any NLG capability, the example description presented to the student and the mapping between this description and the internal representation is done by hand. Thus, each example has a fixed description, containing virtually no solution gaps.

In this paper, we describe how we extended the SE-Coach with NLG techniques to (i) automatically generate the

example presentation from the example internal representation (ii) selectively insert gaps in the example presentation, tailored to a student’s domain knowledge.

Several NLG computational models proposed in the literature generate concise text by taking into account the inferential capabilities of the user. [Young 1999] generates effective plan descriptions tailored to the hearer’s *plan reasoning* capabilities. [Horacek 1997] is an example of models that take into account the hearer’s *logical inference* capabilities. And [Korb, McConachy et al. 1997] proposes a system that relies on a model of user’s *probabilistic inferences* to generate sufficiently persuasive arguments.

In contrast, our generation system tailors the content and organisation of an example to a *probabilistic model of the user logical inferences*, which allows us to explicitly represent the inherent uncertainty involved in assessing a learner’s knowledge and reasoning processes. Furthermore, our system maintains information on what example parts are not initially presented (i.e., solution gaps), which is critical to support gap-filling self-explanations for those students who tend not to self-explain autonomously.

In the following sections, we first illustrate our general framework for example generation. We then describe in detail the NLG techniques used and an example of the tailored presentations they generate. Finally, we show how the output of the NLG process supports an interface to guide gap filling self-explanation.

2 The Framework for Example Generation

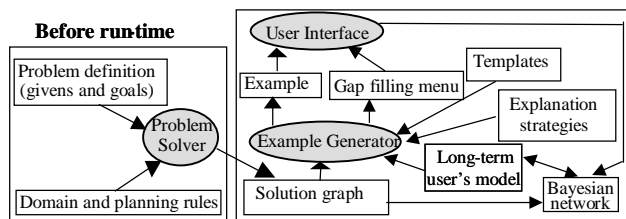


Figure 1: Framework for example generation

Figure 1 shows the architecture of our framework for generating tailored example presentations. The part of the framework labelled “before run-time” is responsible for generating the internal representation of an example solution from (i) a knowledge base (KB) of domain and planning rules (for physics in this particular application); (ii) a formal description of the example initial situation, given quantities and sought quantities [Conati and VanLehn 2000]. A problem solver uses these two knowledge sources to generate the example solution represented as a dependency network, known as the *solution graph*. The solution graph encodes how each intermediate result in the example solution is derived from a domain or planning rule and from previous results matching that rule’s preconditions. Consider, for instance, the physics example in Figure 2 (Example1). Figure 3 shows the part of solution graph that derives the first three steps mentioned in Example1 solution:

establish the goal to apply Newton’s 2nd Law; select the body to which to apply the law; identify the existence of a tension force on the body.

In the solution graph, intermediate solution facts and goals (F- and G- nodes in Figure 3) are connected to the rules (R-nodes) used to derive them and to previous facts and goals matching these rules’ enabling conditions. The connection goes through rule-application nodes (RA- nodes in Figure 3), explicitly representing the application of each rule in the context of a specific example. Thus, the segment of network in Figure 3 encodes that the rule *R-try-Newton-2law* establishes the goal to apply Newton’s 2nd Law (node *G-try-Newton-2law*) to solve the goal to find the force on Jake (node *G-force-on Jake*).

<p>EXAMPLE 1: Boy rescued by a helicopter Jake, an 80Kg undergrad, is rescued from a burning building by a helicopter. He hangs at the end of a rope dangling beneath the helicopter. If the helicopter accelerates, straight downward with respect to the ground, with an acceleration $a = 2\text{m/s}^2$, FIND: The force exerted by the rope on Jake.</p> <p>FREE BODY DIAGRAM: Diagram</p>	<p>SOLUTION</p> <p>Because we want to find a force, we apply Newton’s 2nd law to solve this problem. We choose Jake as the body. The helicopter’s rope exerts a tension force T on Jake. The tension force T is directed upwards. The other force acting on Jake is his weight W. The weight W is directed downwards. To apply Newton’s 2nd law to Jake, we choose a coordinate system with the Y axis directed downward. The Y component of Jake’s weight W is $W_y = W$. The Y component of the tension T on Jake is $T_y = -T$. The net force acting on Jake along the Y axis is $\text{Net-force}_y = W_y + T_y$. Therefore, substituting $W_y = W$, and $T_y = -T$ into the net force equation, we obtain $\text{Net-force}_y = W - T$. If we apply Newton’s 2nd Law to Jake, along the Y axis, we obtain: $\text{Net-force}_y = m \cdot a_y$ The Y component of Jake’s acceleration a is $a_y = a$. Therefore, if we substitute a_y and $\text{Net-force}_y = W - T$ into</p>
--	---

Figure 2: Sample Newtonian physics example

The rule *R-goal-choose-body* sets the subgoal to find a body to apply the Newton’s 2nd Law (node *G-goal-choose-body*), while the rule *R-find-forces* sets the subgoal to find all the forces on the body (node *G-find-forces*). The rule *R-body-by-force* dictates that, if one has the goals to find the force on an object and to select a body to apply Newton’s 2nd Law, that object should be selected as the body. Thus, in Figure 3 this rule selects Jake as the body for Example1 (node *F-Jake-is the body*). The rule *R-tension-exists* says that if an object is tied to a taut string, then there is a tension force exerted by the string on the object. When applied to Example1, this rule generates the fact that there is a tension force on Jake (node *F-tension-on-Jake* in Figure 3).

The solution graph can be seen as a model of correct self-explanation for the example solution, because for each solution fact it encodes the various types of self-explanations relevant to understand it: *step correctness* (what domain rule generated that fact), *step utility* (what

goal that fact fulfils) and *gap filling* (how the fact derives from previous solution steps).

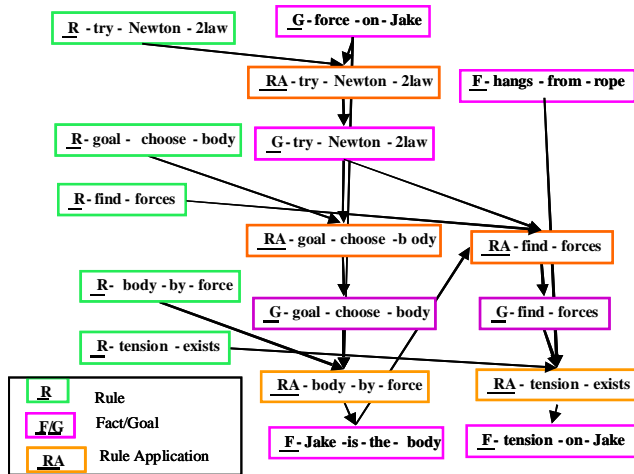


Figure 3: Segment of solution graph for Example 1

In the SE-Coach, every time a student is shown an example, the corresponding solution graph provides the structure for a Bayesian network (see right bottom side of Figure 1) that uses information on how the student reads and self-explains that example to generate a probabilistic assessment of how well the student understands the example and the related rules [Conati and VanLehn 2001]. The prior probabilities to initialise the rule nodes in the Bayesian network come from the long-term student model (see Figure 1), which contains a probabilistic assessment of a student’s current knowledge of each rule in the KB. This assessment is updated every time the student finishes studying an example, with the new rule probabilities computed by the corresponding Bayesian network.

In the SE-Coach, the solution graph and Bayesian network described above are used to support students in generating self-explanations for correctness and utility only. No explicit monitoring and support for gap filling self-explanation is provided. This is because in the SE-Coach, the description of the example solutions presented to the student and the mapping between these descriptions and the corresponding solution graphs are done by hand. This makes it impossible to tailor an example description to the dynamically changing student model by inserting gaps at the appropriate difficulty level for a given student. We have overcome this limitation by adding to the SE-Coach the example generator (see right part of Figure 1), a NLG system that can automatically tailor the detail level of an example description to the student’s knowledge, in order to stimulate and support gap-filling self-explanation.

3 The Example Generator (EG)

EG is designed as a standard pipelined NLG system [Reiter and Dale 2000]. A text planner [Young and Moore 1994] selects and organizes the example content, then a

microplanner and a sentence generator realize this content into language. In generating an example, EG relies on two key communicative knowledge sources (right part of Figure 1): (i) a set of explanation strategies that allow the text planner to determine the example’s content, organization and rhetorical structure; (ii) a set of templates that specifies how the selected content can be phrased in English.

The design of these sources involved a complex acquisition process. We obtained an abstract model of an example’s content and organisation from a detailed analysis of the rules used to generate the solution graph. This was combined with an extensive examination of several physics textbook examples, which also allowed us to model the examples’ rhetorical structure and the syntactic and semantic structure of their clauses. To analyse the rhetorical structure of the examples, we followed Relational Discourse Analysis (RDA) [Moser, Moore et al. 1996], a coding scheme devised to analyse tutorial explanations. The semantic and syntactic structure of the examples’ clauses was used to design the set of templates that map content into English.

We now provide the details of the selection and organisation of the example content. In EG, this process relies on the solution graph and on the probabilistic long term student model. It consists of two phases, text planning and revision, to reduce the complexity of the plan operators and increase the efficiency of the planning process. Text planning selects from the solution graph a knowledge pool of all the propositions (i.e., goals and facts) necessary to solve a given example, and it organizes them according to ordering constraints also extracted from the solution graph. The output of this phase, if realized, would generate a fully detailed example solution. After text planning, a revision process uses the assessment in the student’s long-term model to decide whether further content selection can be performed to insert appropriate solution gaps. Text planning and revision are described in the following sub-sections.

3.1 Text Planning Process

The input to the text planner consists of (i) the abstract communicative action of describing an example solution; (ii) the example solution graph; (iii) the explanation strategies. The planning process selects and organizes the content of the example solution by iterating through a loop of communicative action decomposition¹. Abstract actions are decomposed until primitive communicative actions (executable as speech acts) are reached. In performing this task, the text planner relies on the set of explanation strategies that specify possible decompositions for each communicative action and the constraints dictating when they may be applied. These constraints are checked against

¹ Communicative actions satisfy communicative goals. So, text planning actually involves two intertwined processes of goal and action decomposition. To simplify our presentation, we only refer to communicative actions and their decomposition.

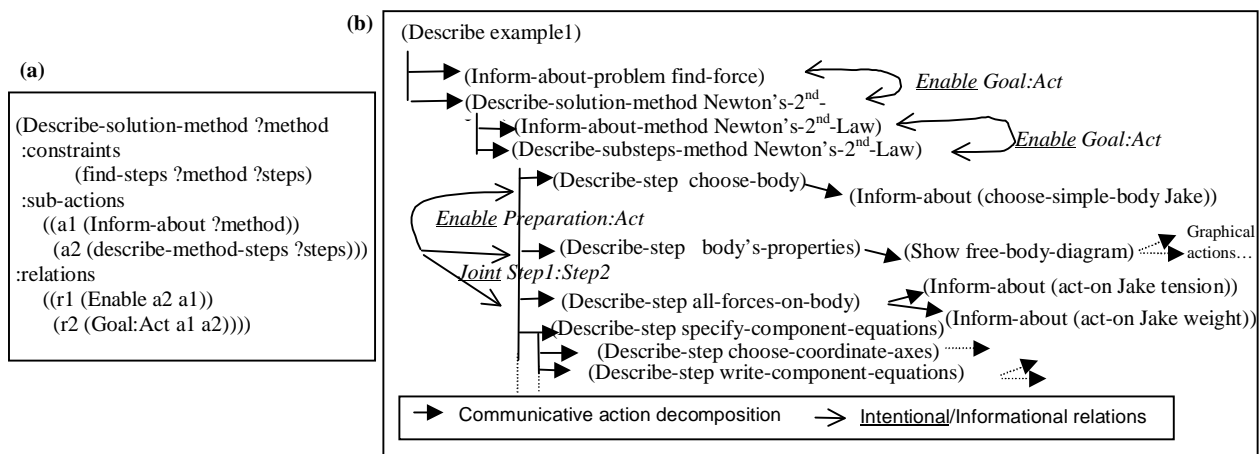


Figure 4: (a) Sample explanation strategy. (b) Portion of the text plan

the solution graph and when they are satisfied the decomposition is selected and appropriate content is also extracted from the solution graph. For illustration, Figure 4(a) shows a simplified explanation strategy that decomposes the communicative action *describe-solution-method*. Possible arguments for this action are, for instance, the Newton's-2nd-Law and the Conservation-of-Energy methods. Looking at the details of the strategy, the function *find-steps* (:constraints field) checks in the solution graph whether the method has any steps. If this is the case, the steps are retrieved from the solution graph and the *describe-solution-method* action is decomposed in an *inform-about* primitive action and in a *describe-method-steps* abstract action. The output of the planning process is a text plan, a data structure that specifies what propositions the example should convey, a partial order over those propositions and the example rhetorical structure. A portion of the text plan generated by EG for Example1 is shown in Figure 4(b).

The propositions that the example should convey are specified as arguments of the primitive actions in the text plan. In Figure 4(b) all primitive actions are of type *inform*. For instance, the primitive action (*Inform-about (act-on Jake weight)*) specifies the proposition (*act-on Jake weight*), which is realized in the example description as “*the other force acting on Jake is his weight*”. In the text plan, the communicative actions are partially ordered. This ordering is not shown in the figure for clarity's sake; the reader can assume that the actions are ordered starting at the top. The example rhetorical structure consists of the action decomposition tree and the informational/intentional relations among the communicative actions. For instance, in (b), the rhetorical structure associated with the action *describe-solution-method* specifies that, to describe the solution method, the system has to perform two actions: (i) *inform the user about the method adopted*; (ii) *describe all the steps of the method*. Between these two actions the *Enable* intentional relation and the *Goal:Act* informational relation hold. All the informational /intentional relations used in EG are discussed in [Moser, Moore et al. 1996]. We

clarify here only the meaning of the *Enable* relation because this relation is critical in supporting gap-filling self-explanations. An intentional *Enable* relation holds between two communicative actions if one provides information intended to increase either the hearer's understanding of the material presented by the other, or her ability to perform the domain action presented by the other.

3.2 The Revision Process

Once the text planner has generated a text plan for the complete example, the revision process revises the plan to possibly insert solution gaps that can make the example more stimulating for a specific student. The idea is to insert solution gaps of adequate difficulty, so that the student can practice applying newly acquired knowledge without incurring in the excessive cognitive load that too demanding problem solving can generate [Sweller 1988].

The revision process performs further content selection by consulting the probabilistic long-term student model that estimates the current student's domain knowledge. More specifically, the revision process examines each proposition specified by a primitive communicative action in the text plan and, if according to the student model, there is high probability that the student knows the rule necessary to infer that proposition, the action is de-activated. De-activated actions are kept in the text plan but are not realized in the text, thus creating solution gaps. However, as we will see in the next section, de-activated actions may be realized in follow-up interactions.

As an illustration of the effects of the revision process on content selection, compare the example solutions shown in Figure 6 and Figure 7. Figure 6 displays the worked out solution for Example2 which, similarly to Example1, does not contain any solution gaps. In contrast, the same portion of Example2 solution shown in Figure 7 is much shorter, including several solution gaps. As previously described, EG determines what information to leave out by consulting the long-term probabilistic student model. In particular, the concise solution in Figure 7 is generated by EG if the

student had previously studied Example1 with the SE-Coach and generated self-explanations of correctness and utility providing sufficient evidence that she understands the rules used to derive Example1 solution. When selecting the content for Example2, EG leaves out all the propositions derived from the rules that the student has learned from Example1. Notice, for instance, that the concise solution in Figure 7 does not mention the solution method used and the weight force. Also, the choice of the body and of the coordinate system is only conveyed indirectly.

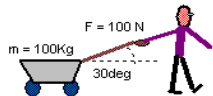
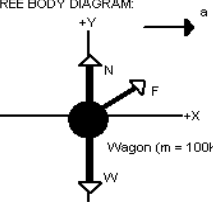
<p>EXAMPLE 2: Person pulling a wagon</p> <p>A person pulls a loaded wagon. The wagon has mass $m = 100\text{Kg}$. The person pulls it with a force F of 100 Newtons, applied at 30 degrees from the horizontal.</p> <p>FIND</p> <ol style="list-style-type: none"> 1) the force N exerted on the wagon by the ground 2) the acceleration a of the wagon.  <p>FREE BODY DIAGRAM:</p> 	<p>SOLUTION</p> <p>We solve this problem by applying Newton's 2nd law. We choose the wagon as the body. One of the forces acting on the wagon is its weight W. The wagon's weight W is directed downward. The force N exerted by the ground on the wagon is a normal force. This normal force N is directed upward. We choose a coordinate system with the X axis directed to the right and the Y axis directed upward. Therefore the weight has components:</p> $w_{y} = -W$ $w_{x} = 0.$ <p>The normal force N has components:</p> $N_{y} = N$ $N_{x} = 0.$ <p>Finally, the pulling force F on the wagon has components:</p> $F_{y} = F \cdot \cos(60)$ $F_{x} = F \cdot \cos(30)$ <p>Because the Y component of the net force on the wagon is:</p> $\text{Net-force}_{y} = N_{y} + w_{y} + F_{y},$ <p>the Y component of the wagon's acceleration is:</p> $a_{y} = 0$ <p>the general equation for Newton's 2nd law applied to the wagon along the Y axis, $\text{Net-Force}_{y} = m \cdot a_{y}$, becomes:</p> $N - W + F \cdot \cos(60) = 0.$ <p>Since the value of the wagon's weight W is:</p> $W = m \cdot g = 980 \text{ Newtons},$
--	---

Figure 6 Portion of Example2 without solution gaps

Even if a student has sufficient knowledge to fill in the solution gaps inserted by the revision process, she may not actually perform the required inferences when studying the example. As a matter of fact, cognitive science studies show that most students tend not to self-explain spontaneously [Chi 2000]. Thus, once the text plan is revised and realized, the system presents the concise example with tools designed to stimulate gap filling self-explanation as we illustrate in the next section.

4 Support for Gap Filling Self-explanation

To support gap-filling self-explanation, we have extended the interface that the SE-Coach uses to support self-explanations for step correctness and utility. In this interface, each example's graphical element and solution step presented to the student is covered with grey boxes. Figure 8(a) shows a segment of the example solution in Figure 7 as presented with the masking interface.

To view an example part, the student must move the mouse over the box that covers it, thus allowing the interface to track what the student is reading. When the

SOLUTION

The force N exerted by the ground on the wagon is a normal force.

This normal force N is directed upward.

The pulling force F on the wagon has components:

$$F_{y} = F \cdot \cos(60)$$

$$F_{x} = F \cdot \cos(30)$$

and the Y component of the wagon's acceleration is:

$$a_{y} = 0$$

The general equation for Newton's 2nd law applied to the wagon along the Y axis, $\text{Net-Force}_{y} = m \cdot a_{y}$, becomes:

$$N - W + F \cdot \cos(60) = 0.$$

Since the value of the wagon's weight W is:

$$W = m \cdot g = 980 \text{ Newtons},$$

Figure 7 Portion of Example2 with solution gap

<p>EXAMPLE 2:</p> <p>[Masked text]</p>	<p>SOLUTION</p> <p>The force N exerted on the wagon by the ground is a normal force</p> <p>[Masked text]</p>	<p>(a)</p> <p>Self-Explain</p> <p>Filling in missing step(s)</p> <p>This fact is true because...</p> <p>The role of this fact in the solution plan is...</p>
<p>SOLUTION</p> <p>Text item for gap</p> <p>The force N exerted on the wagon by the ground is a normal force</p> <p>[Masked text]</p>	<p>FILLING MISSING STEPS</p> <p>Fill in the following missing step(s)</p> <p>Text item for gap</p> <p>[Input field]</p> <p>Submit Done</p>	<p>(b)</p>

Figure 8 Interface tools for gap filling self-explanation

student uncovers an example part, a “self-explain” button appears next to it (see Figure 8(a)). Clicking on this button generates more specific prompts that suggest one or more of the self-explanations for correctness, utility or gap filling, depending upon which of them are needed by the current student to fully understand the uncovered step. In particular, the text plan produced by EG is the key element in determining whether a prompt for gap filling is generated. A prompt for gap filling is generated whenever some of the primitive communicative actions that were de-activated during the revision process are related through an *Enable* intentional relation to the communicative action expressing the uncovered example part. The rationale behind this condition is that a solution gap with respect to an example part comprises all the solution steps that were left out, but whose understanding is a direct precondition to derive that example part. For instance, given the example part uncovered in Figure 8(a), there is only one solution gap preceding it, namely the one corresponding to the communicative action *Inform-about (choose-simple-body-Jake)*². As shown in Figure 8(a), the prompt for gap filling is generated by adding the item “filling in missing steps” to the self-explain menu. If the student clicks on this item, the

² Since the text plans for Example1 and Example2 are structurally the same, this can be verified in Figure 4(b)

interface inserts in the solution text an appropriate number of masking boxes, representing the missing steps (see Figure 8(b), left panel, first box from top). The interface also activates a dialogue box containing a blank for each missing step, that the student can use to fill in the step (see Figure 8(b), right panel). Since the interface currently does not process natural language input, the student fills each blank by selecting an item in the associated pull-down menu. EG generates the entries in this menu by applying the realisation component to unrealised communicative actions in the text plan (see Figure 1).

The student receives immediate feedback on the correctness of his selection, which is also sent to the Bayesian network built for the current example (see Figure 1). The network fact node that corresponds to the missing step is clamped to either true or false, depending on the correctness of the student's selection, and the network updates the probability of the corresponding rule consequently. Thus, if the student's actions show that he is not ready to apply a given rule to fill a solution gap, this rule's probability will decrease in the long-term student model. As a consequence, the next presented example involving this rule will include the solution steps the rule generates, giving the student another opportunity to see how the rule is applied.

5 Conclusions and Future Work

We have presented a tutoring framework that integrates principles and techniques from ITS and NLG to improve the effectiveness of example studying for learning. Our framework uses an NLG module and a probabilistic student model to introduce solution gaps in the example solutions presented to a student. Gaps are introduced when the student model assesses that the student has gained from previous examples sufficient knowledge of the rules necessary to derive the eliminated steps. The goal is to allow the student to practice applying these rules in problem solving episodes of difficulty adequate for his knowledge.

Our framework is innovative in two ways. First, it extends ITS research on supporting the acquisition of the learning skill known as self-explanation, by providing tailored guidance for gap filling self-explanation. Second, it extends NLG techniques on producing user-tailored text by relying on a dynamically updated probabilistic model of the user logical inferences.

The next step in our research will be to test the effectiveness of our framework through empirical studies. These studies are crucial to refine the probability threshold currently used to decide when to leave out a solution step, and possibly to identify additional principles to inform the text plan revision. Additional future work involves NLG research on how the example text plan can be used to maintain the coherence of the other example portions, when the student fills a solution gap.

References

- [Aleven and Ashley 1997] Aleven, V. and K. Ashley. Teaching case-based argumentation through a model and examples: empirical evaluation of an intelligent learning environment. *AIED'99*, Kobe, Japan, August 1997.
- [Burrow and Weber 1996] Burrow, R. and G. Weber. Example explanation in learning environments. *Intelligent Tutoring Systems - Proceedings of the Third International Conference, ITS '96*, Springer, June 1996.
- [Chi 2000] Chi, M. T. H. Self-Explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models. *Advances in Instructional Psychology*. R. Glaser. Mahwah, NJ, Lawrence Erlbaum Associates: 161-238, 2000.
- [Conati and VanLehn 2001] Conati, C. and K. VanLehn. "Providing adaptive support to the understanding of instructional material". *Proc. of IUI 2001, International Conference on Intelligent User Interfaces*, Santa Fe, NM, January 2001.
- [Conati and VanLehn 2000] Conati, C. and K. VanLehn. "Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation." *Int. Journal of AI in Education* **11**, 2000.
- [Freedman 2000] Freedman, R. Plan-based dialogue management in a physics tutor. *Sixth Applied Natural Language Processing Conference*, Seattle, WA, 2000.
- [Horacek 1997] Horacek, H. "A Model for Adapting Explanations to the User's Likely Inferences." *UMUAI* **7**(1): 1-55, 1997.
- [Korb, McConachy et al. 1997] Korb, K. B., R. McConachy, et al. A Cognitive Model of Argumentation. *Proc. 19th Cognitive Science Conf*, Stanford, CA, August 1997.
- [Moore 1996] Moore, J. "Discourse generation for instructional applications: Making computer-based tutors more like humans." *Journal of AI in Education* **7**(2), 1996.
- [Moser, Moore et al. 1996] Moser, M. G., J. D. Moore and E. Glendeling. Instructions for Coding Explanations: Identifying Segments, Relations and Minimal Units, TR 96-17 Univ. of Pittsburgh, Dept. of Computer Science, 1996.
- [Reiter and Dale 2000] Reiter, E. and R. Dale. *Building NLG Systems*, Cambridge University Press, 2000.
- [Sweller 1988] Sweller, J. "Cognitive load during problem solving: effects on learning." *Cognitive Science* **12**: 257-285, 1988.
- [Young 1999] Young, M. "Using Grice's maxim of Quantity to select the content of plan descriptions." *Artificial Intelligence Journal* **115**(2): 215-256, 1999.
- [Young and Moore 1994] Young, M. and J. Moore. DPOCL: A Principled Approach to Discourse Planning. *Proc. 7th Int. Workshop on NLG*, 13-20, 1994.