

G. CARENINI AND C. CONATI

## GENERATING TAILORED WORKED-OUT PROBLEM SOLUTIONS TO HELP STUDENTS LEARN FROM EXAMPLES

### 1. INTRODUCTION

Studying examples is one of the most natural ways of learning a new skill. When studying a worked out solution to a problem, students can focus all their cognitive resources on understanding one solution step at a time, without being overwhelmed by the possibly too demanding task of solving the whole problem from scratch. Given the critical role played by worked out examples in learning, it is not surprising that substantial research in the field of Intelligent Tutoring Systems (ITS) has been devoted to understand how to use examples effectively.

Most of this research has focused on how to select examples that can help a student during problem solving. ELM-PE [Burrow and Weber 1996] and ELM-ART [Weber and Specht 1997] are two tutoring systems that allow the student to access relevant examples while solving LISP programming problems and provide explanations on how each example is relevant for the problem solution. SHERLOCK [Gott, Lesgold et al. 1996], provides expert solutions to troubleshooting problems, and helps students compare these solutions with their own solutions at the end of each problem solving task. CATO [Alevan and Ashley 1997] helps students building legal arguments by generating relevant example cases and by reifying the connection between the content of the cases and their use in the arguments.

In contrast with previous work, the research presented in this chapter does not investigate how to select examples that can help a student *during* problem solving. Rather, our focus is on how to describe an example solution so that a student can learn the most by studying it *prior* 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 [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 [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 findings about self-explanation (i.e., generate explanations to themselves to clarify an example solution) and cognitive load (i.e., the load that performing a particular task imposes on the learner's cognitive system). Several studies indicate that those students who self-explain examples 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. Furthermore, if the gaps are too few and too easy the learner's cognitive system can be overloaded by the unnecessary information.

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 appropriate solution gaps, thus facilitating gap filling self-explanation and providing a smooth transition from example study to problem solving. We are exploring this 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 chapter, we describe how we extended the SE-Coach with NLG techniques to (i) automatically generate the example presentation from the example internal representation, and (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] presents a system that generates concise plan descriptions tailored to the hearer's *plan reasoning* capabilities. In instructing the user how to accomplish a certain task, for instance initializing a PDA, the system leaves out details if the user is assumed to be able to fill them in, given a model of the user's planning algorithm and preferences. [Horacek 1997] describes a system that takes into account the hearer's *logical inference* capabilities. The system generates explanations for planning office space. Facts are omitted from an explanation, if the user is expected to be able to infer them from previous information and context. The model of the user's inferential capabilities consists of a set of deterministic logical

rules associated with a stereotypical model of the user's domain expertise. Finally, [Korb, McConachy et al. 1997] proposes a system that relies on a model of user's *probabilistic inferences* to generate sufficiently persuasive arguments. In order to assess whether the user's degree of belief in the argument conclusion will be within a target range once an argument is presented, the system represents the argument as a Bayesian network which encodes a probabilistic model of the user's beliefs. This model can also incorporate common errors in human reasoning under uncertainty.

In contrast to all of these approaches, our generation system tailors the content and organisation of an example to a *probabilistic model of the user's logical inferences*, namely, a model of the system's uncertainty about the user's knowledge of a set of deterministic rules. This 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 briefly present the Cognitive Load Theory (CLT), a theory of instructional design rooted in cognitive science that provides the background and motivation of our work. After that, we illustrate our general framework for example generation and presentation. We start by describing in detail the NLG techniques used and an example of the tailored presentations they generate. Then, we show how the output of the NLG process supports an interface to guide gap filling self-explanation. We conclude with a discussion of future work.

## 2. COGNITIVE LOAD THEORY

Cognitive Load Theory (CLT) is an instructional theory that takes into account the student's cognitive limitations as primary factors in learning processes [Sweller 1994], [Cooper 1998]. At the core of the theory is the recognition that, in order to learn a new skill, relevant information must be first attended to and then processed in working memory, before it is eventually stored in long-term memory. However, working memory is limited in capacity and it can store information only for a limited time. Therefore, according to CLT, effective learning can occur only if the learner's cognitive system (working memory in particular) is not overloaded with information.

On these grounds, CLT strongly criticizes conventional instructional strategies in which problem solving is excessively emphasized. Traditionally, when a new topic is taught, only few worked out examples are shown and learning how to apply newly introduced principles and rules is supposed to occur when students practice them in solving problems. However, novices who did not have enough practice with sufficiently varied worked out examples cannot learn effectively when they perform problems solving. The reason is that, when novices solve a problem, they typically waste a considerable amount of their limited cognitive resources on processes (e.g., means-ends analyses) that are not directly relevant for attending and processing the information intended to be learned (i.e., how to apply and combine relevant rules and principles in the target domain).

CLT proposes several remedies to improve traditional instructional strategies by taking into account the learner's cognitive limitations. These include: goal free problem solving (i.e., instead of asking students to find a specific quantity, students are asked to find what they can), have learners study many worked out examples and then complete partially solved problems, avoid splitting the student's attention across multiple sources of information (e.g., graphic and text), avoid presenting information redundantly (where what is redundant may depend on the learner's level of expertise [Kalyuga, Chandler et al. 1997]). Notice that all these suggestions from CLT have been empirically shown to be beneficial to learners in several domains, ranging from geometry to biology (e.g., [Paas and Merrienboer 1994]).

The key aspect of CLT that inspires our work is the importance placed on example studying in learning. CLT claims that, by studying a worked out example, the learner needs only to attend and process one solution step at a time. As a result, learning how to apply the rule used to derive that step is unlikely to be hampered by cognitive overload. However, it is clear that, as the student gains expertise in the domain, fully specified examples may become less and less effective, because the student already masters how to derive some of the required steps. Ideally, what the student needs are examples that leave out steps she is already familiar with, but still specify the steps she needs to learn how to derive.

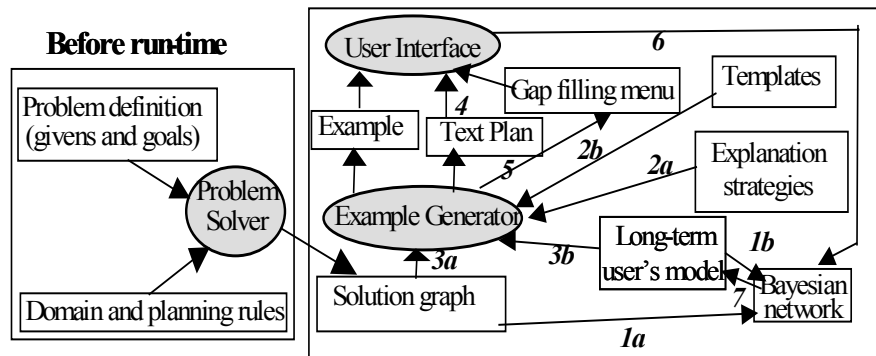


Figure 1. Framework for example generation

In the following sections, we describe a framework that does exactly that. It generates examples whose level of detail is tailored to the learner's degree of domain knowledge. According to CLT, on one hand, if the example says too much (i.e., too few and/or too small gaps in the solution), it will generate unnecessary cognitive load because the learner will have to pay attention to steps that she already knows how to derive. On the other hand, if the example says too little (i.e., too many

and/or too large gaps in the solution), it will also generate unnecessary cognitive load, because the learner will have to fill in the missing steps through problem solving that may be too cognitively demanding. Essentially, the challenge for our system is to strike a difficult balance between saying too much and saying too little, in order to avoid overloading the learner and therefore promote effective learning.

### 3. THE 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 the solution graph that derives the first three steps mentioned in the Example1 solution:

- establish the goal to apply Newton’s 2<sup>nd</sup> Law (the corresponding text in Example1 is: “*Because we want to find a force, we apply Newton’s 2<sup>nd</sup> law to solve this problem*”);
- select the body to which to apply the law (the corresponding text in Example1 is: “*We choose Jake as the body*”);
- identify the existence of a tension force on the body (the corresponding text in Example1 is: “*The helicopter’s rope exerts a tension force T on Jake*”).

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), explicitly representing the application of each rule in the context of a specific example. Thus, the segment of network in Figure 2 encodes that the rule *R-try-Newton-2law* establishes the goal to apply Newton’s 2<sup>nd</sup> Law (node *G-try-Newton-2law*) to solve the goal to find the force on Jake (node *G-force-on Jake*).

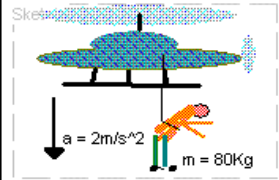
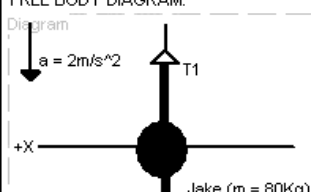
<p><b>EXAMPLE 1: Boy rescued by a helicopter</b></p> <p>Jake, an 80Kg undergrad, is rescued from a burning building by a helicopter.</p> <p>He hangs at the end of a rope dangling beneath the helicopter.</p> <p>If the helicopter accelerates, straight downward with respect to the ground, with an acceleration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

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, link-1a) that uses information about 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 2000]. The prior probabilities to initialise the rule nodes in the Bayesian network come from the long-term student model (see Figure, link-3b), which contains a probabilistic assessment of a student's current knowledge of each rule in the KB.

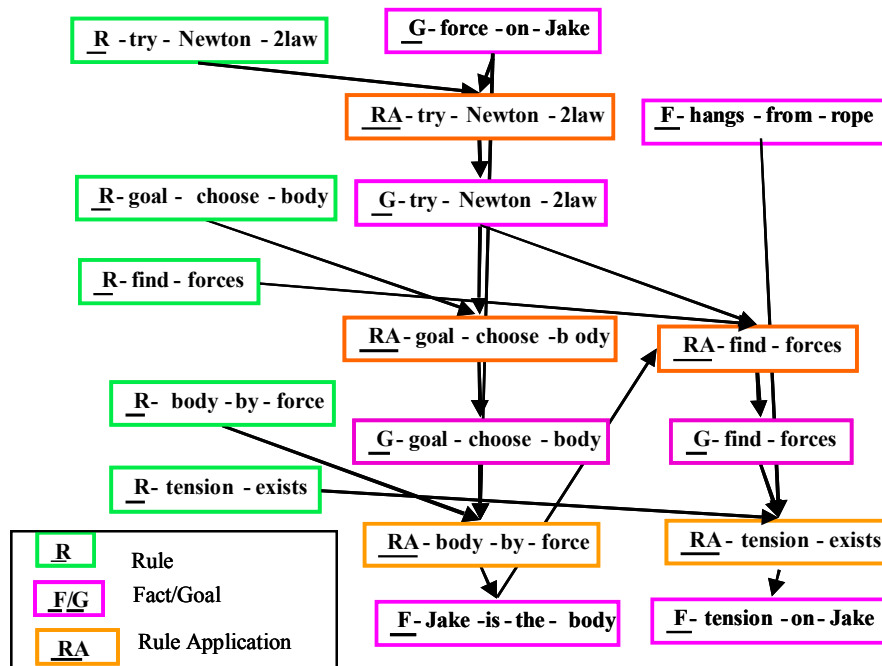


Figure 3: Segment of solution graph for Example 1.

The long-term student model is created when the user starts using the SE-Coach. At first, the model is initialized either with existing data on student relevant Physics knowledge, or with uniform priors, when these data are not available. Then, the model is updated every time the student finishes studying an example, with the new rule probabilities computed by the corresponding Bayesian network (see Figure, link-7).

In the SE-Coach, the solution graph and Bayesian network described above are used to support students in generating self-explanations for step correctness and step 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 Figure), an NLG system that can automatically tailor the detail level of an example description to the student's knowledge, in order to avoid overloading the student's cognitive resources and thus stimulate and support gap-filling self-explanation.

#### 4. 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 (Figure, links 2a and 2b): (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 (Figure 1, links 3a and 3b). 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.

##### *4.1 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 decomposition<sup>1</sup>.



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 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-2<sup>nd</sup>-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 Example 1 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*

Figure(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.

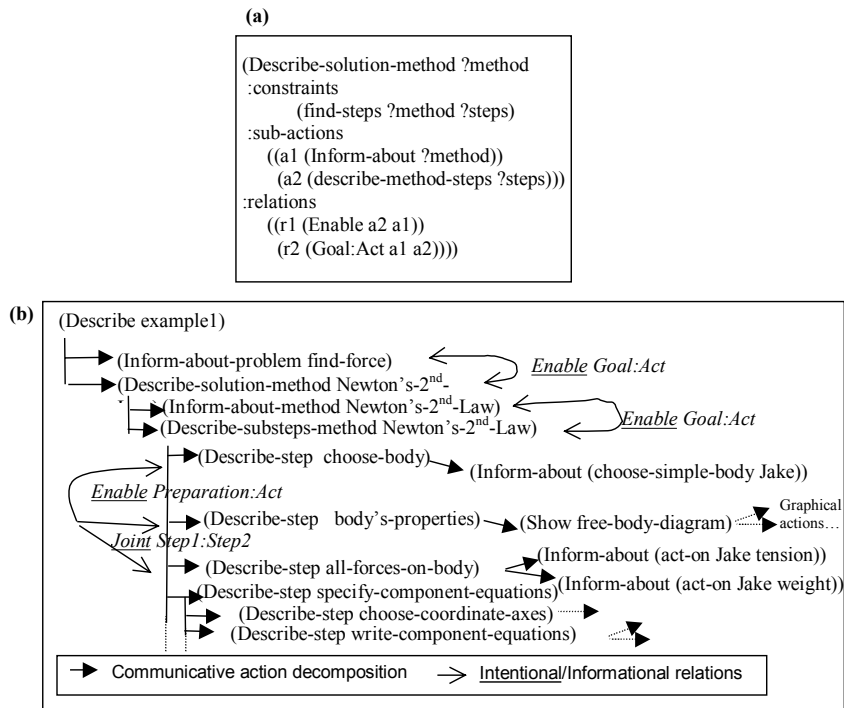


Figure 4: (a) Sample explanation strategy. (b) Portion of the text plan.

#### 4.2 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. As described in the section on CLT, the key idea is to insert solution gaps of adequate difficulty. In this way, the student can practice applying newly acquired knowledge without incurring in the excessive cognitive load due either to too demanding problem solving (i.e., when gaps are too many/large), or to unnecessary information (i.e., when gaps are too few/small).

In most NLG systems, the revision process typically involves reorganizing the text plan. In contrast, revision in EG does not change the structure of the text plan. Rather, the EG revision process only performs further content selection by consulting the probabilistic long-term student model that estimates the current student's domain knowledge (see of Figure, link-3b). 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 a high probability (above a given threshold  $\theta$ ) 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 5 and Figure 6. Figure 5 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 6 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 6 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 6 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.

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. These tools help a student to detect gaps in an example solution and to fill the gaps.

## 5. 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, described in [Conati and Vanlehn 2000] each example's graphical element and solution step presented to the student is covered with gray boxes.

Figure 7 shows a segment of the example solution in Figure 6 as presented with the masking interface.

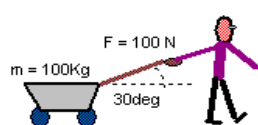
To view an example part, the student must move the mouse over the box that covers it, thus allowing the interface to track what the student is reading. When the student uncovers an example part, if the SE-Coach determines that the student needs to self-explain that step at that time, a "self-explain" button appears next to it (see Figure 8 (a1)). 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.

## EXAMPLE 2: Person pulling a wagon

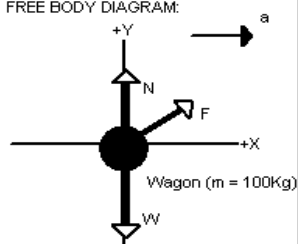
A person pulls a loaded wagon.  
The wagon has mass  $m = 100\text{Kg}$ .  
The person pulls it with a force  $F$  of 100 Newtons,  
applied at 30 degrees from  
the horizontal.

## FIND

- 1) the force  $N$  exerted on the wagon by the ground
- 2) The acceleration  $a$  of the wagon.



## FREE BODY DIAGRAM:



## SOLUTION

We solve this problem by applying Newtons' 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:

$$W_y = -W$$

$$W_x = 0.$$

The normal force  $N$  has components:

$$N_y = N$$

$$N_x = 0.$$

Finally, the pulling force  $F$  on the wagon has components:

$$F_y = F \cdot \cos(60)$$

$$F_x = F \cdot \cos(30)$$

Because the  $Y$  component of the net force on the wagon is:

$$\text{Net-force}_y = N_y + W_y + F_y,$$

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 5. Portion of Example2 without solution gaps.

In particular, the text plan produced by EG is the key element in determining whether a prompt for gap filling is generated (Figure 1, link 4). 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(a1), there is only one solution gap preceding it, namely the one corresponding to the communicative action *Inform-about (choose-simple-body wagon)*<sup>2</sup>.

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,  $Net-Force_y = m \cdot a_y$ , 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 6. Portion of Example2 with solution gap.

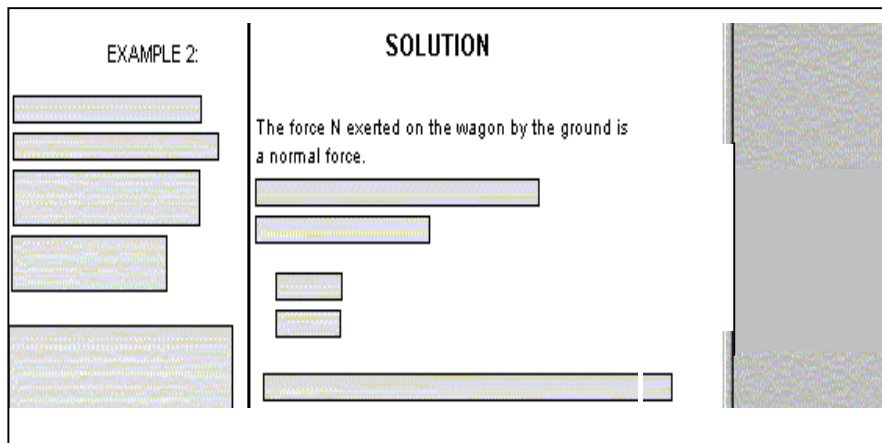
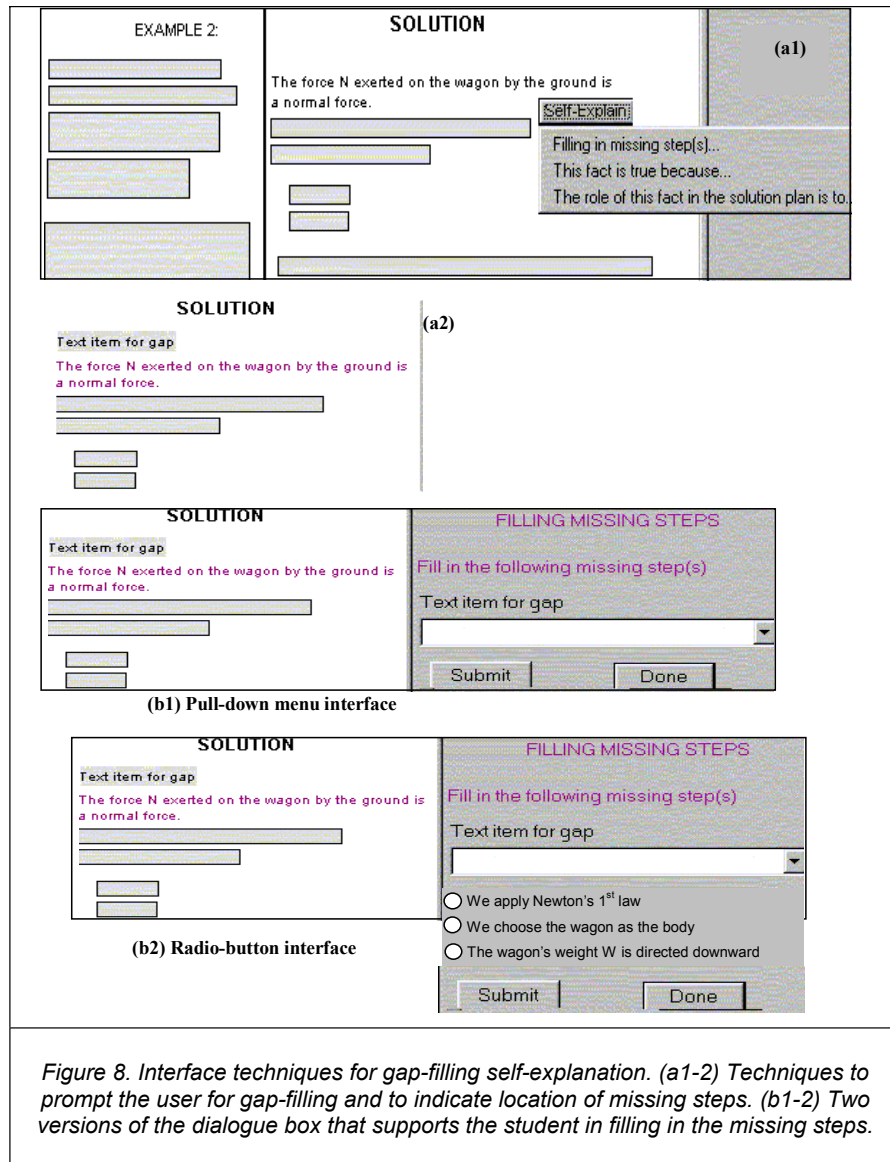


Figure 7. Masking interface to track what the student is attending to.

As shown in Figure 8 (a1), 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 interface inserts in the solution text an appropriate number of masking boxes, representing the missing steps (see Figure 8 (a2), first box from top). The interface also activates a dialogue box that allows the student to fill in the missing steps. Since the interface currently does not process natural language input, the student needs to fill each missing step by selecting a sentence describing the step from a set of possible alternatives. Each alternative is either generated by EG by applying the realisation component to unrealised communicative actions in the text plan (see Figure 1, link 5), or it is randomly chosen from a pool of steps unrelated to the current problem.



We have implemented two versions of the dialogue box that support the student in filling in the missing steps. As shown in Figure 8 (b1-2), one version is based on pull-down menus, while the other on radio-buttons. A key difference between the two is that, when more than one step is missing, radio buttons allow the student to

inspect all the alternatives for all the missing steps at the same time. In contrast, with pull-down menus the student has to focus on one missing step at a time.

We have performed a preliminary user study involving 26 students to determine which of the two versions students prefer. Unfortunately, the result of the study was inconclusive with roughly half of the students preferring one version and the other half preferring the other. Several students mentioned that radio-buttons were confusing when more than one missing step was present, because too much text was displayed. As for pull-down menus, one complaint was the need of a mouse click to see what options are available.

Additional studies are clearly needed, especially to identify user differences that can explain the differences in their preferences; and also to explore whether hybrid combinations of pull-down menus and radio-buttons might be more effective and more acceptable to users.

Going back to the student interaction with the gap-filling interface, let's assume now that the student, regardless of which of the two dialogue boxes had been provided, makes her selection for filling in the missing step. When this happens, 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, link 6). 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.

As we mentioned in Section 3, at the end of the interaction with the current example, the long-term student model is updated with the new rule probabilities from the example Bayesian network. In particular, if the student's actions have shown 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 (see Figure 1, link-7). When this happens, if the resulting rule probability is too low (i.e., below the threshold  $\theta$ ), 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.

## 6. 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.

As future work, in the short term, we plan to investigate how the *prediction capabilities* of the probabilistic student model can be used to refine the strategies that EG uses to decide when to leave out a solution step. Currently the model is used to *diagnose* the student's understanding of the relevant physics rules given the student's interaction with the system. The results of this diagnosis, i.e., the updated rule probabilities, are then used directly to decide if the student can apply each rule to derive a corresponding solution step, and if so to deactivate that step in the presented solution. This approach is reasonable, if we assume that we always insert only gaps that include only one missing solution step. However, if we want to insert gaps that consist of chains of two or more steps, the fact that the student has enough knowledge to infer each of these steps in isolation does not imply that the user can actually make the chain of inferences, because of the increased cognitive load that this process entails. Thus, we are working on using the student model to *predict*, given one or more chains of deactivation, if the student can actually derive them given the current rule probabilities in the model.

A more long-term 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  $\theta$  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. First, we plan to investigate how the example text plan can be used to maintain the coherence of the other example portions, when the student fills a solution gap. Second, we will start working on how to integrate the generation of the textual example solution and of the graphical elements of the example (e.g., the free body diagram). Third, we aim to verify whether a template-based approach to realization is sufficiently powerful to cover our target domain of physics examples.

## 7. ACKNOWLEDGEMENTS

We thank Andréa Burnt for help in reviewing literature on Cognitive Load Theory and its applications in Intelligent Tutoring Systems.

## 8. NOTES

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.
2. Since the text plans for Example1 and Example2 are structurally the same, this can be verified in Figure 4(b), in which the missing step would have been Inform about (choose-simple-body Jake).

## 9. REFERENCES

- [Aleven and Ashley 1997] V. Aleven and K. D. Ashley. Teaching case-based argumentation through a model and examples: Empirical evaluation of an intelligent learning environment. Proc. of the *Artificial Intelligence in Education*, Kobe, Japan, 1997.



- [Burrow and Weber 1996] R. Burrow and G. Weber. Example explanation in learning environments. Proc. of the *Intelligent Tutoring Systems - Proceedings of the Third International Conference, ITS '96*, Springer: 457-465, 1996.
- [Chi 2000] M. T. H. Chi. *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 Carenini 2001] C. Conati and G. Carenini. Generating Tailored Examples to Support Learning via Self-Explanation. Proc. of the *Proceedings of the 17th Joint International Conference on Artificial Intelligence (IJCAI 2001)*, Seattle, USA, 2001.
- [Conati and Vanlehn 2000] C. Conati and K. Vanlehn. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education* 11: 398-415, 2000.
- [Cooper 1998] G. Cooper. Research into Cognitive Load Theory and Instructional Design at UNSW, University of New South Wales, Australia, 1998.
- [Freedman 2000] R. Freedman. Plan-based dialogue management in a physics tutor. Proc. of the *Sixth Applied Natural Language processing Conference (ANLP '00)*, Seattle, 2000.
- [Gott, Lesgold et al. 1996] S. P. Gott, A. Lesgold and R. S. Kane. *Tutoring for transfer of technical competence*. Constructivist Learning Environments. B. G. Wilson. Englewood Cliffs, NJ, Educational Technology Publications: 33-48, 1996.
- [Horacek 1997] H. Horacek. A Model for Adapting Explanations to the User's Likely Inferences. *UMUAI* 7(1): 1-55, 1997.
- [Kalyuga, Chandler et al. 1997] S. Kalyuga, P. Chandler and J. Sweller. Levels of Expertise and User-Adapted Formats of Instructional Presentations: A Cognitive Load Approach. Proc. of the *User Modeling: Proceedings of the Sixth International Conference, UM97*, 1997.
- [Korb, McConachy et al. 1997] K. B. Korb, R. McConachy and I. Zukerman. A Cognitive Model of Argumentation. Proc. of the *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*: 400-405, 1997.
- [Moore 1996] J. D. Moore. Discourse generation for instructional applications: Making computer-based tutors more like humans. *Journal of Artificial Intelligence in Education* 7(2): 181-124, 1996.
- [Moser, Moore et al. 1996] M. G. Moser, J. D. Moore and E. Glendening. Instructions for Coding Explanations: Identifying Segments, Relations and Minimal Units, University of Pittsburgh, Department of Computer Science, 1996.
- [Paas and Merriënboer 1994] F. Paas and J. Merriënboer. Variability of Worked Examples and Transfer of Geometrical problem-Solving Skills: A Cognitive-Load Approach. *Journal of Educational Psychology* 86(1): 122-133, 1994.
- [Reiter and Dale 2000] E. Reiter and R. Dale. *Building Natural Language Generation Systems*, Cambridge University Press, 2000.
- [Sweller 1988] J. Sweller. Cognitive load during problem solving: effects on learning. *Cognitive Science* 12: 257-285, 1988.
- [Sweller 1994] J. Sweller. Cognitive Load Theory, Learning Difficulty, and Instructional Design. *Learning and Instruction* 4: 295-312, 1994.
- [Weber and Specht 1997] G. Weber and M. Specht. User modeling and adaptive navigation support in WWW-based tutoring systems. Proc. of the *Proceedings of User Modeling '97*, 1997.
- [Young 1999] M. R. Young. Using Grice's maxim of Quantity to select the content of plan descriptions. *Artificial Intelligence* 115(2): 215-256, 1999.
- [Young and Moore 1994] R. M. Young and J. D. Moore. DPOCL: A Principled Approach to Discourse Planning. Proc. of the *Proceedings of the 7th International Workshop on Text Generation*, Montreal, Canada, 1994.

## 10. AFFILIATIONS

Giuseppe Carenini and Cristina Conati  
 University of British Columbia  
 Vancouver, Canada