

Tailored Scaffolding for Meta-Cognitive Skills during Analogical Problem Solving

by

Katarzyna Muldner

B.Sc., Acadia University, 1995

M.Sc., University of Victoria, 1997

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

The Faculty of Graduate Studies

(Computer Science)

The University of British Columbia

June, 2007

© Katarzyna Muldner, 2007

Abstract

Although examples play a key role in cognitive skill acquisition, research demonstrates that learning outcomes are heavily influenced by the meta-cognitive skills students bring to bear while using examples. This dissertation involves the design, implementation and evaluation of the Example Analogy (EA)-Coach, an Intelligent Tutoring System that provides adaptive support for meta-cognitive skills during a specific type of example-based learning known as *analogical problem solving* (APS, using examples to aid problem solving). To encourage the targeted meta-cognitive skills, the EA-Coach provides multiple levels of scaffolding, including an innovative example-selection mechanism that aims to choose examples with the best potential to trigger learning and enable problem solving for a given student.

To find such examples, a key factor that needs to be taken into account is problem/example similarity, because it impacts the APS process. However, full understanding has yet to be reached on how various levels of similarity between a problem and an example influence students' APS behaviours and subsequent outcomes. Here, we provide a novel classification of problem/example differences and hypotheses regarding their impact on APS. In particular, we propose that certain differences between a problem and an example may actually be beneficial in helping students learn from APS, because they promote the necessary meta-cognitive skills. However, given the great variance in terms of knowledge and meta-cognitive skills that exists between students, a key challenge with our approach is how to select examples that provide enough scaffolding for different learners. Our solution to this challenge involves a two-step decision-theoretic process. First, the EA-Coach student model, which corresponds to a dynamic Bayesian network, is used to predict how a candidate example will help a student solve the problem and learn from doing so. Second, the model's prediction is quantified via a utility function, which assigns an expected utility to the candidate example. This process allows the framework to present to the student the example with the highest expected utility for enabling learning and problem solving. We evaluated this approach via a controlled laboratory study, which demonstrated the EA-Coach's pedagogical effectiveness for supporting problem solving *and* triggering meta-cognitive skills needed for learning during APS.

Table of Contents

1 Introduction	1
1.1 Background and Motivation	1
1.2 An Intelligent Tutoring System for APS: the Example Analogy (EA)- Coach	3
1.3 Thesis Contributions	5
1.4 Thesis Outline	6
2 Cognitive Science Foundations	7
2.1 Cognitive Science Background	8
2.1.1 Example-Selection Phase	8
2.1.2 Example-Application Phase	9
2.1.2.1 Application Phase: ACT-R	9
2.1.2.2 Application Phase: Impact of Meta-Cognitive Skills	11
2.1.2.3 Application Phase: Impact of Problem/Example Differences	16
2.2 Leveraging Cognitive Science Findings: Impact of Problem/Example Differences on APS	18
2.2.1 Terminology	18
2.2.1.1 Formal Definitions	21
2.2.2 Impact of Differences: Hypotheses	24
2.3 Summary	25
3 Pilot Evaluations	27
3.1 Primary Pilot Evaluation	27
3.1.1 Preliminary EA-Coach Interface	28
3.1.2 Primary Pilot Evaluation Particulars: Participants, Methodology and Materials	30
3.1.2.1 Participants	30
3.1.2.2 Methodology	31

3.1.2.3 Materials.....	32
3.1.3 Primary Pilot Evaluation: Results	35
3.1.4 Primary Pilot: Discussion	40
3.2 Two Follow-up Pilot Evaluations	43
3.3 Summary.....	45
4 Introduction to the EA-Coach.....	47
4.1 The EA-Coach Architecture	48
4.1.1 Before Run-time	49
4.1.2 At Run-time.....	52
4.2 Interacting with the EA-Coach	53
4.3 Summary.....	57
5 Tailored Support for APS through Example Selection	58
5.1 EA-Coach's Example-Selection Objectives	59
5.2 Introduction to the EA-Coach Student Model	59
5.2.1 Handling Uncertainty through Dynamic Bayesian Networks	61
5.2.2 Student Model Construction through the Andes Approach.....	64
5.3 A Decision-theoretic Approach for Example Selection.....	67
5.4 Simulation Phase	68
5.4.1 Simulation Slice	69
5.4.2 Nodes to Model APS Behaviors.....	70
5.4.2.1 Similarity Nodes.....	72
5.4.2.2 Copy Nodes	73
5.4.2.3 EBLC Nodes.....	76
5.4.3 Prediction of Learning and Problem-Solving Success	79
5.4.4 Accounting for Goals in the Simulation.....	82
5.5 Expected Utility Calculation Phase	84
5.5.1 Using a Linearly Additive Multi-Attribute Utility Function: Implications.....	87
5.6 Example-Selection Process: Summary	88
5.7 Role of the Example in the Student Model's Assessment	89
5.8 Summary.....	93

6 Evaluation of the EA-Coach	95
6.1 Objectives & Approach	95
6.2 Study Participants	97
6.3 Study Methodology	98
6.3.1 Training Phase	99
6.3.2 Experimental Phase	99
6.3.3 Initialization Phase	102
6.4 Study Materials	105
6.4.1 Pre and Post Tests	105
6.4.2 The EA-Coach Problems and Examples	105
6.5 Dependent Measures	113
6.6 Results	113
6.6.1 Results: Learning	115
6.6.1.1 APS Behaviors: Min-Analogy	115
6.6.1.2 APS Behaviors: Self-explanation/EBLC Events	116
6.6.1.3 Pre/Post Test Gains	120
6.6.1.4 Learning: Summary of Key Results and Discussion	122
6.6.2 Results: Problem-Solving Performance	127
6.6.2.1 Problem-Solving Success	128
6.6.2.2 Task Time & Errors	128
6.6.2.3 Problem-Solving Performance: Summary of Key Results and Discussion	132
6.7 Summary	136
7 Related Work	138
7.1 ITS Supporting APS	138
7.2 ITS Supporting Pure Problem Solving	142
7.3 ITS Supporting Meta-Cognitive Skills	143
7.4 ITS Relying on a Decision-Theoretic Approach for Action Selection	146
8 Conclusions and Future Work	149
8.1 Future Work	152
8.1.1 Additional Scaffolding	152
8.1.2 Scaffolding the Progression of Skill Acquisition	154

8.1.3 Further Analysis	154
8.1.4 Novel Technologies.....	155
8.1.5 Refinements to the Student Model	156
8.1.6 Subsequent Evaluations.....	156
References	158
Appendix 1	171
Appendix 2	175
Appendix 3	184
Appendix 4	195

List of Figures

Figure 2-1:	(1) <i>Mapping</i> process for a LISP problem & example (see dotted lines) and (2) <i>Application of mapping</i> to generate the '*' in the problem solution (see solid lines/corresponding dotted line, bottom).....	10
Figure 2-2:	Rule inferred from LISP problem/example in figure 2-1	10
Figure 2-3:	Physics problem & example.....	12
Figure 2-4:	Illustration of transformational analogy applied to the problem/example in figure 2-3	12
Figure 2-5:	Rule inferred via EBLC from physics problem/example in figure 2-3	14
Figure 2-6:	Rules formulated via the ACT-R analogy mechanism & EBLC.....	15
Figure 2-7:	Illustration of superficial & structural relations	19
Figure 2-8:	Classification of superficial & structural relations from figure 2-7 supplemented with formal representation	23
Figure 3-1:	Preliminary EA-Coach interface (shown without the masking interface, which is displayed in Figure 3-2)	29
Figure 3-2:	Masking interface	30
Figure 3-3:	Sample viewing sequence in masking interface; two example regions are uncovered: (1) example solution step and (2) example specification.....	38

Figure 3-4:	Sample self-explanations	39
Figure 3-5:	Variable definition pane	44
Figure 4-1:	The EA-Coach architecture	48
Figure 4-2:	Sample rules in the knowledge base.....	49
Figure 4-3:	(a) Fragment of formal specification for problem in figure 2-3 (shown here, top); (b) corresponding solution graph fragment for the problem.....	51
Figure 4-4:	The EA-Coach interface (shown without the masking interface, which is displayed in figure 4-5)	54
Figure 4-5:	Masking interface	56
Figure 5-1:	Example of a Bayesian network	63
Figure 5-2:	Simple dynamic Bayesian network	64
Figure 5-3:	Andes Bayesian network student model.....	65
Figure 5-4:	Fragment of the EA-Coach dynamic Bayesian network	70
Figure 5-5:	Additional nodes in simulation slice to predict APS behaviors	71
Figure 5-6:	Copy nodes in simulation slice.....	74
Figure 5-7:	EBLC nodes in simulation slice	77
Figure 5-8:	Special case for EBLC node.....	79
Figure 5-9:	Prediction of learning & problem-solving success.....	80
Figure 5-10:	Special case in simulation: goal nodes	83

Figure 5-11: The EA-Coach utility model	86
Figure 5-12: Fragment of the EA-Coach dynamic Bayesian network used during assessment mode	90
Figure 6-1: The EA-Coach interface	100
Figure 6-2: Pre-test questions/corresponding marking scheme used to initialize the <i>normal-exists/normal-dir</i> rules	103
Figure 6-3: Sample self-explanations	117
Figure 6-4: Interaction between selection and selection order for task time	129
Figure 6-5: Interaction between selection and selection order for error	131
Figure 8-1: The SE-Coach tool used to generate a self-explanation on the existence of a normal force	153
Figure A-1: Representation of vector-component equations in the solution graph	173

List of Tables

Table 3-1:	Problems & examples used in primary pilot	32
Table 3-2:	Structural differences between problem _{trivial} and problem _{non-trivial}	33
Table 3-3:	Superficial differences between problem _{trivial} & corresponding example steps	33
Table 3-4:	Superficial differences between problem _{non-trivial} & corresponding example steps	34
Table 3-5:	Summary of data analysis for primary pilot	36
Table 3-6:	Summary of analysis for the four subjects who copied/self-explained from examples	37
Table 3-7:	Source of errors in primary pilot evaluation: copying vs. problem- solving	41
Table 5-1:	Copy node CPT	74
Table 5-2:	EBLC node CPT	77
Table 5-3:	Rule node CPT	80
Table 5-4:	Step (fact) node CPT	82
Table 5-5:	Goal node CPT	84
Table 5-6:	Copy node CPT used during assessment	92

Table 6-1:	Problems used during the experimental phase	106
Table 6-2:	Examples included the EA-Coach example pool during the experimental phase	106
Table 6-3:	Structural Differences between problems p_1 and p_2	108
Table 6-4:	Problem/example pairs used in the static condition	109
Table 6-5:	Superficial differences between problem p_1 & corresponding example e_1 in the static condition	110
Table 6-6:	Superficial differences between problem p_2 & corresponding example e_4 in the static condition	111
Table 6-7:	Superficial differences: mean number of trivially and non-trivially different problem/example steps in the two conditions.....	112
Table 6-8:	Summary of ANOVA analysis for copy events ($N=14$).....	116
Table 6-9:	Summary of ANOVA analysis for self-explanation events ($N=14$).....	118
Table 6-10:	Summary of ANOVA analysis for EBLC events ($N=14$)	120
Table 6-11:	Subjects' pre and post test performance ($N=14$)	121
Table 6-12:	Pre/Post test gains.....	122
Table 6-13:	Results of correlation analysis (learning; $N=14$).....	125
Table 6-14:	Summary of ANOVA analysis for task time ($N=14$).....	129
Table 6-15:	Summary of the ANOVA analysis for error ($N=14$).....	131
Table 6-16:	Results of correlation analysis (problem-solving performance; $N=14$) ...	134
Table 6-17:	Source of errors ($N=14$)	135

Acknowledgements

I would like to begin by thanking the members of my supervisory committee: my supervisor, Cristina Conati, as well as James Carolan, Alan Mackworth, and Joanna McGrenere. Thank-you to Cristina for helping me find a project that combines my interests in computer science and education, for sharing her knowledge and time with me throughout my degree, and for her generosity in sending me to conferences that allowed me to establish important ties in the research community. Thank-you to my other committee members, Jim, Joanna, and Alan, for their invaluable advice, support and guidance. I would also like to thank my examining committee: my university examiners, Lee Iverson and David Poole, as well as my external examiner, Jim Greer, for their helpful comments and feedback.

Many thanks go to Maria Klawe, who was my research supervisor while she was at UBC, and who after leaving continued to give me some of her very scarce time to offer guidance and encouragement that it would all work out. Thank-you also to Valerie McRae and Lara Hall, the LCI group assistants and surrogate moms, who looked out for me from behind the scenes.

While at UBC I was fortunate to meet and get to know some amazing fellow grad students. LCI is a great laboratory, and I was lucky enough to work there – so thank-you all for making it memorable. A big thank-you goes to Andrea Bunt, my co-conspirator during square dancing events at conference banquets, reading groups, and our final graduation push. Andrea's encouragement and advice throughout the course of my degree was incredibly helpful. Thank-you also to the other member of 107, Robert, as well as to Mike H., Karyn M. and Leah. Also, to Brian F., both for his support and for introducing me to climbing, which proved to be an invaluable stress reliever, Young-Oon and my friend Elaine. Thank-you to Lea, Trish and Magda for the long distance phone conversations, and for never (almost never) asking me if I was done yet.

This thesis would not have been possible without my family. My parents, who are my friends and greatest mentors, believed in me throughout the highs and lows of this degree and provided endless support, both emotional and financial. Thank-you for coming to Vancouver for my defence - this also applies to my brother “Mincio” and Liz - the day would not have been complete without all of you here. I would also like to thank my grandparents – *Babciu i Dziaku, tak Wam dziękuje za wsparcie, pamięć i modlitwy.*

Finally, I would like to thank my partner and best friend Benjamin for never losing faith in me. Thank you for your love and support during this degree: for all the times you listened patiently to incoherent thesis ramblings, for never complaining about the many physical and mental absences, for knowing when I most needed a hug, and for your pride and support on the big defence day. You offered all this without ever judging or criticising me, for which I am forever grateful – I could not have done this without you.

Chapter 1

Introduction

This dissertation research involves the design, implementation and evaluation of an Intelligent Tutoring System that fosters meta-cognitive skills needed for *analogical problem solving* (APS), that is, using examples to aid problem solving.

1.1 Background and Motivation

An example is a problem whose solution is given to the student, along with the solution's derivation. Research shows that students rely heavily on examples during problem solving when learning a new skill [Anderson, Farrell et al., 1984; Pirolli and Anderson, 1985; Reed, Dempster et al., 1985; LeFevre and Dixon, 1986; Novick, 1995; VanLehn, 1996; VanLehn, 1998], and that examples are more effective aids to problem solving than general procedures alone [Reed and Bolstad, 1991] or hints on the instructional material [Ringenberg and VanLehn, 2006]. However, research in cognitive science also indicates that how beneficial examples are for supporting learning strongly depends on a student's ability to apply the relevant *meta-cognitive* skills [VanLehn, 1998; VanLehn, 1999].

Meta-cognition refers to “*one’s knowledge concerning one’s own cognitive processes and products or anything related to them*” [Flavell, 1976]; more informally, meta-cognition has been referred to as “*thinking about thinking*”. Meta-cognitive skills are therefore domain-independent abilities that are an important aspect of knowing how to learn in general. Unfortunately, research shows that not all students can apply meta-cognitive skills effectively, during either APS or other instructional activities such as studying examples, which hinders learning outcomes [Chi, Bassok et al., 1989; Renkl, 1997; VanLehn, 1998; VanLehn, 1999]. Two meta-cognitive skills that are relevant to APS include:

Min-analogy: solving the problem on one’s own as much as possible instead of by copying from examples, and referring to examples only to resolve problem-solving impasses [VanLehn and Jones, 1993a; VanLehn, 1998].

Explanation-based learning of correctness (EBLC): learning new domain principles via a form of self-explanation (the process of explaining and clarifying instructional material to oneself [Chi, Bassok et al., 1989; Bielaczyc, Pirolli et al., 1995; Renkl, 1999]). EBLC involves relying on one’s existing common sense or overly general knowledge to explain how an example solution step is derived [Chi and VanLehn, 1991; VanLehn, 1992; VanLehn, 1999].

Min-analogy and EBLC are complementary meta-cognitive skills that are beneficial for learning from APS: min-analogy allows students to strengthen their knowledge through practise and discover knowledge gaps, while EBLC can be used to fill these gaps. Unfortunately, some students prefer more shallow processes that hinder learning, such as copying as much as possible from examples without any proactive reasoning on the underlying domain principles (e.g., [VanLehn, 1998; VanLehn, 1999]). For this reason, in this thesis we aim to devise computer-based support to help students learn effectively from APS by fostering min-analogy and EBLC.

1.2 An Intelligent Tutoring System for APS: the Example Analogy (EA)-Coach

Intelligent Tutoring Systems (ITS) are computer applications that employ artificial intelligence techniques to instruct students in an “intelligent way” [VanLehn, 1988]. Although there isn’t an accepted definition of the term “intelligent”, a characteristic shared by many ITS is that they possess knowledge and reasoning capabilities to adapt the instruction to the needs of each individual student. This functionality is motivated by research demonstrating that tailored, one-on-one instruction is more effective in improving learning outcomes than standard classroom instruction [Bloom, 1984]. To provide tailored instruction, a traditional ITS architecture contains the following three components [Shute and Psotka, 1996]:

- The domain model represents/reasons about the domain.
- The tutoring model represents/reasons about pedagogical strategies.
- The user *model* represents/reasons about the user. In the ITS community, a *user model* is also commonly referred to as a *student model*, a term we adopt in this thesis.

The ITS developed for this dissertation work is referred to as the *Example Analogy (EA)-Coach*. The EA-Coach aims to support effective APS by fostering the two meta-cognitive skills we introduced above, min-analogy and EBLC. The EA-Coach’s target instructional domain is introductory Newtonian physics. We chose physics because it is one of the domains for which there is a well-recognized need for innovative educational tools, due to the extreme difficulties that students often encounter in bridging theory and practice. These difficulties arise because students struggle making the leap from theory acquisition to effective problem solving, and/or because they may learn to solve problems effectively without grasping the underlying theory [Halloun and Hestenes, 1985].

The design of the EA-Coach is based on cognitive science research, allowing the tutor to target known student shortcomings during APS. In particular, cognitive science findings show that students have difficulty selecting appropriate examples and applying example

solutions effectively [Reed, 1987; Novick, 1988; VanLehn, 1998]. Thus, the EA-Coach provides students with

- adaptively selected examples that are chosen to encourage min-analogy and EBLC;
- multiple layers of scaffolding embedded in its interface aimed at encouraging students to use the selected example effectively during problem solving.

In order to find appropriate examples, a key factor that needs to be taken into account is the similarity between the problem the student is working on (*target problem* from now on) and the selected example. This factor has substantial impact on the APS process. On the one hand, differences between the target problem and the example may hinder both learning and problem-solving success during APS, if students lack the skills to bridge these differences [Novick, 1995]. On the other hand, while examples that are highly similar to the target problem support problem-solving success, they may interfere with learning, because they allow the solution to be generated by copying from the example [Reed, Dempster et al., 1985]. Unfortunately, there is not much understanding of *how* different levels of similarity influence students' APS behaviors. We argue that certain problem/example differences may actually be beneficial to learning because they promote the necessary APS meta-cognitive skills. This assumption is embedded in the EA-Coach's example-selection process, and one goal of this research is to test it via an empirical evaluation of the EA-Coach.

A key challenge with our approach is how to balance learning and problem-solving success for different students. That is, the goal is to find examples that are different enough from the problem to discourage copying and trigger min-analogy and EBLC, but still provide enough scaffolding to help students achieve problem-solving success. Our solution to this challenge is a decision-theoretic approach that allows the EA-Coach to adaptively select examples tailored to each student's specific needs. The approach entails

- incorporating relevant factors (student knowledge and meta-cognitive skills, as well as problem/example similarity) into a probabilistic student model that corresponds to a dynamic Bayesian network [Dean and Kanazawa, 1989], and using the model

to *predict* how a student will solve the problem and learn from doing so in the presence of a candidate example;

- using a utility function to quantify the model's prediction in terms of the candidate example's expected utility for learning and problem-solving success.

The empirical findings from our evaluation of the EA-Coach validate this approach. In particular, the results demonstrate that the EA-Coach's adaptively selected examples trigger the targeted meta-cognitive skills needed for learning, while supporting successful problem solving.

1.3 Thesis Contributions

The contributions of this dissertation are to the Intelligent Tutoring Systems, User Modeling and Cognitive Science communities [Muldner and Conati, 2005a; Muldner and Conati, 2005b; Conati, Muldner et al., 2006; Muldner and Conati, 2007], as follows:

1. We provide a novel Intelligent Tutoring System that fosters the meta-cognitive skills needed for effective APS by selecting examples with different levels of similarity to the problem, tailored to a student's needs. To date, Intelligent Tutoring Systems that support APS via example selection choose the example with a solution most similar to the target problem's solution, and do not target students' meta-cognitive skills (e.g., [Weber, 1996b; Guin-Duclosson, Jean-Duabias et al., 2002]). Although some Intelligent Tutoring Systems do target meta-cognitive skills related to those supported by the EA-Coach (e.g., [Conati and VanLehn, 2000; Aleven, McLaren et al., 2004]), none do so during APS. Consequently, the design of these tutors does not need to account for an example's impact on problem solving.
2. We provide a probabilistic user model that infers how problem/example similarity and student characteristics impact learning and problem-solving outcomes during APS. Although there has been substantial work on devising probabilistic user models to assess and/or predict learning and/or problem solving outcomes [e.g.,

Pek and Poh, 2000; Mayo and Mitrovic, 2001; Conati, Gertner et al., 2002; Murray, VanLehn et al., 2004; Ting, Zadeh et al., 2006], all these models are embedded into ITS that support instructional activities different from APS.

3. Through the evaluation of the EA-Coach, we provide insight on the impact of problem/example similarity on student APS behaviors and subsequent learning and problem solving outcomes. Work in the cognitive science community has investigated how students solve problems and learn from doing so during APS (e.g., [Reed, Dempster et al., 1985; Novick, 1988; VanLehn, 1998]). However, to date there has been a lack of understanding on how similarity influences APS behaviors and subsequent outcomes, and whether some examples can actually help students to both solve the target problem and learn from doing so.

1.4 Thesis Outline

In chapter 2, we first describe the cognitive science background on APS. We then show how we leveraged this research to generate our own hypotheses on the impact of student characteristics and problem/example similarity on APS outcomes. In chapter 3, we describe a series of pilot evaluations we conducted to gain some preliminary insight into whether our hypotheses were generally appropriate. The pilot evaluations were also used to refine the design of the EA-Coach interface, which along with the system architecture is described in chapter 4. In chapter 5, we present details on the computational mechanisms enabling the EA-Coach to provide students with adaptively selected examples. In chapter 6, we describe the full evaluation that we conducted to evaluate the pedagogical effectiveness of the EA-Coach. We then present the related work from the Intelligent Tutoring Systems community in chapter 7. Finally, we conclude in chapter 8 and offer suggestions for future work.

Chapter 2

Cognitive Science Foundations

In this chapter, we begin by describing the cognitive science background that has shaped the design of the EA-Coach. Although there is considerable cognitive science research on APS, full understanding has yet to be reached on the impact of student characteristics and problem/example similarity on learning and problem solving outcomes. In the latter portion of the chapter, we show how we leveraged cognitive science findings to propose our own hypotheses on the relationship between similarity, student characteristics and APS outcomes, as described in [Muldner and Conati, 2005]. These hypotheses are embedded into the computational mechanisms that deliver the EA-Coach's instructional support, which are described in subsequent chapters.

2.1 Cognitive Science Background

APS can be characterized by two phases: (1) *example selection*, i.e., choosing an example and (2) *example application*, i.e., using the example to solve the target problem [VanLehn, 1998]. If students select appropriate examples and apply example solutions effectively during problem solving, then APS provides opportunities for them to strengthen existing and acquire new domain knowledge (e.g., [Anderson, 1993; VanLehn and Jones, 1993a; VanLehn, 1998; VanLehn, 1999]). Unfortunately, difficulties that students experience in either or both the selection and application phases can hinder learning and problem solving.

2.1.1 Example-Selection Phase

The example-selection phase involves retrieving an example that is similar to the target problem and therefore helpful during problem solving. Problem/example similarity is typically characterized using two dimensions, *superficial* (also referred to as *surface*) and *structural* [Chi, Feltovich et al., 1981; Novick, 1988; Bassok, Wu et al., 1995], defined as follows:

- *Superficial similarity* is determined by comparing features that appear in the problem/example specifications and/or solutions but that are not part of the domain knowledge needed to derive the respective solutions.
- *Structural similarity* is determined by comparing the domain knowledge needed to derive the problem and example solutions.

If an example is not structurally similar to the target problem then it is not appropriate for APS, because its solution cannot be applied to generate the problem solution (as we will discuss in section 2.1.2.3). Unfortunately, students have a tendency to rely heavily on superficial similarity during example selection, without considering structural similarity [Holyoak and Koh, 1987; Novick, 1988; Reed, Willis et al., 1994]. Novick and colleagues [Novick, 1988; Novick and Holyoak, 1991] showed that when ‘distracter examples’ are available, i.e., examples that are *only* superficially similar to the target

problem, students selected them over structurally-appropriate but superficially dissimilar examples. When students only have access to examples that are *both* superficially and structurally similar to the target problem, then reliance on superficial similarity leads them to select maximally similar examples that *are* structurally appropriate [Reed, Willis et al., 1994]. However, students don't learn effectively from using highly similar examples, as we will describe in section 2.1.2.3. Students' reliance on superficial similarity also means that they may fail to notice when a structurally appropriate example is helpful, if the example is not superficially similar to the target problem [Holyoak and Koh, 1987]. Thus, in general, students' reliance on superficial similarity hinders their ability to find appropriate examples during APS.

Reliance on superficial similarity during example selection is particularly common among novice students who have low domain expertise, while students with higher domain expertise are more likely to take into account structural similarity during example selection [Novick, 1988]. The impact of expertise on students' similarity judgments has also been demonstrated in studies involving *problem categorization*, where students classified mathematics problems according to common mathematical structure [Silver, 1979; Chi, Feltovich et al., 1981; Schoenfeld, 1982; Quilici and Mayer, 1996].

2.1.2 Example-Application Phase

The example-application phase involves applying the selected example's solution to generate the target problem's solution. How students do so and what they learn as a result has been the topic of substantial research, which we now describe.

2.1.2.1 Application Phase: ACT-R

John Anderson's *Atomic Components of Thought-Rational (ACT-R)* is a theory of human cognition. ACT-R includes the 'analogy mechanism' as a fundamental mechanism of human learning [Anderson and Thomson, 1989; Anderson, 1993]. The mechanism is derived from studies on APS in the LISP programming domain. It encodes how to (1) use the example solution to generate the problem solution and (2) generalize from doing so to learn a new domain rule.

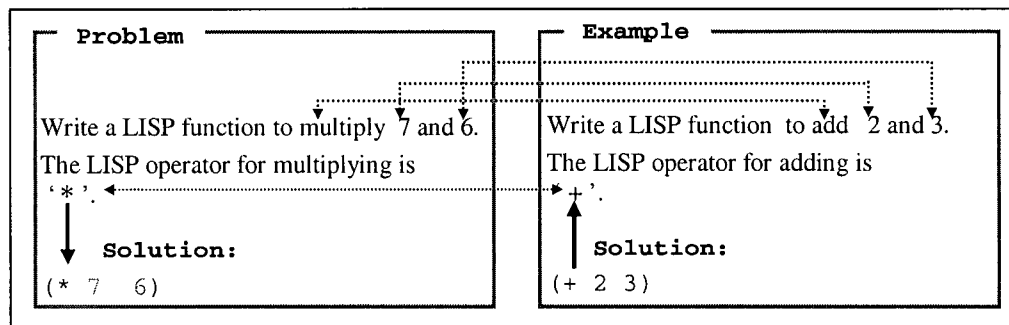


Figure 2-1: (1) *Mapping* process for a LISP problem & example (see dotted lines) and (2) *Application of mapping* to generate the '*' in the problem solution (see solid lines/corresponding dotted line, bottom)

```

if the goal is to write a LISP function
    and the function applies arithmeticOperation to num1 & num2
    and the LISP operator for arithmeticOperation is Op
then
    (Op num1 num2)

```

Figure 2-2: Rule inferred from LISP problem/example in figure 2-1

Using the example to generate the problem solution involves the following two processes:

1. *Mapping*: finding a correspondence between the problem/example constants in the respective specifications. Figure 2-1 shows the mapping between a LISP problem/example specifications via dotted lines.
2. *Application of mapping*: transferring the example solution over to the problem, relying on the mapping to replace example-specific constants by those needed for the problem solution. Figure 2-1 shows how the mapping guides the replacement of the example constant '+' by the constant '*' in the problem solution.

The mapping and application processes are generalized by creating a rule, where

- the rule's antecedent is the conjunction of the elements making up the problem/example specifications (in figure 2-2, see *if* part of the rule inferred from the problem/example in figure 2-1);

- the rule's consequent is the example solution (in figure 2-2, see *then* part of the rule inferred from the problem/example in figure 2-1);
- constants involved in the mapping phase are replaced by variables (e.g., in figure 2-2, *num1* and *num2* replace constants $7/6$ and $2/3$ from the problem/example in figure 2-1, while *Op* replaces constants '*' and '+').

2.1.2.2 Application Phase: Impact of Meta-Cognitive Skills

VanLehn's work on the role of examples in skill acquisition focuses on investigating how learning outcomes are influenced by individual differences between students, and particularly the meta-cognitive skills that students bring to bear [VanLehn, 1992; VanLehn and Jones, 1993a; VanLehn, 1998; VanLehn, 1999]. The research is based on protocol analysis of students studying Newtonian physics examples and subsequently solving problems with access to the examples.

During APS, students have the choice of generating the problem solution on their own or by copying it from the available examples. Copying is the transfer of the example solution over to the problem with no changes or minor changes. An example of a minor change is the substitution of example constants by those needed for the problem solution, a process VanLehn refers to as *transformational analogy*. Figure 2-4 illustrates how, given the physics problem/example in figure 2-3, transformational analogy can be used to transform a solution step in the example to make it suitable for the problem solution. Note that this process is analogous to the mapping/application processes in ACT-R. However, while this process is associated with learning in ACT-R, in VanLehn's work it is associated with a *lack* of learning, as we will describe shortly.

How much a student copies during APS characterizes a meta-cognitive skill that VanLehn identifies as relevant to APS. Specifically, this meta-cognitive skill is *min-analogy*: generating the problem solution through the application of one's own knowledge rather than by copying, relying on examples primarily to overcome impasses that block problem-solving progress. Therefore, students who have a min-analogy tendency prefer to minimize copying from examples. Min-analogy is beneficial to learning for two reasons.

Problem: A workman pushes a 50 kg. block along the floor. He pushes it hard, with a magnitude of 120 N, applied at an angle of 25 degrees as shown. Find the normal force on the block.

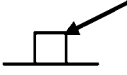
Solution:

[step 1] To solve this problem, we apply Newton's Second Law.

[step 2] We choose the block as the body.

[step 3] There is a normal force N on the block that is due to the floor.

[step 4] It is directed straight-up (90 degrees CCW from the horizontal).



Example: A person pulls a 9 kg. crate up a ramp inclined 30 degrees to the horizontal. The pulling force is applied at an angle of 30 degrees CCW from the horizontal, with a magnitude of 100N. Find the normal force on the crate.

Solution:

[step 1] To solve this problem, we apply Newton's Second Law.

[step 2] We choose the crate as the body.

[step 3] There is a normal force N on the crate that is due to the ramp.

[step 4] It is directed 120 degrees CCW from the horizontal.

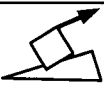


Figure 2-3: Physics problem & example

Problem Specification	Example Specification
<p>A workman pushes a 50 kg. block along the floor. He pushes it hard, with a magnitude of 120 N. applied at an angle of 25 degrees as shown. Find the normal force on the block</p> <p>Solution:</p> <p>We choose the block as the body</p>	<p>A person pulls a 9 kg. crate up a ramp inclined 30 degrees to the horizontal. The pulling force is applied at an angle of 30 degrees CCW from the horizontal, with a magnitude of 100N. Find the normal force on the crate.</p> <p>Solution:</p> <p>[step 1] To solve this problem, we apply Newton's Second Law.</p> <p>[step 2] We choose the crate as the body.</p> <p>[step 3] There is a normal force N on the crate.</p> <p>[step 4] It is directed 120 CCW from the horizontal.</p>

To transform example solution step 2 to make it suitable for the problem:

[1] Isolate the example constant 'crate' in the example specification

[2] Find the corresponding constant 'block' in the problem specification

[3] Generate the step in the problem solution, replacing 'crate' by 'block'

Figure 2-4: Illustration of transformational analogy applied to the problem/example in figure 2-3

First, if students solve problems on their own, they strengthen their existing domain knowledge through practice. Like Anderson, VanLehn proposes that procedural knowledge is represented by *if-then* rules. Each time a student applies a rule on her¹ own without copying, she solidifies that rule in her memory. This increases the likelihood that she will remember the rule the next time it is needed, which increases the chances of problem-solving success (provided that the student has learned a correct rule). Second, min-analogy provides students with opportunities to encounter problem-solving impasses, which expose missing knowledge (referred to as *knowledge gaps* by VanLehn). Gap discovery is an important prerequisite to learning domain principles, since once a student becomes aware of a gap, she may take action to repair it. Unfortunately, although some students prefer min-analogy, others have the opposite tendency, i.e., a *max-analogy* tendency. Students with a max-analogy tendency 'turn control over to the example', maximizing copying without trying to generate the solution on their own, even if they have the knowledge to do so. Thus, these students miss the opportunity to strengthen their knowledge through practice and discover their knowledge gaps. Of course, gap discovery is not sufficient to learn new domain principles. VanLehn postulates that learning is accomplished through a form of self-explanation referred to as *Explanation-based learning of correctness* (EBLC) [Chi and VanLehn, 1991; VanLehn, 1992; VanLehn, 1999]. EBLC is a meta-cognitive skill that involves using one's *existing* common sense and overly general knowledge, in conjunction with domain rules, to infer new rules that explain how a given example solution step is derived. To demonstrate how EBLC works, figure 2-5 shows how it can be used to explain the existence of the normal force mentioned in step 3 of the example in figure 2-3. Specifically, the student relies on her existing

1. *common sense knowledge* to infer that since the ramp supports the crate, there is a force on the ramp applied by the crate (*Common sense* rule in figure 2-5);
2. *overly general knowledge* to infer that this force is an 'official physics force' (*Overly general* rule in figure 2-5);

¹ In this thesis, we will use a mix of feminine and masculine pronouns.

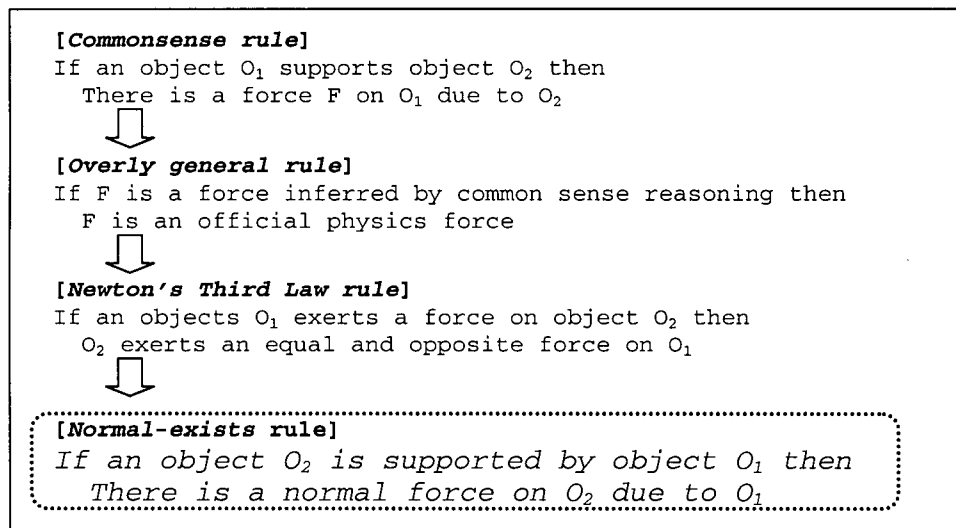


Figure 2-5: Rule inferred via EBLC from physics problem/example in figure 2-3

3. *domain knowledge* to infer that there is a reaction force exerted by the ramp on the crate (*Newton's Third Law* rule in figure 2-5).

As a consequence of this line of reasoning, the student learns a new domain rule (*Normal-exists* rule, figure 2-5). Unfortunately, some students miss learning opportunities because they do not have a tendency for EBLC. These students resolve impasses during APS by resorting to alternative strategies not as conducive to learning to generate the problem solution, such as guessing or copying from the available examples.

To summarize, the meta-cognitive skills of min-analogy and EBLC are complimentary in supporting learning from APS. In particular, min-analogy helps students to strengthen their knowledge through practice and to discover knowledge gaps, while EBLC can be used to fill those gaps. However, having one meta-cognitive skill does not necessarily guarantee the presence of the other. For instance, min-analogy will help a student discover her knowledge gaps, but this does not guarantee that she will rely on EBLC to fill those gaps, since she can resort to alternative means such as guessing to do so. To date, however, VanLehn has not explored how the two meta-cognitive skills interact with one another, and so more research is needed to gain understanding of this issue.

If an agent <i>A</i> applies a force <i>P</i> with a magnitude <i>M</i> and angle θ to an object <i>Obj</i> that is on surface <i>S</i> then there is a normal force on object <i>Obj</i>	} <i>Analogy Mechanism</i>
If an object <i>O</i> ₂ is supported by object <i>O</i> ₁ then There is a normal force on <i>O</i> ₂ due to <i>O</i> ₁	} <i>EBLC</i>

Figure 2-6: Rules formulated via the ACT-R analogy mechanism & EBLC

We should point out that compared to ACT-R's analogy mechanism, EBLC appears to have better potential for inferring appropriate rules in the presence of complex examples, like those in the physics domain targeted by the EA-Coach. ACT-R's analogy mechanism is appropriate for learning from simple examples involving a minimal number of solution steps, such as the LISP examples on which the mechanism is based (e.g., such as the example in figure 2-1). However, with more complex multi-step examples, rules inferred via the analogy mechanism are overly specific and do not reflect a true understanding of the underlying domain principles. To illustrate this limitation, figure 2-6 shows the 'Normal-exists' rule inferred by each mechanism from the problem/example pair in figure 2-3. The analogy-mechanism rule's antecedent is based on the complete problem/example specifications, while its consequent is based on the example solution step. Therefore, the rule reflects at best a shallow understanding of the normal force, because its structure is too closely tied to the problem/example specifications from which it is derived. The rule inferred by EBLC, on the other hand, reflects the kind of knowledge students should possess with respect to the normal force. Consequently, EBLC appears to be the more appropriate learning mechanism for the EA-Coach's domain, and so this is the learning mechanism targeted by the EA-Coach. Similarly, we adopt VanLehn's view that although transformational analogy may enable problem solving during APS, it hinders learning.

2.1.2.3 Application Phase: Impact of Problem/Example Differences

Typically, an example shares some differences with the target problem (otherwise, the example is a direct solution). The differences are commonly classified according to the two similarity dimensions introduced in section 2.1.1, i.e., (1) *structural* and (2) *superficial* (e.g., [Reed, Dempster et al., 1985; Novick, 1995; Quilici and Mayer, 1996]).

[Reed, Ackinclose et al., 1990] compared the impact of two kinds of structural differences on the example-application phase

- differences due to the fact that the problem solution includes steps derived by domain principles (rules) not needed for the example's solution;
- differences due to the fact that the example solution includes steps derived by rules not needed for the problem's solution.

The first type of difference means that the example does not afford students the opportunity to apply its solution to generate all the steps in the problem solution (i.e., either by copying or learning new domain principles needed to generate the problem solution). The second type of difference means that students have to 'filter out' extraneous solution steps in the example. Reed showed that the first type of structural difference hindered students' ability to generate the problem solution more than did the second type of difference.

Several other studies compared the impact of examples, including the first type of structural difference vs. examples that had no structural differences with the target problem [Reed, Dempster et al., 1985; Reed, 1987; Novick, 1995]. All of these studies showed that this type of structural difference hinders the application phase. [Novick, 1995] also found an expertise effect. In particular, students with low domain expertise had more difficulty overcoming structural differences when applying the example solution, compared with students with high expertise. A likely explanation for this finding is that students with high expertise have the knowledge to generate steps corresponding to structural differences on their own, although this was not verified in [Novick, 1995] (i.e., the author does not provide a fine-grained analysis of the

relationship between students' pre-existing domain knowledge and structural differences). In our work, we focus on this type of structural difference, i.e., when the problem solution is derived by rules not needed for the example's solution, because research shows that this difference hinders APS (while the impact of the second type of structural difference is still under investigation).

There is also some research exploring the impact of superficial differences on the example application phase. In [Quilici and Mayer, 1996] students were given two problems to solve, with access to two examples. The superficial similarity between the problems and the examples was assessed using a binary classification, as either *high* or *low*. The classification was performed by the authors based on their subjective judgment on how similar the problem/example specifications were to each other. In this study, the superficial similarity between the problems and the examples was held constant at *low* for all students; the superficial similarity between the two examples was varied, so that some students received two examples that were superficially similar to each other (*high* similarity), while other students received two examples that were not similar to each other (*low* similarity). Compared to *high* superficial similarity, *low* superficial similarity between the examples better triggered learning (measured via subsequent post-test). The authors suggest that an explanation for this finding is that (1) exposure to multiple examples influences the nature of the principles students infer from the examples and (2) low superficial similarity emphasizes structural characteristics, helping students to abstract the irrelevant superficial aspects of the example specifications. However, the paper does not discuss how the abstraction occurs.

Although [Quilici and Mayer, 1996] found that high superficial variability between examples encouraged learning, they did not directly explore the impact of varying the superficial similarity between the problems and the examples. This was investigated by [Ross, 1987], who found that low superficial similarity between a problem/example hindered example application. Ross investigated the impact of reversing the relationships between objects appearing in the problem/example specifications of algebra word problems (e.g., in a problem specification, *computers* are assigned to *offices*, while in an example specification, *offices* are assigned to *computers*). Students were more successful

in applying the example's solution to generate the problem solution when the object correspondences were not reversed than when they were, i.e., when the examples were more superficially similar to the problem. However, neither type of example triggered students to learn the underlying domain principles. This finding that high superficial similarity better helps students generate the problem solution than low superficial similarity, but that neither low nor high superficial similarity stimulates learning was also shown in [Reed, Dempster et al., 1985].

2.2 Leveraging Cognitive Science Findings: Impact of Problem/Example Differences on APS

Although cognitive science research shows that problem/example differences have an impact on APS, there is not yet a complete understanding of how various kinds of differences influence APS behaviors and subsequent learning and problem solving outcomes for learners with various levels of cognitive and meta-cognitive skills. To design the support delivered by the EA-Coach, it was therefore necessary to formulate our own hypotheses. These hypotheses draw directly from the cognitive science background that we presented in the first portion of this chapter on student reasoning during APS. Before we can present our hypotheses, however, we need to introduce some terminology.

2.2.1 Terminology

The terminology we present here is based on the established approach of classifying problem/example relations as *superficial/structural* (e.g., [Reed, Dempster et al., 1985; Novick, 1995; Quilici and Mayer, 1996]). However, we extend this work by proposing a novel classification of superficial differences.

We begin by providing some intuition regarding the structural and superficial relations in our classification. We consider a pair of problem/example steps as *structurally identical* if the steps are generated by the same rule and *structurally different* otherwise. Two

structurally identical steps may include different constants and so be *superficially* different. This scenario is illustrated in figure 2-7, which shows steps 3 and 4 from the problem and example originally shown in figure 2-3. For each solution step, figure 2-7 shows a portion of the step's textual description and the rule used to derive that step. Since *problemStep₃* and *exampleStep₃* are generated by the same rule (*Normal-exists*), they are structurally identical. Likewise, *problemStep₄* and *exampleStep₄* are generated by the same rule (*Normal-dir*) and so are structurally identical. However, both *problemStep₃* and *problemStep₄* are superficially different from their corresponding example steps, in the following ways:

- *problemStep₃* and *exampleStep₃* are superficially different in two respects: (1) the object a normal force exists on, *block* vs. *crate* and (2) the object the force is due to, *floor* vs. *ramp*
- *problemStep₄* and *exampleStep₄* are superficially different with respect to the angle specifying the normal force, 90° vs. 120°

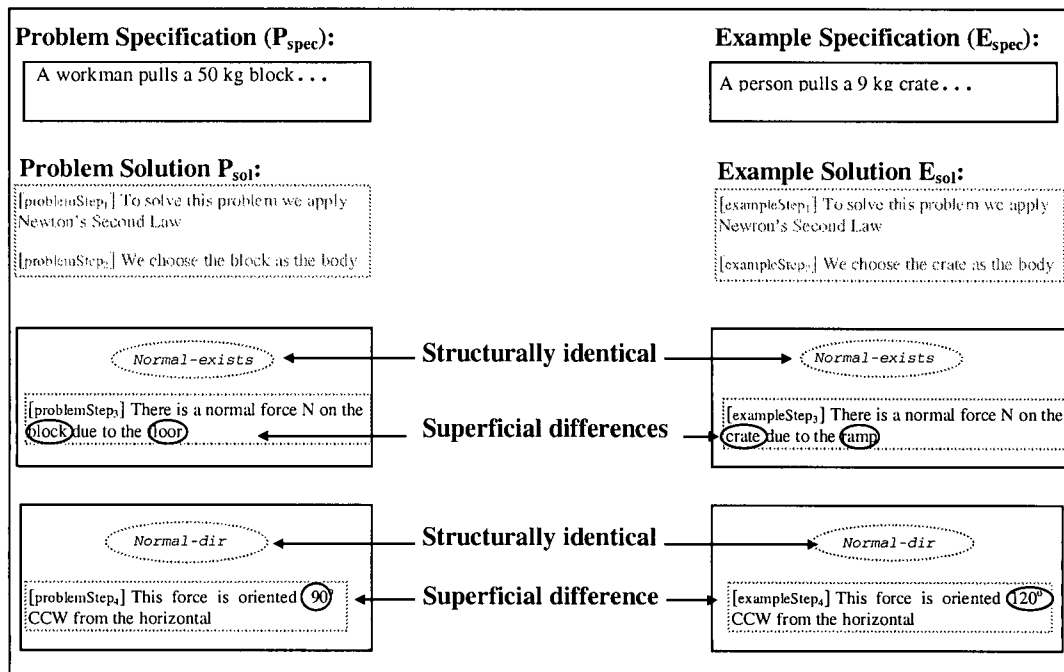


Figure 2-7: Illustration of superficial & structural relations

The key question is how these various superficial differences should be classified and in particular, whether they should all be considered 'equal'. Our approach entails classifying the differences according to how easily they can be resolved: whether they require in-depth reasoning such as EBLC or whether they allow the problem solution to be generated by copying from the example. As mentioned in section 2.1.2.2, copying involves transferring the example solution over to the problem with no changes, or with 'minor' changes, including

- i) replacing problem specific constants with ones needed to solve the problem via transformational analogy (see section 2.1.2.2);
- ii) changing the order of terms in the equations of the transferred example step;
- iii) minor omissions, such as leaving out brackets;
- iv) choosing to use different variable names to denote problem solution elements (e.g., see '*N*' in exampleStep₃ of the example in figure 2-7).

Notice that modification (i) above corresponds to resolving a problem/example difference *required* to generate a correct problem solution. If a student is successful at resolving the difference, then he can generate the problem solution by copying. In contrast, the other three modifications (ii-iv) are changes a student *chooses* to introduce but that are not required in order to generate a correct problem solution. Since the first type of modification allows the problem solution to be generated by copying, it is critical to identify superficial differences that can be resolved in this manner.

Returning to our scenario, the two superficial differences between problemStep₃ and exampleStep₃ can easily be resolved by transformational analogy. Recall that transformational analogy relies on (1) devising a mapping between the example and problem specifications and (2) using the mapping to substitute example-specific constants by ones needed for the problem solution [Anderson, 1993; VanLehn, 1998]. Because both differences between problemStep₃ and exampleStep₃ correspond to constants that appear in the problem/example specifications, transformational analogy is enabled to resolve the differences, allowing a student to generate the problem step by copying from the example. We classify such differences as *trivial*. In contrast, the superficial differences between problemStep₄ and exampleStep₄ can not be easily be resolved by

transformational analogy because the example constant corresponding to the difference is *missing* from the problem/example specifications, and so a student cannot generate the necessary mapping between the specifications. Thus, the student needs to perform more in-depth reasoning. We classify such differences as *non-trivial*.

The subsequent section presents formal definitions corresponding to our classification, which are the formalisms the EA-Coach relies on to assess problem/example similarity, as we will see in subsequent chapters.

2.2.1.1 Formal Definitions

A problem and an example consists of (1) a specification and (2) a solution. We represent both specifications and a solutions as a sequence of elements e_1, e_2, \dots, e_n . Each element e_i (for $1 \leq i \leq n$) is a vector of the form $\langle type_i, slot_{i,1}, \dots, slot_{i,k_i} \rangle$ where:

- $type_i$ identifies the type of element e_i
- $slot_{i,j}$ (for $1 \leq j \leq k_i$) is the pair $(a_{i,j}, c_{i,j})$, where $a_{i,j}$ is the slot's attribute and $c_{i,j}$ is the constant assigned to that attribute

For a problem P , we denote its specification as P_{spec} and its solution as P_{sol} . Similarly, for an example E , we denote its specification as E_{spec} and its solution as E_{sol} . In the following, we will refer to

- elements in a problem or example solution as *steps*, where each problem/example step is generated by a domain principle that dictates the form of the step's vector representation;
- a solution step in P_{sol} as *problemStep* and a slot in problemStep as *problemSlot*;
- a solution step in E_{sol} as *exampleStep* and a slot in exampleStep as *exampleSlot*.

Given the following:

P_{sol} : problemStep₁, problemStep₂, ..., problemStep_n

E_{sol} : exampleStep₁, exampleStep₂, ..., exampleStep_m

we define two steps, *problemStep_i* and *exampleStep_j* (for $1 \leq i \leq n$ and $1 \leq j \leq m$), to be **structurally different** if they are not generated by the same domain principle and **structurally identical** otherwise.

Now let's consider two structurally identical steps $problemStep_i$ and $exampleStep_j$. To simplify subsequent discussion, we drop the subscripts i and j and refer to the steps as $problemStep$ and $exampleStep$. When two steps are structurally identical, their vectors are of the same type and length:

$$problemStep = \langle type\ problemSlot_1, \dots, problemSlot_k \rangle$$

$$exampleStep = \langle type\ exampleSlot_1, \dots, exampleSlot_k \rangle$$

In addition, their *corresponding* slots (i.e., slots appearing at the same position in the vector) have identical attributes:

$$problemSlot_h = (a_h, c_h), exampleSlot_h = (a_h, c'_h) \text{ for } 1 \leq h \leq k$$

However, even though steps $problemStep$ and $exampleStep$ are structurally identical, they may include some superficial differences between their respective *slots*. Specifically, two corresponding slots $problemSlot = (a, c)$ and $exampleSlot = (a, c')$ are *superficially different* if $c \neq c'$. The difference is defined as:

1. *trivial* if
 - i. example constant c' is in an element e in E_{spec} and
 - ii. problem constant c is in an element p in P_{spec} and
 - iii. problem and example specification elements e and p are of the same type²
2. *non-trivial* otherwise

To illustrate the formal definition, figure 2-8 shows the problem/example pair in figure 2-7 supplemented with the formal representation of the problem/example specifications and steps. As we have already pointed out, although both $problemStep_3$ and $problemStep_4$ are structurally identical to the corresponding example steps $exampleStep_3$ and $exampleStep_4$, they are superficially different from them. In section 2.2.1, we characterized the superficial differences as trivial or non-trivial; we will now do so by

² Recall that as we defined above, specification elements are vectors of the form $\langle type_i, slot_{i,1}, \dots, slot_{i,k_i} \rangle$

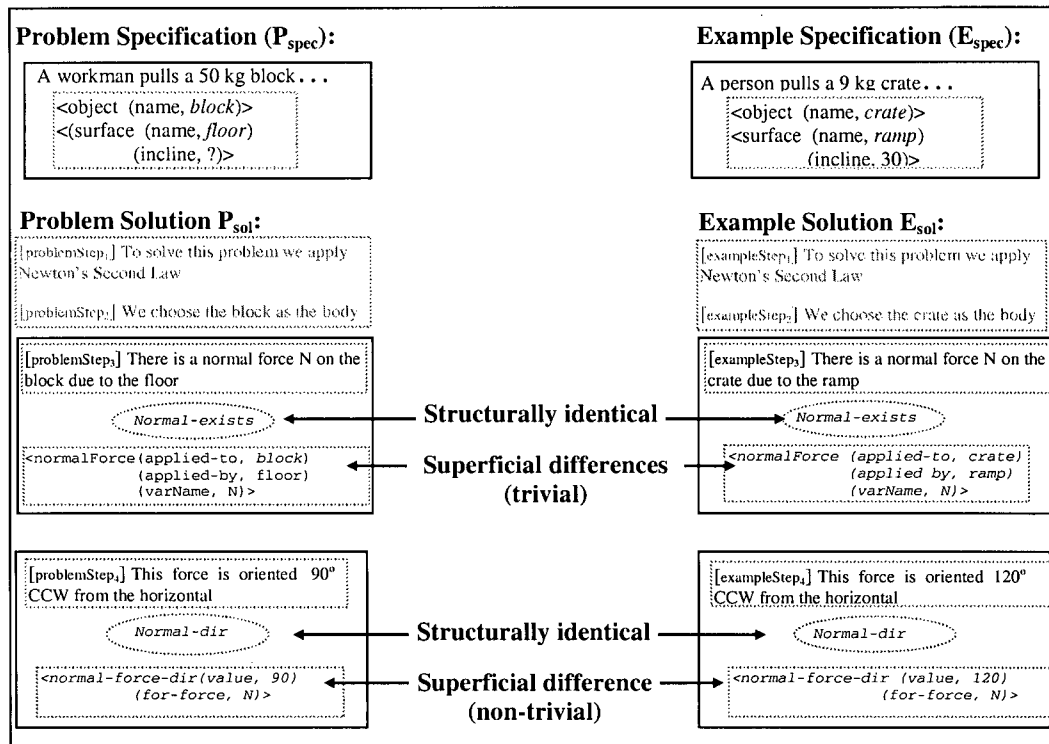


Figure 2-8: Classification of superficial & structural relations from figure 2-7 supplemented with formal representation

relying on the formalisms introduced above. *ProblemStep₃* and *exampleStep₃* include two trivial differences:

- A trivial difference corresponding to the objects the normal force is acting on, *crate* and *block* (see exampleSlot (applied-to, *crate*) and problemSlot (applied-to, *block*) in figure 2-8). The difference is trivial because the example constant *crate* appears in the example specification (see `<object (name, crate)>` in E_{spec} , figure 2-8) and has a corresponding constant in the problem specification (see `<object (name, block)>` in P_{spec} , figure 2-8).
- A trivial difference corresponding to the objects applying the normal force, *ramp* and *floor* (see exampleSlot (applied-by, *ramp*) and problemSlot (applied-by,

floor) in figure 2-8). The difference is trivial because the example constant *ramp* appears in the example specification (see <surface (name, **ramp**) (incline, 30)> in E_{spec} , figure 2-8) and has a corresponding constant in the problem specification (see <surface (name, **floor**) (incline, ?)> in P_{spec} , figure 2-8).

On the other hand, *problemStep₄* and *exampleStep₄* include one non-trivial difference:

- The difference corresponds to a constant specifying the direction of the normal force (see exampleSlot (value, **120**) and problemSlot (value, **90**) in figure 2-8). The difference is non-trivial because the constant is missing from the problem/example specifications (i.e., in figure 2-8, the constant *120* does not appear in the example specification; the constant *90* does not appear in the problem specification).

2.2.2 Impact of Differences: Hypotheses

Given our classification of superficial and structural relations, the key question is: what impact do various kinds of differences have on APS behaviors and subsequent problem-solving success and learning? We argue that this impact depends on the students' APS meta-cognitive skills and domain knowledge.

If the problem and example are structurally different with respect to a problem step, then the student cannot rely on the example to derive it, i.e., transfer of the step from the example is blocked. This hinders both problem solving and learning if the student lacks the knowledge to generate the problem step on his own. If, on the other hand, the student does have the knowledge to generate the step on his own, then the structural difference encourages min-analogy by blocking copying, which will help to solidify the student's knowledge of the corresponding rule.

In contrast, superficial differences between structurally identical problem/example steps do afford students the opportunity to rely on the example to derive the problem step, because the two steps are derived by the same rule. However, as we already pointed out in Section 2.2.1, what distinguishes trivial vs. non-trivial differences is how easily they may be resolved in order to apply the example step to generate the problem step. Because

trivial differences are easily resolved by transformational analogy, they encourage copying for students with poor APS meta-cognitive skills. Although copying increases the chances the student will generate the problem solution, it does not stimulate min-analogy or EBLC for students with poor domain knowledge and poor meta-cognitive skills, and thus hinders learning.

On the other hand, because non-trivial differences are not easily resolved by transformational analogy, they may encourage min-analogy and EBLC for students with poor meta-cognitive skills and poor domain knowledge. To see how this could occur, let's consider the non-trivial difference between *exampleStep₄* and *problemStep₄* in figure 2-8. Since the direction of the normal force in *exampleStep₄* does not appear in the example specification, a student cannot generate the problem solution step by copying from the example, because he cannot rely on transformational analogy to resolve the difference between the problem/example steps. Consequently, if the student knows the *Normal-dir* rule needed to generate this step, then the non-trivial difference may encourage him to generate the step via the application of his own knowledge, i.e., to engage in min-analogy. If, on the other hand, the student does not know the *Normal-dir* rule, then the non-trivial difference may encourage him to explain through EBLC how the example step was generated, i.e., to learn the rule. If the student is successful at learning the rule, then he can apply the newly-acquired knowledge to generate the corresponding solution step, thereby resolving the difference between the problem and example on this step.

Our hypotheses on problem/example differences presented in this section are embedded into the EA-Coach student model's operation, which we describe in detail in chapter 5.

2.3 Summary

In this chapter, we first described the cognitive science background on APS that has shaped our design of the EA-Coach. We then presented our own hypotheses on how problem/example similarity and student characteristics impact APS outcomes. In the next

chapter, we will describe a series of pilot evaluations that helped us gain some preliminary insight as to whether our hypotheses are generally appropriate.

Chapter 3

Pilot Evaluations

In this chapter, we describe a series of pilot evaluations, including a primary pilot as well as two follow-up pilots. The goal of the primary pilot was to gather some first-hand observations on how the various types of superficial differences defined in our classification influence APS, in order to make sure that we were on the right track in terms of our hypotheses presented in the previous chapter. In particular, we wanted to compare how students solved problems in the presence of examples that included trivial vs. non-trivial differences with respect to the target problem. The primary and two follow-up pilot evaluations were also used to inform interface design decisions for the EA-Coach interface.

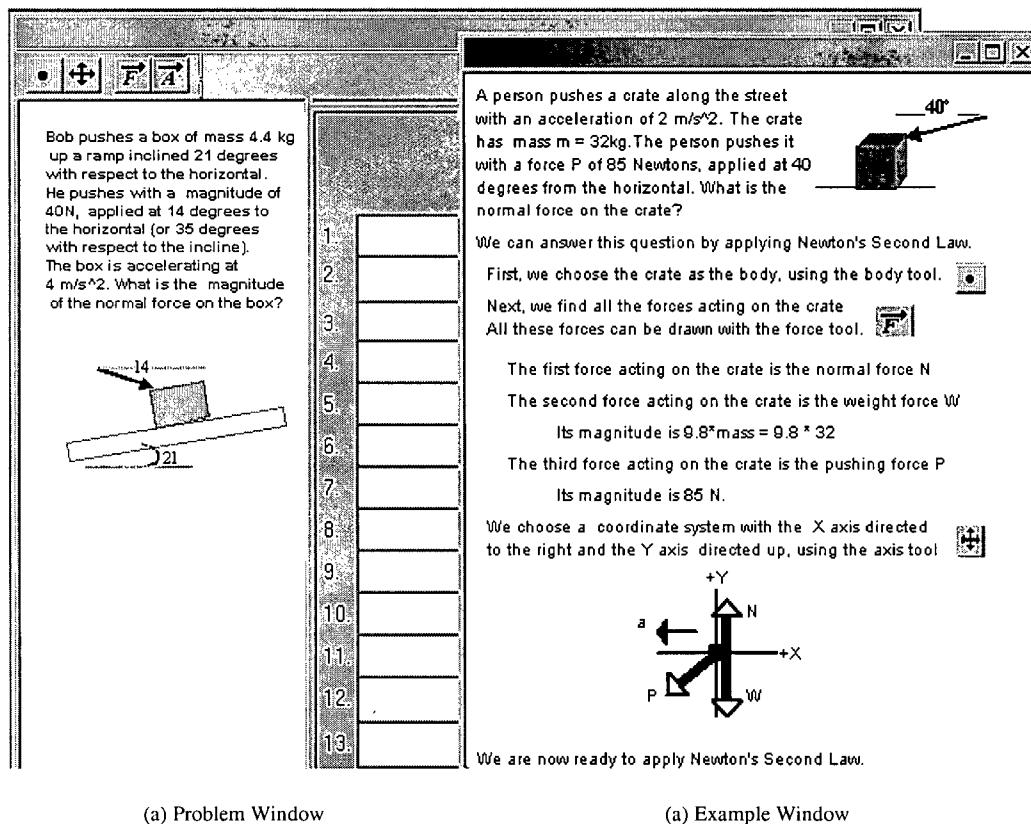
3.1 Primary Pilot Evaluation

The primary pilot evaluation involved students using a preliminary version of the EA-Coach interface. Before we provide details on the pilot, we will describe this interface.

3.1.1 Preliminary EA-Coach Interface

The preliminary EA-Coach interface included a problem and an example window, used to enter the problem solution and access an example, respectively (see figure 3-1a and figure 3-1b). The windows were based on windows in two existing ITS supporting learning of Newtonian physics, namely Andes [Conati, Gertner et al. 2002] and the SE-Coach [Conati and VanLehn 2000]. Andes supports pure problem solving without providing access to examples, while the SE-Coach supports pure example studying without providing problems. Since the EA-Coach supports problem solving *with* access to examples and the Andes and SE-Coach interfaces have undergone extensive usability testing, we intended to base the EA-Coach's interface on their respective designs for the problem and example windows.

The problem-solving window in the preliminary interface (see figure 3-1a) included two panels for entering the problem solution. The *Free-Body* panel (see left panel in figure 3-1a) allowed students to draw free-body diagrams via the provided tools (see *Free-Body* toolbar above the problem statement in figure 3-1a). The tools included (1) the *body* tool, used to select the body to apply Newton's Law to, (2) the *axes* tool, used to draw the axes, (3) the *force* tool, used to draw the forces and (4) the *acceleration* tool, used to draw acceleration vectors. The *Equation* panel (see bottom-right panel in figure 3-1a) allowed students to type equations via the keyboard. The window's design was directly based on the Andes design, the key exception being that the Andes interface provides feedback for correctness (realized by coloring students' solution entries red or green for incorrect and correct entries, respectively). The preliminary EA-Coach interface did not provide such feedback, for two main reasons. First, although cognitive science work suggests that students are not always effective at diagnosing their own errors [Chi, Bassok et al., 1989], we wanted to gather some first-hand information on whether students diagnosed errors in their solutions. Second, we wanted to investigate APS behaviors in a context that was similar to existing cognitive science research (e.g., [VanLehn, 1998]), which does not provide such feedback.

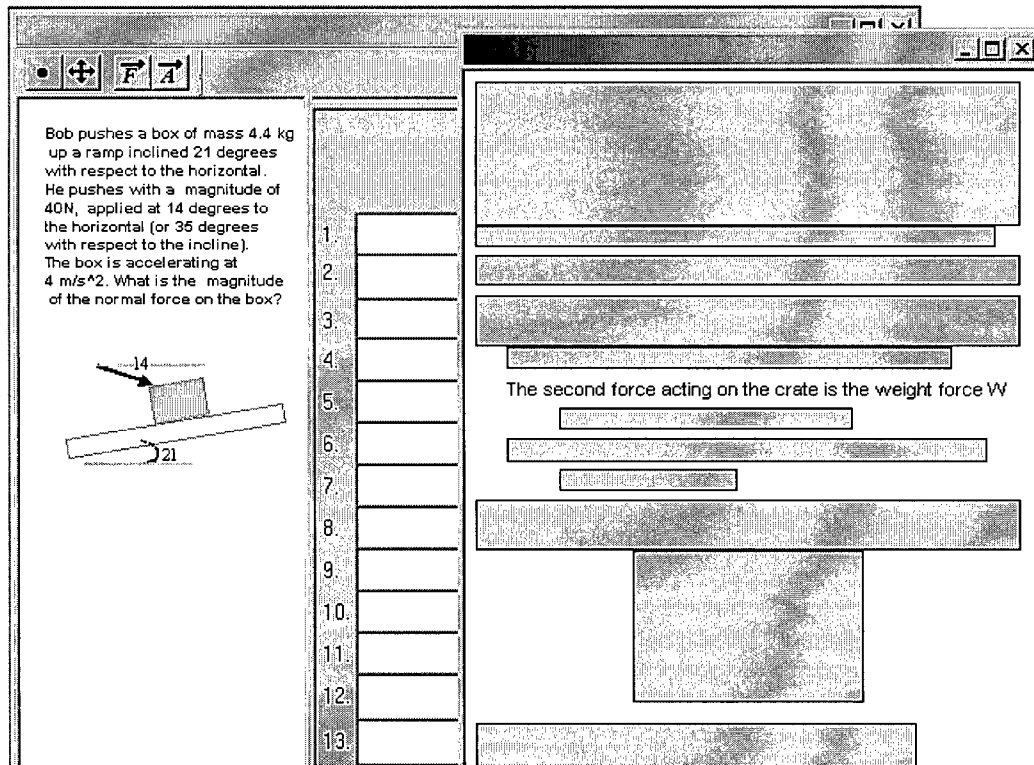


(a) Problem Window

(b) Example Window

Figure 3-1: Preliminary EA-Coach interface (shown without the masking interface, which is displayed in Figure 3-2)

The example-viewing window in the preliminary EA-Coach interface (see figure 3-1b) was directly based on the SE-Coach's design. The example was presented using a format loosely based on the one used in physics textbooks. To provide information on which example steps students viewed, the example window included the so-called *masking interface*, following the SE-Coach design (see figure 3-2; note that the masking interface is not shown in figure 3-1). The masking interface covered the example specification and solution steps. Moving the mouse over a region in the masking interface uncovered the region and covered whatever region was previously uncovered. A region remained uncovered until the mouse was moved over a new region.



a) Problem Window

a) Example Window

Figure 3-2: Masking interface

3.1.2 Primary Pilot Evaluation Particulars: Participants, Methodology and Materials

3.1.2.1 Participants

The participants were 8 first-year university students who were in the process of taking or had completed Physics 100 at UBC (the equivalent to a high-school grade-twelve physics class that provides an introduction to Newtonian physics of the type targeted by the EA-Coach). Participants were paid ten dollars per hour.

3.1.2.2 Methodology

Each participant

1. completed a physics pencil and paper pre-test (Appendix 2);
2. solved two physics problems (*experimental phase*);
3. completed a physics pencil and paper post-test (Appendix 2).

For all three phases, subjects were told to take as much time as they needed, with the typical session lasting approximately 1.5 hours.


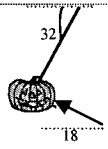
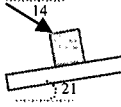
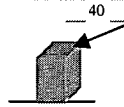
During the experimental phase, for each problem that subjects were asked to solve, they had access to a corresponding worked-out example (see table 3-1). Students selected the problem they wished to work on, which automatically opened the corresponding example. Once students closed a problem, which also automatically closed the corresponding example, they were not allowed to re-open it. To enter solutions and access examples, subjects used the preliminary version of the EA-Coach interface described in section 3.1.1.

For each problem, the corresponding example was hand-selected before run-time (i.e., not selected by the system), and all students saw the same problem/example pairs (see table 3-1). To explore the impact of similarity on students' APS behaviors, the superficial similarity between the problem and its corresponding example was manipulated for the problem/example pairs, as follows:

- [*trivial condition*] problem_{trivial} and its example only included trivial differences (summarized in table 3-3, section 3.1.2.3)
- [*non-trivial condition*] problem_{non-trivial} and its example included both trivial and non-trivial superficial differences (summarized in table 3-4, section 3.1.2.3)

In the pilot evaluation, each subject was exposed to both conditions (i.e., a within-subject design was used). We chose this design because it increases the ability to detect differences between the conditions, by accounting for the variability between subjects. The variability arises from differences between subjects in expertise, APS tendencies and verbal expression of self-explanation (which we captured).

Table 3-1: Problems & examples used in primary pilot

Problem _{trivial}	Corresponding Example
<p>A cat pushes his toy, a mouse of mass 2.3 kg which is hanging from a string, with a magnitude of 31 N, applied at an angle of 16 degrees. The string is now at an angle of 26 degrees to the horizontal as shown. What is the tension force in the string?</p> 	<p>A pumpkin of mass 2.21 kg is suspended by a cord and pushed with a 24 N force until the cord forms an angle of 32 degrees with the horizontal (the pushing force is applied at an angle of 18 degrees to the horizontal). What is the tension force in the cord?</p> 
Problem _{non-trivial}	Corresponding Example
<p>Bob pushes a box of mass 4.4 kg up a ramp inclined 21 degrees with respect to the horizontal. He pushes with a magnitude of 40N, applied at 14 degrees to the horizontal (or 35 degrees with respect to the incline). The box is accelerating at 4 m/s². What is the normal force on the box?</p> 	<p>A person pushes a crate along the street with an acceleration of 2 m/s². The crate has mass $m = 32\text{kg}$. The person pushes with a magnitude of 85N, applied at 40 degrees to the horizontal. What is the normal force on the crate?</p> 

During the study, we used two types of data collection techniques. First, to capture students' reasoning processes, we used the talk-aloud method and asked students to verbalize their thoughts during the experimental phase of the study [Ericsson and Simon, 1980]. We video-taped and subsequently transcribed all sessions. Second, we logged all student problem-solving and example-viewing actions in the EA-Coach interface.

3.1.2.3 Materials

The problems and examples used in the primary pilot were based on typical "Newton's Second Law" problems used in physics courses (as we identified through on-line searches/textbooks, e.g., [Halliday and Resnick, 1988]). The problems we selected (problem_{trivial} and problem_{non-trivial} in table 3-1) were intended to be equivalent in terms of difficulty, because we did not want problem type (problem_{trivial} vs. problem_{non-trivial}) to obscure the results by having an impact on the measures of interest (described in section 3.1.3). However, we also wanted the problems to be somewhat different, because it does not make sense to ask students to solve two very similar problems, for two reasons. First, students would likely not be motivated to generate the solution to two very similar

Table 3-2: Structural differences between problem_{trivial} and problem_{non-trivial}

Problem_{trivial}	Problem_{non-trivial}
Normal force exists, acceleration vector exists, parallel-vector component equation.	Tension force exists, acceleration is null.

problems. Second, learning effects between two very similar problems could be so high as to obscure any other effects. To balance these two constraints, the two problem solutions were similar (e.g., required the application of ‘Newton’s Second Law’, had the same number of steps, had many structurally identical steps between the two problems, i.e., corresponded to the same rules) but also had several structural differences (summarized in table 3-2). The structural differences were present because it is very difficult to satisfy the second constraint without introducing some of these differences.

The superficial differences between the problem/example pairs are summarized in table 3-3 and table 3-4.

Table 3-3: Superficial differences between problem_{trivial} & corresponding example steps

Simplified fragment of solution step corresponding to the difference	Problem_{trivial} Solution Step	Corresponding Example Solution Step	Type of Difference
Body to apply Newton’s 2nd Law	mouse	pumpkin	Trivial
There is a pushing force on the ...	mouse	pumpkin	Trivial
The pushing force magnitude is...	31N	24N	Trivial
There is a weight force on the ...	mouse	pumpkin	Trivial
The magnitude of weight force is 9.8 *...	2.3kg	2.21 kg	Trivial
There is a tension force on the ...	mouse	pumpkin	Trivial
The x-component equation for the tension force is	$T \cdot \cos(26)$	$T \cdot \cos(32)$	Trivial
The x-component equation for the pushing force is	$-P \cdot \cos(16)$	$-P \cdot \cos(18)$	Trivial

Table 3-4: Superficial differences between problem_{non-trivial} & corresponding example steps

Simplified fragment of solution step corresponding to the difference	Problem _{non-trivial} Solution Step	Corresponding Example Solution Step	Type of Difference
Body to apply Newton's 2nd Law	box	crate	Trivial
The acceleration is....	4 m/s ²	2 m/s ²	Trivial
There is a pushing force on ...	box	crate	Trivial
The pushing force direction (drawn in free-body diagram)	--	--	Non-Trivial ³
The pushing force magnitude is ...	40 N	85 N	Trivial
There is a weight force on the ...	box	crate	Trivial
The magnitude of weight force is 9.8 *...	4.4 kg	32 kg	Trivial
There is a normal force on ...	box	crate	Trivial
The normal force direction (drawn in free-body diagram)	--	--	Non-Trivial
The acceleration direction (drawn in free-body diagram)	--	--	Non-Trivial
The axis is inclined ⁴	21 degrees	0 degrees	Non-Trivial
The y-component equation of the weight force is	-W cos (21)	-P cos (50)	Non-Trivial
The y-component equation of the pushing force is	-P cos (55)	-P cos (50)	Non-Trivial

³ Note that this entry was not classified as a difference in the trivial condition. For the primary pilot, we hand-classified the superficial differences. In the trivial condition, the pushing force direction appeared virtually identical in the problem/example free-body diagrams and so it was not classified as a difference (students were not required to enter the value corresponding to a force direction explicitly, i.e., they could simply draw the force in the free-body panel without specifying the angle). On the other hand, in the non-trivial condition, the pushing force direction does appear quite different and so was classified as such.

⁴ Students may choose several different axis configurations – the data in the table assumes a slanted axis is chosen in the problem solution because this is what the rule in the Knowledge Base for optimal axis selection suggests: to minimize the number of vectors that need to be decomposed into components.

3.1.3 Primary Pilot Evaluation: Results

To analyze the impact of example similarity on APS, we considered the following dependent measures in the data analysis:

- *Task time*: the time students took to generate a problem solution
- *Errors*: the number of errors in a final problem solution
- *Copy Events*: the number of copy episodes for a problem solution
- *Self-explanation Events*: the number of self-explanations expressed while generating a problem solution

Since the pilot involved a small subject pool, we limited the statistical analysis to obtaining means/standard deviations, which would provide a sense of the relevant trends. We describe the results of this analysis below, and summarize them in table 3-5.

Task Time. As the data in table 3-5 shows, on average students spent similar amounts of time to generate a problem solution in both conditions (14min., 8sec. in the trivial condition vs. 14min., 28 sec. in the non-trivial condition).

Errors. To see how successful students were in problem solving, we analyzed their final solutions. For each solution, we checked to see (1) if the solution was *complete* in that it had all the necessary parts (e.g., all forces were specified, Newton's Second Law was applied to the corresponding components, etc.) and (2) if the solution elements contained any errors. When checking for errors, solution steps that contained errors corresponding to the same misconception were only counted once. Minor typographical errors were not counted as errors (e.g., if variable names were slightly inconsistent, such as if students used Ax to specify the acceleration component in one equation, but A_x to do so in another equation).

Note that we did not analyze the number of errors a student generated *while* producing a problem solution. This is because we did not indicate to students that they had to finish working on a given entry (i.e., make sure it was correct) before moving on to generate another entry. Thus, students would sometimes enter a portion of a solution step, start generating another step, and then return to the original step to finish it. This made it

Table 3-5: Summary of data analysis for primary pilot

	Trivial Condition (mean, std. dev.)	Non-trivial Condition (mean, std. dev.)
Task Time	14min, 8sec (<i>4 min, 31sec</i>)	14min, 28sec (<i>3 min, 38sec</i>)
Errors	0.63 (<i>1.06</i>)	1.75 (<i>2.05</i>)
Copy Events	3.5 (<i>4.03</i>)	2.5 (<i>3.5</i>)
Self-explanation Events	1 (<i>2.07</i>)	0.75 (<i>1.16</i>)

impossible to determine the correctness of a given step until students declared that they was done with the target problem.

Although all problem solutions in both conditions were complete, on average the problem solution in the trivial condition contained fewer errors than the problem solution in the non-trivial condition (0.63 vs. 1.75, respectively, table 3-5). The difference in the number of errors between the two conditions was due to the data of four subjects. The other four subjects had no errors in their solutions – these were also the subjects who had answered all pre-test questions correctly. A key difference between these two groups of subjects (denoted as successful vs. unsuccessful from now on) is that the unsuccessful subjects provided no indication of being aware of the errors. A second difference is that the unsuccessful subjects were also the only ones who copied or self-explained from examples. A likely explanation for this latter finding is that the successful subjects had the necessary knowledge to generate the solution, as suggested by their pre-test performance, and so did not need to rely on examples (they also likely had a strong min-analogy tendency, as suggested by the lack of copying). Since we wanted data on how students copied/self-explained from examples, we restricted the remainder of the analysis to the unsuccessful participants. To provide the reader with a better sense of the variability among the unsuccessful subjects, table 3-6 shows the results using data only from these four participants.

Table 3-6: Summary of analysis for the four subjects who copied/self-explained from examples

	Trivial Condition (mean, std. dev.)	Non-Trivial Condition (mean, std. dev.)
Task Time	13min, 31sec (<i>4 min, 37sec</i>)	16min, 35sec (<i>2 min, 13sec</i>)
Errors	1.25 (<i>1.3</i>)	3.5 (<i>1.2</i>)
Copy Events	7 (<i>2.31</i>)	5 (<i>3.46</i>)
Self-explanation Events	2 (<i>2.71</i>)	1.5 (<i>1.29</i>)

Copy Events. Recall that as we indicated in section 2.2.1, copying involves transferring content from an example's solution over to the problem with no changes or minor changes. To recognize copy events, we used the log files to identify (1) which steps students accessed in the example solution and (2) whether these steps were subsequently copied. To identify which example steps students accessed, we primarily relied on the information provided by the masking interface, since students were not always reliable in verbalizing example solution steps they accessed. To identify if accessed example solution steps were subsequently copied, we checked for correspondences between accessed steps and students' subsequent input to the problem solution. Problem entries that were identical to accessed example steps or that shared minor differences of the type listed in section 2.2.1 were flagged as 'copied'. In our coding scheme, access to an example solution step corresponded to at most one copy event (i.e., each term in the step was not counted separately). To decide whether a given problem-solving entry was copied, we only considered recently viewed example steps, i.e., the last *three* example steps accessed before a given problem entry. Although it is conceivable that we missed some copy events because students may have remembered and copied example steps that were accessed earlier, we did not see indications of this in our study. In fact, students who relied on copying from examples tended to do so on a line-by-line basis (i.e., uncover an example step, copy it, uncover the next example step, copy it, etc.), in a fashion similar to that described in VanLehn's study [VanLehn, 1998]. We did not consider only the *last* step accessed in the example, for two reasons. First, when students

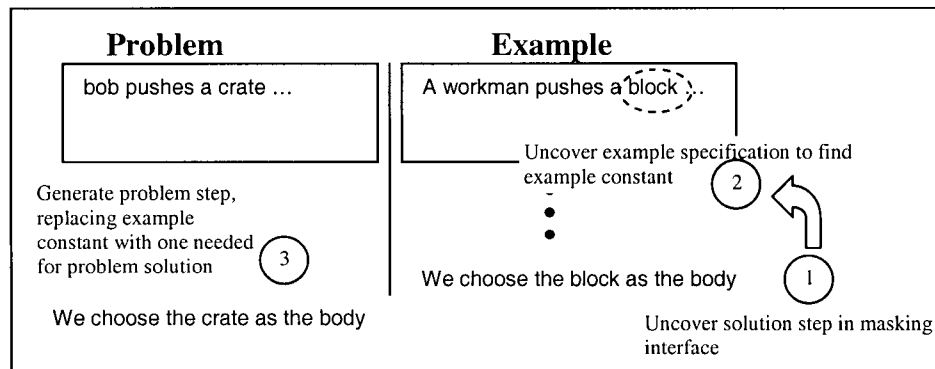


Figure 3-3: Sample viewing sequence in masking interface; two example regions are uncovered: (1) example solution step and (2) example specification.

accessed an example step, students sometimes inadvertently uncovered a line or two in the example as they were moving the mouse over to the problem window to copy the step. If we only considered the last accessed step, we would have missed the copy event. Second, after viewing a given example solution step, students would sometimes refer to the example specification or free-body diagram to identify example-specific constants, thereby uncovering additional steps that were related to the copied step. For instance, figure 3-3 shows a viewing sequence where a student (1) uncovered an example step stating that the *block* is chosen as the body, (2) uncovered the example specification to find the constant *block* and (3) copied the example step, replacing the example constant *block* by the one needed for the problem solution, *crate*.

As table 3-6 shows, on average students copied more in the trivial condition than the non-trivial condition (on average, 7 vs. 5 copy events respectively).

Self-Explanation Events. Since EBLC is a form of self-explanation, for the purposes of the pilot study we focused on identifying self-explanations, without classifying whether they involved common sense or overly general reasoning characteristic of EBLC. This decision is based on the fact that self-explanation is a highly constructive activity that correlates strongly with positive learning outcomes [Chi, Bassok et al., 1989; Bielaczyc, Pirolli et al., 1995; Renkl, 1997; Renkl, 1999; Chi 2000]. Therefore, we felt that analyzing all possible self-explanations would provide insight into the impact of

similarity on how they generate explanations during APS. As defined in [Chi, Bassok et al., 1989], self-explanations are

“any comments that pertain to physics content but are not paraphrases. Thus, self-explanations infer some additional pieces of information, regardless of how minute they are.”

We relied on this definition to identify self-explanations in the verbal protocols (e.g., see figure 3-4). We considered utterances expressed both during example studying and problem solving. Utterances related to the same concept were only counted separately if they were separated by problem-solving entries or explanations pertaining to other concepts. Utterances that were not considered self-explanations included

- paraphrases, such as *“the x component of W is zero”* from $W_x = 0$ example equation;
- self-monitoring statements, such as *“I’m confused”*;
- statements related to transformational analogy that *only* mentioned syntactic mappings without mentioning any physics content, such as *“my mouse is like their block”*.

As table 3-6 shows, on average, students generated slightly more self-explanations in the trivial condition than in the non-trivial condition (on average, 2 vs. 1.5 respectively).

<p><i>“w_x is zero because it doesn’t have an x component”</i></p> <p>Expressed after reading $w_x = 0$ in the example corresponding to problem_{trivial}</p>
<p><i>“it’s negative because it is in the negative direction”</i></p> <p>Expressed after reading $P_y = -P \cos(50)$ in the example corresponding to problem_{nonTrivial}</p>

Figure 3-4: Sample self-explanations

3.1.4 Primary Pilot: Discussion

We will limit our discussion of the primary pilot evaluation to the four students who copied and/or self-explained from examples during problem solving. These students had similar task times in the two conditions. However, in the trivial condition, on average students copied more than in the non-trivial condition. An explanation for this finding is related to the *type* of superficial differences between the problem/example pairs in the two conditions. Recall that in the trivial condition, problem_{trivial} only had trivial superficial differences with the corresponding example. As discussed in chapter 2, this type of difference allows students to generate the problem solution by copying from the corresponding example. In contrast, in the non-trivial condition, the problem/example pair had some non-trivial superficial differences, which blocked copying of the corresponding steps. Students appeared to realize this, as is supported by the fact that over all the students, only two non-trivially different example steps were copied (from a total of 20 copy episodes; the remaining copy episodes corresponded to example steps that included only trivial differences or no differences with the problem). As one subject expressed: *“this one (points at example in the trivial condition) related a lot more so I was able to use all of it but the other one – I just took little bits from”*. Ironically, although this subject felt the example in the trivial condition “related more”, the behavior imposed on her by the example in the non-trivial condition was more desirable, since it discouraged copying of the entire solution. Her behavior supports her claim, in that she copied the majority of her solution in the trivial condition, but only a few steps in the non-trivial condition (nine vs. two copy events, respectively; the two copied entries in the non-trivial condition corresponded to steps with trivial differences or no differences with the problem solution).

The fact that students copied less in the non-trivial condition increased the number of errors in that condition, when students tried to generate the solution on their own but did not have the necessary domain knowledge. As is summarized in table 3-7, out of all the errors, 68.4% were due to entries students self-generated without copying during problem solving: while in the trivial condition, only 15.8% of errors were the result of incorrectly

Table 3-7: Source of errors in primary pilot evaluation: copying vs. problem-solving

	Trivial	Non-trivial	Total
% of errors from problem-solving without copying	15.8% (3/19)	52.6% (10/19)	68.4% (13/19)
% of errors from incorrectly copied steps	10.5% (2/19)	21.1% (4/19)	31.6% (6/19)
Total	26.3% (5/19)	73.7% (14/19)	100% (19/19)

generated steps, in the non-trivial condition, a higher percentage of errors, i.e., 52.6%, corresponded to incorrectly generated steps. A second source of errors was the result of students *incorrectly* copying from the example, which was slightly higher in the non-trivial condition. As is summarized in see table 3-7, out of all the errors, 31.6% were due to incorrectly copied steps; of these, 10.5% came from incorrectly copied steps in the trivial condition, and 21.1% came from incorrectly copied steps in the non-trivial condition. In the trivial condition, all of the incorrectly copied entries (2 from a total of 28 copied entries) corresponded to steps that included trivial differences or no differences with the problem step and were the result of slips (i.e., a subject would neglect to copy a necessary portion of an equation)⁵. In the non-trivial condition, half of the incorrectly copied entries corresponded to non-trivial differences (2 of the 4 incorrectly copied steps, from a total of 20 copy events); the other half of the incorrectly copied entries corresponded to slips.

Above, we argue that the superficial similarity influenced students' success in generating a problem solution, because it influenced how much they copied and how successful they were at doing so. However, there are two alternative explanations deriving from the fact

⁵ All of the correctly copied entries corresponded to example steps that included trivial differences or no differences with the problem step.

that the design of the pilot evaluation was not fully counterbalanced, although our data suggests that these are not likely. One of these alternative explanations is that students found the problem in the non-trivial condition more difficult than the problem in the trivial condition. Unfortunately, the assignment of problems to conditions was not counter-balanced, making this a confounding variable in the analysis (i.e., $\text{problem}_{\text{trivial}}$ was *always* assigned to the trivial condition, while $\text{problem}_{\text{non-trivial}}$ was *always* assigned to the non-trivial condition). To shed some light on this issue, we checked how many of the students' errors were due to structural differences between the problems (i.e., differences in the domain knowledge needed to generate the solution). We found that only one error in the non-trivial condition was related to a structural difference between the two problems (a subject incorrectly drew a normal force). This suggests that the difference in the number of errors between the two conditions was not likely due to the differences between the problems. The second alternative explanation for the findings is related to the lack of counterbalancing of the study conditions (recall that students were free to select the order in which they solved the two problems). Thus, the difference in the number of errors could have been due to *learning*. Specifically, if most students solved the problem in the non-trivial condition first (i.e., $\text{problem}_{\text{non-trivial}}$) and learned from doing so, this could have helped reduce errors when they solved the subsequent problem in the trivial condition ($\text{problem}_{\text{trivial}}$). To see if this was the case, we checked the order in which students solved the two problems. We found that three of the four subjects solved $\text{problem}_{\text{trivial}}$ first. Thus, if learning was the cause of the difference, students should have had fewer errors in the $\text{problem}_{\text{non-trivial}}$'s solution, which was not the case.

To summarize, although further investigation is needed because of the small number of subjects, the results from this pilot evaluation gave us an initial indication that our hypotheses on the impact of problem/example similarity on copying are generally appropriate. In particular, we hypothesized that non-trivial superficial differences have better potential to encourage min-analogy by discouraging copying, compared with trivial differences. The pilot results showed encouraging trends that this occurs. Since min-analogy is beneficial for learning, this finding is positive. On the other hand, in our pilot evaluation non-trivial differences lowered students' problem-solving success. This

suggests that if students do not have all of the appropriate domain knowledge needed to generate a correct problem solution, then non-trivial differences can hinder problem completion. In addition, contrary to what we hypothesized, non-trivial differences did not encourage students to self-explain. We believe that one likely explanation for both of these findings, however, is related to the lack of feedback for correctness in the pilot. As we already pointed out, students did not seem aware of the errors in their final problem solutions. Feedback for correctness would make the errors explicit and so encourage students to fix them, which in turn would help them achieve problem-solving success. The process of fixing the errors could trigger students to fill knowledge gaps through self-explanation. Since students had more errors in the non-trivial condition, this condition had higher potential than the trivial condition to trigger self-explanation. Therefore, helping students achieve problem-solving success could be a way of encouraging them to learn the necessary domain principles. To summarize, feedback for correctness has the potential to trigger both learning and problem-solving success. Thus, as our next step, we incorporated such feedback into the EA-Coach. The feedback followed the Andes design and was realized by coloring student problem-solving entries green or red in the interface (for correct and incorrect entries, respectively).

3.2 Two Follow-up Pilot Evaluations

In this section, we describe the two follow-up pilot evaluations that we conducted and the subsequent refinements made to the preliminary EA-Coach interface. The first follow-up pilot involved three students and was intended to evaluate the incorporation of feedback for correctness to the EA-Coach. The addition of this feedback meant that students now had to use a more rigorous entry format than during the primary pilot. In particular, equations in the problem solution now had to include appropriate and consistent variable names. To support students in doing so, a 'variable definition' pane was added to the problem-solving window, following the Andes design (figure 3-5 shows a sample pane that contains two variables corresponding to a *block* and a *weight force*).

Variables			
Name	Definition	X-comp	Y-comp
block	mass of block		
W	magnitude of Weight Force on block due to Earth		

Figure 3-5: Variable definition pane

As we will describe in chapter 4, a variable-definition pane contains the variable names and corresponding definitions for steps entered into the problem solution. Apart from this one exception, the same interface was used in the first follow-up pilot as during the primary pilot. In particular, the example format corresponded to the SE-Coach format for presentation of examples. Since students did not indicate being uncomfortable with this format in the primary pilot evaluation, we intended to use it in the EA-Coach. However, the follow-up pilot revealed that students felt this format was not sufficiently similar to the problem-solving window's design. One common complaint was related to the lack of a variable definition pane in the example window. In the context of the example, the variable definition pane could help identify the meaning of the variables contained in the example solution⁶. Although this lack of similarity between the problem and example windows could be a form of scaffolding to discourage copying, we found that for several subjects it hindered problem solving. To address this issue, we changed the example format to more closely mirror the problem-solving format (we describe this format in chapter 4). We also made the following two refinements:

⁶ A possibility for why this was not an issue during the primary pilot is that students in that pilot were not aware of instances when their solutions contained errors corresponding to inappropriate variable names.

- We added one additional layer of scaffolding to the feedback delivered by the system, corresponding to having the EA-Coach inform students when their entry could not be interpreted due to *syntactic* errors (such as undefined variable names, missing algebraic operators, etc). We made this change after we observed one pilot subject struggling because she had incorrect variable names in her equation, but was not aware of the problem and so could not interpret the Coach's feedback. Since syntactic errors are not the result of domain misconceptions but rather of the input format imposed by the EA-Coach, we wanted to prevent students from wasting time trying to fix the errors.
- We modified the format for example solution steps corresponding to equations that represented vector components inclined with respect to an axis. In the preliminary pilot, component equations corresponding to vectors that were inclined with respect to the x or y axis were shown in the example using the same form (i.e., with the *cos* trigonometric function, $(F_{\{x,y\}} = F * \cos(\text{angle}))$). This format is used in the SE-Coach. However, the pilot revealed that some students found this representation confusing, because it was not what they were taught. Instead, students were accustomed to representing vector component equations inclined with respect to the y -axis using the *sin* function instead of the *cos* function (i.e., $F_y = F * \sin(\text{angle})$). Thus, we decided to use this representation in the example window.

The second follow-up pilot evaluation involved four subjects and was used to evaluate the refinements to the interface described above. This pilot did not indicate the need for any additional refinements.

3.3 Summary

In this chapter, we have described a series of pilot evaluations. The primary pilot evaluation was carried out to gather preliminary insight into the impact of the various types of superficial differences in our classification on APS. This pilot evaluation as well

as the two follow-up pilots were also used to refine the design of the EA-Coach interface, which we introduce in the next chapter along with the system architecture.

Chapter 4

Introduction to the EA-Coach

This chapter introduces the EA-Coach tutoring framework [Conati, Muldner et al., 2006]. The design of the framework is based on the cognitive science findings presented in chapter 2 and has evolved from the pilot evaluations described in chapter 3. In particular, cognitive science research shows that students experience difficulties during both example selection and application and that these difficulties are due to a lack of both expertise and meta-cognitive skills. Given these findings, the EA-Coach (1) takes over the responsibility of example selection, to ensure that students have access to appropriate examples, i.e., ones that discourage copying by stimulating the targeted meta-cognitive skills, namely min-analogy and EBLC, and (2) provides scaffolding to further encourage students to use the examples effectively. Before we provide details on the EA-Coach, however, we should point out that the tutor is designed to complement rather than replace traditional classroom instruction. Therefore, students are expected to have some domain knowledge obtained through regular curricular activities when using the system. Students

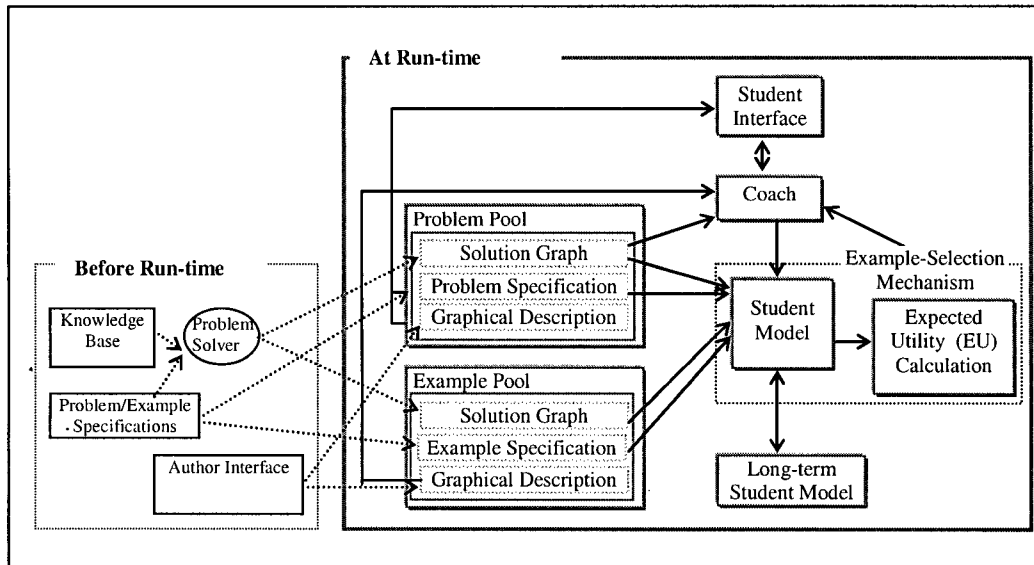


Figure 4-1: The EA-Coach architecture

can refine this knowledge by solving problems and referring to examples with the EA-Coach.

4.1 The EA-Coach Architecture

To automatically assess the impact of a given example on a student's APS behavior, including how she will generate the problem solution and what she will learn from it, the EA-Coach needs a formal representation of the problems and examples in the target domain. The EA-Coach also needs to encode a given student's domain knowledge and relevant meta-cognitive skills in a computational student model that can be used to guide the tailoring of instructional support. These requirements are implemented in the EA-Coach architecture, shown in figure 4-1. Note that two of the EA-Coach's architecture components come from Andes, an ITS we introduced in chapter 3 for problem solving without examples [Conati, Gertner et al. 2002]. These components, which we describe below, include (1) the knowledge base and (2) the problem-solving student interface.

<p>Rule: goal-try-Newton's 2nd Law</p> <p><i>If</i> the problem goal is to find a force <i>then</i> set the goal to try Newton's Second Law to solve the problem</p>
<p>Rule: goal-choose-body</p> <p><i>If</i> there is a goal to try Newton's Second Law to solve the problem <i>then</i> set the goal to select a body to which to apply the law</p>
<p>Rule: body-by-force</p> <p><i>If</i> there is a goal to select a body to apply Newton's Second Law <i>and</i> the problem goal is to find a force <i>then</i> select as the body the object to which the force is applied</p>
<p>Rule: normal-exists</p> <p><i>If</i> there is a goal to find all the forces on a body <i>and</i> the body rests on a surface <i>then</i> there is a normal force exerted on the body by the surface</p>

Figure 4-2: Sample rules in the knowledge base

4.1.1 Before Run-time

The *knowledge base* contains a rule-based representation of the physics domain (e.g., see figure 4-2). The rules were developed by the Andes project team [Conati, Gertner et al. 2002], in collaboration with three physics professors. The rules encode (1) physics knowledge (e.g., rules '*body-by-force*' and '*normal-exists*' in figure 4-2) and (2) planning knowledge that encodes higher-level 'abstract plans' an expert might use to focus the problem-solving process (e.g., rules '*goal-try-Newton's-2ndLaw*' and '*goal-choose-body*' in figure 4-2).

The EA-Coach *problem/example* pools are populated before run-time with the problem/example *graphical descriptions*, *specifications* and *solutions graphs*. The *graphical description* of each problem and example is generated by a human author through the *author interface*. The graphical description corresponds to what the student sees in the *student interface*, which we describe shortly (see figure 4-4). The author interface is based on the interface in Andes [Conati, Gertner et al. 2002], but is extended

to include functionality for creating the graphical description of the EA-Coach examples. The human author also provides, for each problem/example, the formal *specification* of the ‘problem statement’, which is encoded using the vector representation we introduced in chapter 2 (figure 4-3a shows the ‘problem statement’ and a fragment of the corresponding formal specification). The *problem solver*⁷ uses the specification and the rules in the knowledge base to automatically generate the system’s internal representation of the corresponding solution, in a structure called the *solution graph* [Conati, Gertner et al. 2002].

The solution graph is a dependency network representing how each solution step derives from previous steps and rules in the knowledge base. Figure 4-3b shows a fragment of the solution graph for the problem in figure 2-3. In the solution graph, solution steps are represented by facts and goals, collectively referred to as *proposition nodes* (nodes *F:* and *G:* nodes in figure 4-3b). The proposition nodes contain the system’s internal representation of the solution steps, which like the specification are encoded using the vector representation we introduced in chapter 2. The proposition nodes have as parents the rules (*R:* nodes) that derived them and previous propositions matching the rules’ enabling conditions. Thus, to solve the problem of finding the force on the block (node *G:find-force-on-block*), the highlighted segment of the network in figure 4-3b encodes the following (see nodes in bold in figure 4-3b). First, the rule *R:try-Newton’s-2ndLaw* establishes the goal to apply Newton’s 2nd Law (node *G:try-Newton’s-2ndLaw*). Next, the rule *R:goal-choose-body* sets the sub-goal to find a body to apply Newton’s 2nd Law (node *G:choose-body*). Finally, the *R:body-by-force* rule selects the block as the body (node *F:block-is-body*). The solution graph serves a number of key functions in the EA-Coach, as we will describe shortly.

⁷ The Problem Solver is an ‘off-the-shelf’ application: CLIPS, a forward-chaining expert system.

A workman pushes a 50 kg. block along the floor. He pushes it hard, with a magnitude of 120 N, applied at an angle of 25 degrees as shown. Find the normal force on the block.

(a)

```
<object (name, block)>
<mass (obj, block) (value, 50) (units, kg)>
<surface (name, floor) (incline, ?)>
...
<goal-problem (is, find-normal) (applied-to, block)>
```

(b)

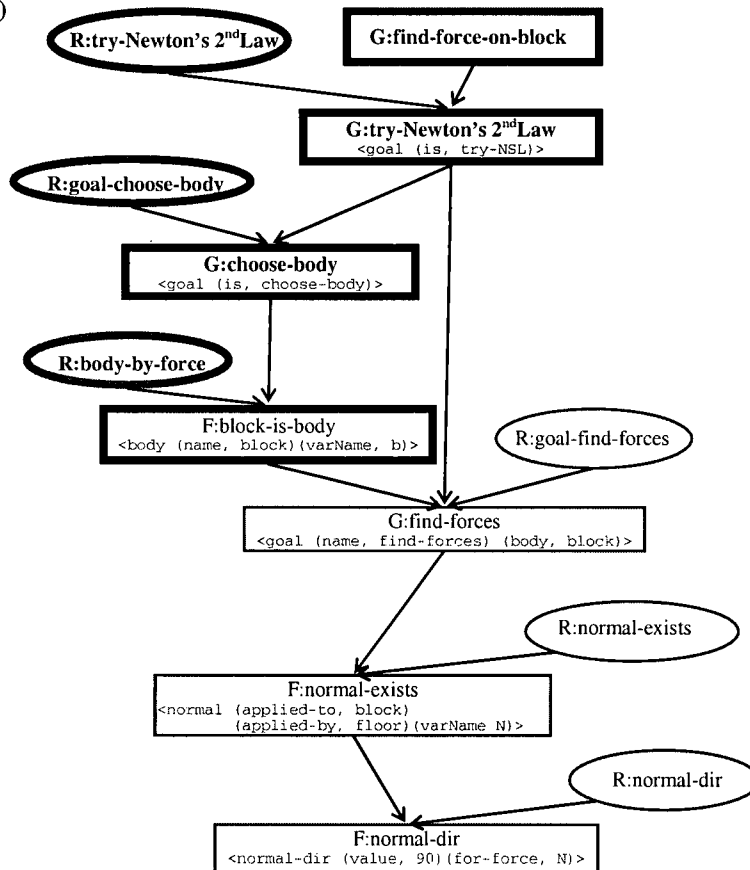


Figure 4-3: (a) Fragment of formal specification for problem in figure 2-3 (shown here, top); (b) corresponding solution graph fragment for the problem.

4.1.2 At Run-time

The *student interface* (interface from now on) allows students to solve problems from the problem pool and refer to examples from the example pool. During problem solving, the EA-Coach uses the solution graph to provide feedback for correctness on students' problem-solving entries, by matching entries to elements in the solution graph, based on the approach in [Conati, Gertner et al. 2002]. The examples that students refer to are dynamically selected by the *example-selection mechanism*, which corresponds to the *student model* and *expected utility (EU) calculation* components.

The student model is a dynamic Bayesian network that is automatically constructed when a student opens a new problem. The network is based on (1) the problem's solution graph and (2) information on the student's domain knowledge and meta-cognitive tendencies, stored in the *long-term student model*. The framework uses the student model to *predict* how each example from the example pool will help a student solve the target problem and what he will learn from it. To generate the simulation, the model integrates information on

- the similarity between the problem and the current example, assessed by comparing the solution graphs and specifications of the problem/example pair;
- the student's knowledge of the domain principles needed to derive the problem solution and his meta-cognitive tendencies.

The student model's prediction is passed to the *EU calculation* component, which quantifies the suitability of a candidate example by calculating its expected utility in terms of enabling successful problem solving and learning. Once the example-selection mechanism has calculated the expected utility for each candidate example, the *coach* retrieves the example with the highest expected utility and presents its graphical description in the interface. As the student proceeds with problem solving, the student model *assesses* how he actually uses the example to solve the problem and what he learns from it by integrating the aforementioned sources of information, as well as the student's interface actions (passed to it by the coach, as shown in figure 4-1). When the student closes the target problem, the long-term model is updated with the student model's

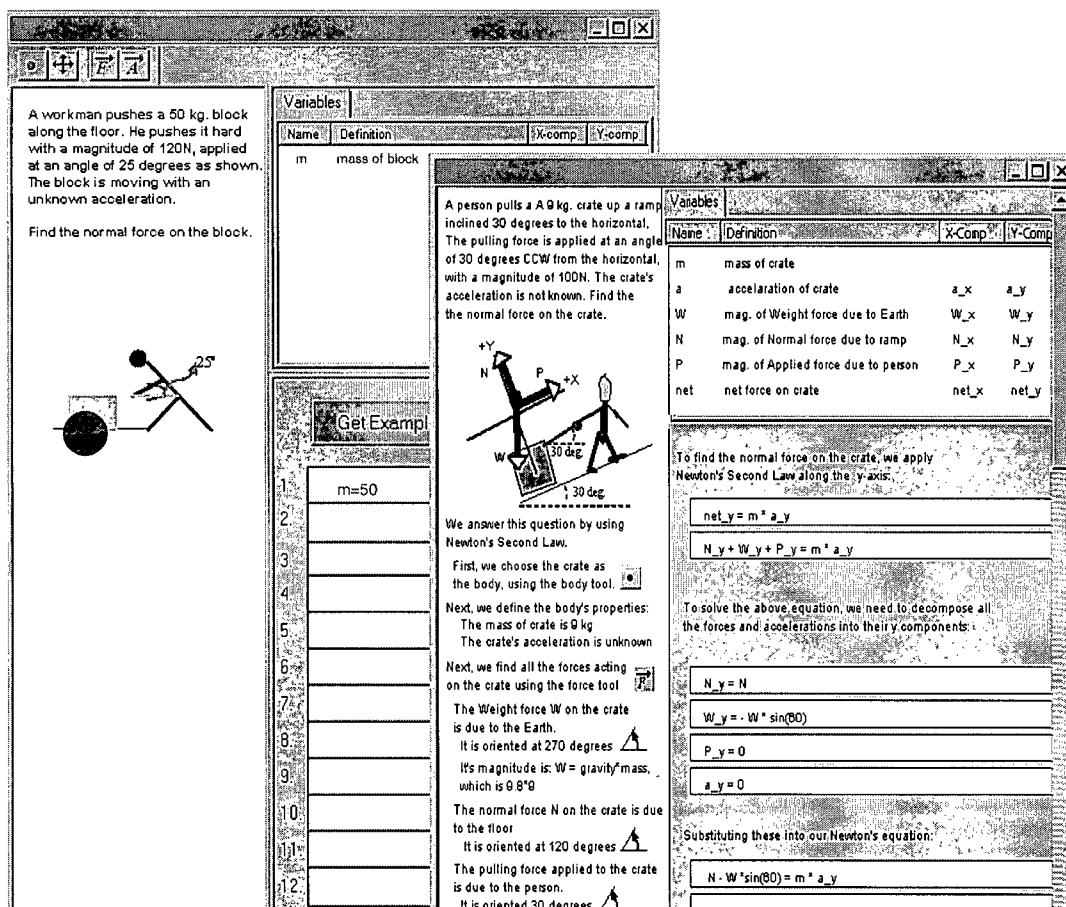
assessment of how the student's knowledge and meta-cognitive tendencies have evolved as a result of solving the problem. This allows the student model to have up-to-date information on the student each time he opens a problem to work on.

4.2 Interacting with the EA-Coach

We will now discuss student interaction with the EA-Coach. The EA-Coach offers several levels of scaffolding for the targeted meta-cognitive skills, i.e., min-analogy and EBLC. In this section, we describe the scaffolding that is embedded in the EA-Coach interface, while in chapter 5 we describe the details of the scaffolding provided through its adaptive example-selection mechanism.

The EA-Coach interface evolved from the pilot evaluations described in chapter 3 to the final version described here. The interface includes two windows that students use to solve problems and refer to examples (problem and example windows in figure 4-4). To work on a problem, students choose one through the '*Open Problem*' menu item found in the 'File' menu⁸. The problem window's design is directly based on that in Andes [Conati, Gertner et al. 2002] and includes three panels. The two panels used for entering the problem solution were described in section 3.1.1, namely the *Free-body* panel used to draw free-body diagrams with the provided tools (see left panel in figure 4-4a) and the *Equation* panel used to enter equations (see bottom-right panel in figure 4-4a). The system does not constrain input of the problem solution, in that students are free to enter the solution steps in any order and/or skip steps if they wish. The problem window also includes a *Variable* panel, which contains the list of variables used in the problem solution (e.g., '*m*' is the variable corresponding to the mass of the block in the top right panel of figure 4-4a). The EA-Coach automatically adds variables to the Variable panel

⁸ The problems are organized by the 'human author' into folders according to topic, and the responsibility is on the student to choose a problem, as is the case when students are doing exercises from a text book (i.e., currently, the EA-Coach does not provide support for helping a student choose a problem).



a) Problem Window

b) Example Window

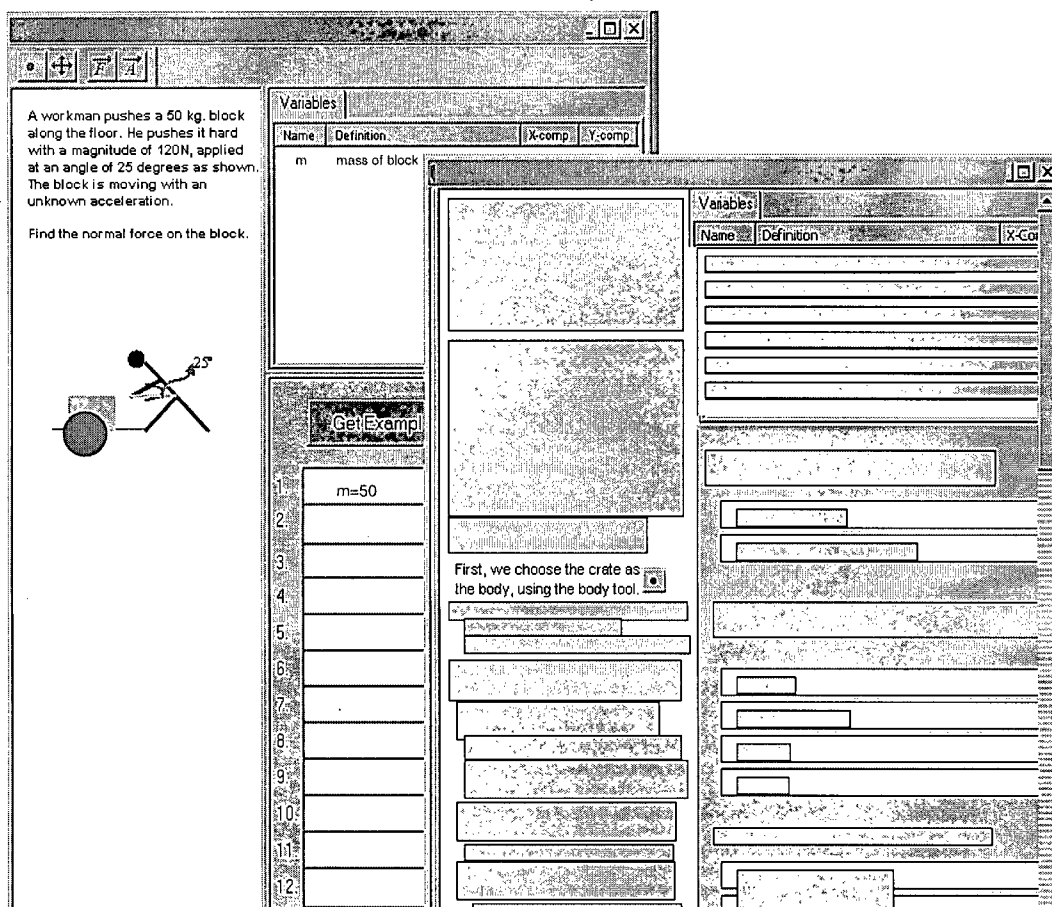
Figure 4-4: The EA-Coach interface (shown without the masking interface, which is displayed in figure 4-5)

when a student draws free-body elements; students can also add variables by clicking on the Variable pane. To allow the EA-Coach to assess the correctness of equations, any variables that students include in their equations must appear in the variable pane.

The EA-Coach provides immediate feedback for correctness on student problem-solving entries, by colouring correct vs. incorrect entries red or green, respectively. The EA-Coach also notifies students when it cannot interpret their entry because of improper

syntax, such as undefined variable names or missing arithmetic operators. This feedback is the first form of scaffolding for effective APS provided by the EA-Coach. As evidence in cognitive science demonstrates [Chi, Bassok et al., 1989] and as we confirmed through our pilot studies, some students lack self-monitoring skills and so are unable to diagnose their own misconceptions or errors. We argue that immediate problem-solving feedback can help trigger the right APS behaviours in these students. For instance, suppose a student with a tendency for max-analogy is generating the problem solution by indiscriminately copying from an example that includes some differences blocking 'correct' copying of its solution. Immediate feedback for correctness can make the student aware of the incorrectly copied steps and so discourage excessive copying by highlighting its limitations. Likewise, suppose a student inferred an incorrect rule via EBLC from an example and applied it to generate the problem solution (students may need several attempts before a correct rule is inferred [Chi 2000]). Feedback for correctness can make the student aware of the misconception and encourage her to repair it. As a final example, consider a min-analogy student who has poor self-monitoring skills and so does not realize she has errors in her solution. Feedback for correctness can help her realize the existence of the errors and seek help from an available example to fix them.

While working on a problem, students can use the '*Get Example*' button to ask for an example, which the EA-Coach adaptively selects and presents in the example window (figure 4-4b). The format of the example shown in the example window evolved from our pilot evaluations described in chapter 3. Based on pilot subjects' feedback, the example format is intended to mirror the problem-solving window's design. In particular, the example window includes a variable definition pane in order to help students identify the meaning of variables that appear in the example solution. The example window includes mechanisms to provide further scaffolding for the targeted meta-cognitive skills. One form of this scaffolding corresponds to the *masking interface* that covers the example specification and solution steps (see figure 4-5; note that the masking interface is not shown in figure 4-4). Moving the mouse over a region in the masking interface uncovers the region and covers whatever region was previously uncovered. The masking interface



a) Problem Window

b) Example Window

Figure 4-5: Masking interface

is intended to (1) discourage copying, because of the effort needed to explicitly uncover the example solution, and (2) encourage EBLC, by helping to focus students' attention on individual example solution steps.

To further discourage copying, another form of scaffolding corresponds to the lack of 'Copy' and 'Paste' functionality between the example and problem windows. This design is based on findings from an earlier study we conducted involving an interface that allowed cutting and pasting [Muldner 2002]. That study showed that some students

abused these functionalities to copy entire example solutions. A likely explanation for this finding is that the functionalities allow for the problem solution to be quickly generated, making it difficult for some students to resist using them.

It should be noted that although the EA-Coach provides several forms of scaffolding for the targeted meta-cognitive skills, they are all quite subtle. In particular, the framework does not provide any hints or prompts on meta-cognitive strategies or the target physics domain. Therefore, much of the responsibility to learn from APS is on the student. This design is intended to stimulate students to take initiative in the learning process, rather than enforcing a strict tutorial interaction in which students passively follow a tutor's directive.

4.3 Summary

In this chapter, we introduced the EA-Coach. We began by describing the framework's architecture and then illustrated how students interact with the EA-Coach.

In the next chapter, we provide details on the key computational mechanisms that enable the EA-Coach to provide tailored support for APS through its example-selection mechanism, including the student model and the expected utility calculation components.

Chapter 5

Tailored Support for APS through Example Selection

In this chapter, we describe the tailored support the EA-Coach provides for APS by presenting students with adaptively selected examples. We begin by stating the EA-Coach's example-selection objectives. We then introduce the EA-Coach student model, which plays a key role in helping the system meet its selection objectives. After presenting the general approach the EA-Coach takes for example selection, we describe the example-selection process and corresponding computational mechanisms in detail [Muldner and Conati, 2007]. Finally, we describe how the selected example is used to refine the student model's assessment of the student [Muldner and Conati, 2005].

5.1 EA-Coach's Example-Selection Objectives

When a student asks for an example, the EA-Coach selects the one from its example pool that best meets the following two objectives:

1. *Learning* objective: the example triggers strengthening of existing and learning of new domain rules by encouraging min-analogy and EBLC
2. *Problem-solving success* objective: the example helps generate the target problem solution

A challenge with our approach is balancing learning with problem-solving success for *different* learners. Examples that are not highly similar to the target problem may support the learning objective by discouraging shallow APS behaviors such as pure copying. However, they may also hinder students from producing a problem solution, because they do not provide enough scaffolding for students who lack the necessary domain knowledge or meta-cognitive skills (also referred to as meta-cognitive *tendencies* here). Thus, it is key that the EA-Coach consider both similarity *and* student characteristics during the example-selection process. In particular, the system needs to take into account (1) the impact of a given student's knowledge and meta-cognitive tendencies on how he will use an example to solve the target problem, so that the EA-Coach can select examples that have the best potential to enable problem solving and learning for that student and (2) how the student's knowledge and meta-cognitive tendencies actually develop from using the example to solve the problem, so that the system will have up-to-date information on the student the next time it selects an example for him. To meet these requirements, the EA-Coach relies on the ITS component that is responsible for representing and reasoning about the student, i.e., the student model.

5.2 Introduction to the EA-Coach Student Model

A traditional function of a student model is to generate an *assessment* of the student based on information coming from her interaction with the ITS. For instance, a student

model could use a student's problem-solving entries to infer that she knows the corresponding domain rules. However, a student model can also be used to simulate the impact that a tutorial action will have on the states of interest. For instance, a student model could simulate how a particular hint will impact a student's knowledge. To meet the example-selection requirements listed in the previous section, the EA-Coach operates in both these modes, as follows:

[*Assessment Mode*] The model generates an *assessment* of how a student's knowledge and meta-cognitive tendencies for min-analogy and EBLC evolve as she engages in APS with the EA-Coach. This assessment allows the example-selection mechanism to have up-to-date information on the student.

[*Simulation Mode*] The model generates a *prediction* of how a student will solve a problem in the presence of a particular example and what she will learn from doing so. This prediction forms the basis for the example-selection mechanism's ability to tailor its operation to a student's needs.

To infer a student's learning and problem-solving outcomes, the model aims to infer the APS behaviors (copying, EBLC) influencing the outcomes. All of these inferences are challenging to generate, due to the quality and type of information available to the model. Recall that, as we indicated in chapter 4, in both modes the model takes into account information on both problem/example similarity and student characteristics (knowledge and meta-cognitive tendencies). In assessment mode, the model also uses information coming from a student's interface actions, including the problem steps the student entered in the interface and the example steps the student viewed in the masking interface. However, in neither mode does the model have *direct* information on students' APS behaviors corresponding to copying and EBLC or meta-cognitive tendencies and subsequent learning outcomes. During assessment mode, the lack of direct information stems from the fact that the EA-Coach interface does not require students to explicitly communicate to the student model how they are reasoning. In particular, the interface is not designed to capture copying/self-explanation through EBLC, for instance via specially designed tools that students would be required to use in order to copy example

solutions or generate EBLC self-explanations. Our design allows for a natural interaction with the EA-Coach, but it means that the information available to the model is much more ambiguous than if the system constrained students' interaction. As far as learning is concerned, the only information available to the model that a student has learned a rule is when she generates the corresponding solution step. However, generating a step does not guarantee that a student has learned the rule needed to do so, because a student could, for instance, generate the step by guessing or copying. Likewise, lack of direct information on copying and EBLC reasoning complicates the assessment of APS meta-cognitive tendencies (min-analogy and EBLC). During simulation mode, the model has even less information than during assessment. That is, the model aims to infer how a student will solve the problem and learn from it solely from information on problem/example similarity and the student's cognitive and meta-cognitive skills.

The fact that the model only has access to indirect information on the target student states during the assessment/simulation modes introduces a good deal of uncertainty to the student-modeling process. Uncertainty also stems from the fact that the relationships between the sources of information and the states the model aims to infer are uncertain, including:

- How similarity impacts APS for different learners. As we pointed in chapter 2, it is not clear how various levels of similarity influence APS behaviors and subsequent learning and problem-solving success for students with different levels of expertise and meta-cognitive skills.
- How EBLC interacts with learning. Although EBLC reasoning increases the likelihood that a student learns a domain principle it does not guarantee it, since students may sometimes require several attempts before correct domain principles are inferred [Chi, 2000].

5.2.1 Handling Uncertainty through Dynamic Bayesian Networks

To handle the uncertainty in the modeling process, the EA-Coach student model relies on dynamic Bayesian networks, an extension of the formal framework for representing and

reasoning under uncertainty provided by Bayesian networks [Pearl, 1988]. A Bayesian network is a directed acyclic graph in which the nodes represent random (uncertain) variables, and the arcs represent the direct probabilistic dependencies between them. To specify a Bayesian network, the following must be defined: (1) the graphical structure of the network; (2) a probability distribution for each node in the network, corresponding to specifying the prior probability of the values of each root node, and the conditional probability of the values of each non-root node given its parents.

A key advantage of a Bayesian network is that it can be used to *efficiently* answer probabilistic queries about the variables encoded in the network. This is because Bayesian network algorithms exploit the independencies between the variables to reduce the computational complexity of probabilistic inference.

Figure 5-1 shows an example of a Bayesian network containing the following binary random variables:

- *Rule1*: the probability the student knows the formula needed to compute the weight of an object ($Weight = m * g$, where m is the mass of the object and g is the gravitational acceleration)
- *Rule2*: the probability the student knows the formula for Newton's Second Law ($\sum F = m * a$, where $\sum F$ is the net force on an object, m is the object's mass and a is the object's acceleration)
- *Fact*: the probability the student derives $W = m * g$ in her head
- *Action*: the student writes the equation $W = m * 9.8$

This network encodes the following relationships between the variables (see figure 5-1): (1) it is possible to derive *Fact* by knowing either *Rule1* or *Rule2* (see [1] figure 5-1, top right), although even if both rules are not known, there is a small probability that *Fact* can be derived by some other means, such as guessing and (2) if the student derives an equation mentally (in his head) then there is a high probability that he will write the equation, but there is also a small probability that the equation will not be written (see [2] figure 5-1, bottom right).

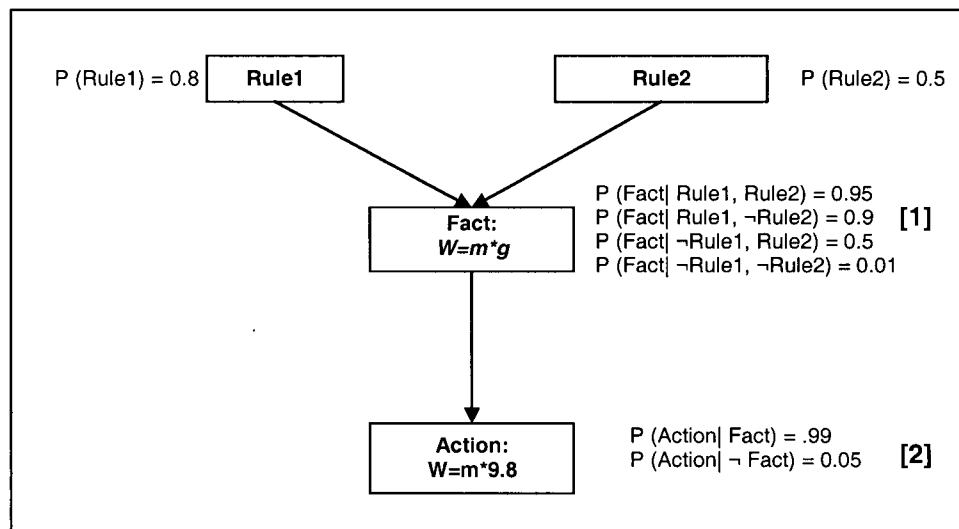


Figure 5-1: Example of a Bayesian network

Given the network depicted in figure 5-1, we can perform various types of inference, including the following:

- Observe that the student wrote ' $W=m*9.8$ ' and ask for the probability that he knows either of the rules (*diagnostic inference*)
- Observe that the student knows *Rule1* and ask for the probability that he will write the correct equation $W=m*9.8$ (*predictive inference*)
- Observe that the student wrote the ' $W=m*9.8$ ' equation and that he knows *Rule1* and ask for the probability that he knows *Rule2* (*inter-causal inference*)

The Bayesian network in figure 5-1 is static: each node represents a variable whose value is *fixed*, i.e., does not change over time (although the model's belief in the variable's value may change). However, the EA-Coach student model aims to represent how a student's knowledge and meta-cognitive tendencies *evolve* throughout the course of her interaction with the tutor. To do so, it uses an extension of a Bayesian network that

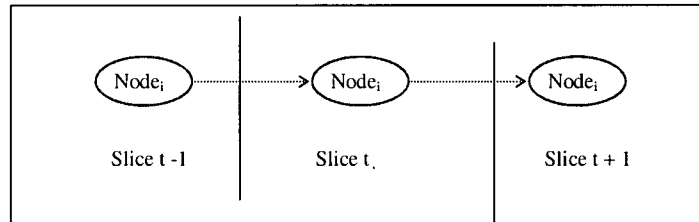


Figure 5-2: Simple dynamic Bayesian network

allows for modeling the evolution of random variables over time, namely a *dynamic Bayesian network (DBN)* [Dean and Kanazawa, 1989].

In a DBN, the temporal evolution of a variable's true value is modeled by a sequence of nodes corresponding to the variable (e.g., node_i in slice $t-1$, slice t and slice $t+1$ in the very simple network shown in figure 5-2). A 'slice' represents the state of the random variables in the model at a given time. Typically, a slice is created at each point one wants to model the variables' temporal evolution (e.g., slice t in figure 5-2 represents a probability distribution over the variable encoded by node_i at time t). This temporal evolution is encoded via temporal links between slices, which capture that a variable's value depends on its earlier value (temporal links are shown as 'dotted' lines in figure 5-2).

5.2.2 Student Model Construction through the Andes Approach

A key challenge associated with using Bayesian networks for the modeling task corresponds to building the network, including specifying its structure and parameters. The EA-Coach constructs its network by relying on the approach taken by Andes, an ITS we introduced in chapter 3 that supports pure problem solving [Conati, Gertner et al., 2002]. However, the EA-Coach student model operates in a different instructional situation (APS) than the Andes model (pure problem solving). Thus, it distinguishes itself from the Andes model in several key ways, which we illustrate once we describe how Andes creates its student model Bayesian network.

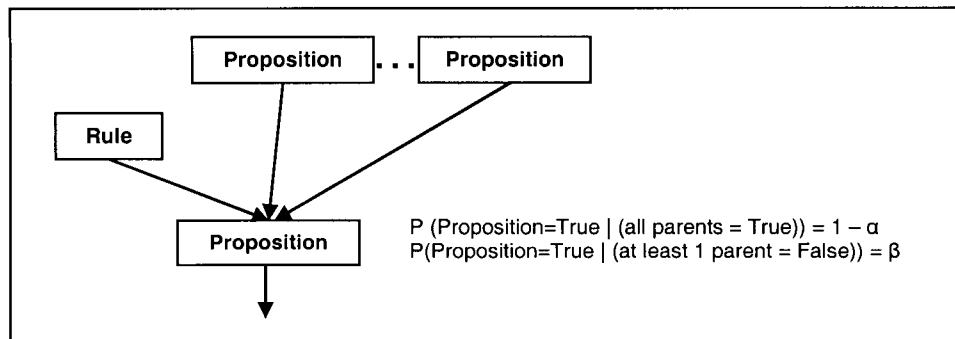


Figure 5-3: Andes Bayesian network student model

When a student opens a new problem to work on, Andes automatically creates the student model Bayesian network from the target problem's *solution graph*. Recall that the solution graph encodes how the problem's solution can be derived from domain rules and intermediate solution steps. The system converts the solution graph into a Bayesian network by supplementing the solution graph nodes (*Rule* & *Proposition* nodes, see figure 5-3) with probability distributions over the nodes' values.

Rule Nodes. Rule nodes have binary values *True* and *False*: $P(\text{Rule}=\text{True})$ represents the probability P that the student knows the rule. In the Andes network, rule nodes are root nodes. The priors come from information on a specific student's knowledge (e.g., from a pre-test) – if this information is not available, the priors are set using generic values (e.g., 0.5).

Proposition Nodes. Proposition nodes have binary values *True* and *False*: $P(\text{Proposition}=\text{True})$ represents the probability P the student can generate the proposition. Propositions represent the following two kinds of elements in the problem solution:

- Equations and free-body entries that a student can generate in the interface (i.e., fact nodes from the solution graph, 'F:' in figure 4-3)
- High-level planning steps that cannot be explicitly entered in the interface (i.e., goal nodes from the solution graph, 'G:' in figure 4-3)

The conditional probabilities for the proposition nodes are defined by using canonical interactions (e.g., Leaky/Noisy-AND [Henrion, 1989]). Doing so adequately approximates the dependencies in the network, while greatly simplifying the specification of the conditional probabilities. Specifically, the conditional probabilities between a proposition node and its parents are modeled by a Leaky/Noisy-And relationship, shown in figure 5-3, which encodes the following assumptions:

- The ‘noise’ parameter α models that there is a non-zero probability a proposition is false even if all parents are true. This encodes the possibility that even if the student *does* have the necessary prerequisites to generate the corresponding problem step, she may not generate the step.
- The ‘leak’ β parameter models that there is a non-zero probability a proposition is true even if some of its parents are false. This encodes the possibility that even if the student *does not* have the necessary prerequisites to generate the step, she may still generate the corresponding problem step by using alternate means, such as guessing or asking a friend.

As a student solves a problem, Andes uses the Bayesian network to perform knowledge assessment and plan recognition of the students’ actions. To do so, a student’s problem-solving entries are used to generate a probabilistic estimate of the student’s knowledge. This enables the Bayesian network to infer which part of the solution the student is working on and which part is causing the student difficulty.

The EA-Coach follows the Andes approach to construct its initial student model when a student opens a problem, by (1) converting the problem’s solution graph into a Bayesian network and (2) specifying the conditional probabilities between the solution graph nodes using canonical interactions. However, while the Andes Bayesian network is static, the EA-Coach uses a dynamic Bayesian network, needed to model the evolution of students’ knowledge from EBLC explanations. The EA-Coach model also has two other key differences from the Andes model. First, as we already pointed out above, the EA-Coach model operates in a different instructional situation than the Andes model, i.e., APS vs. pure problem solving. Thus, the EA-Coach model aims to infer how the presence of an

example influences students' problem solving and subsequent learning. To do so, the EA-Coach dynamic Bayesian network includes additional nodes that represent occurrence of (1) EBLC/copying, (2) problem/example similarity and (3) the students' APS meta-cognitive tendencies. The second difference between the EA-Coach model and the Andes model is related to the type of inference each model performs. Both models are used to generate an *assessment* of the student while she is solving a problem. However, the EA-Coach model is also used to generate a *simulation* (prediction) of how the student will solve the problem given a particular example. As will be described shortly, this simulation is a critical component of the EA-Coach's example-selection mechanism.

5.3 A Decision-theoretic Approach for Example Selection

Before providing details on the example-selection mechanism, we will introduce the general approach we adopted for example selection. As we pointed out above, the student model plays a key role during example selection by generating a simulation of a student's problem solving and learning. However, the fact that the model's information is permeated with uncertainty complicates the decision regarding which example to select, since it means that the EA-Coach needs to consider not only which outcome is desirable but also how likely it is to happen. To handle this uncertainty in a principled manner during the decision-making process, the EA-Coach relies on a decision-theoretic framework [Clement, 1996; Russell and Norvig, 1995]. In a decision-theoretic framework, an agent's decision depends on the following two factors:

- What the agent believes, represented using probability theory.
- What the agent wants, i.e., its preferences between different outcomes, represented using utility theory. Specifically, preferences are represented by a utility function that assigns a value to each outcome expressing its desirability.

A decision-theoretic framework provides the specification of what it means to act rationally when choosing between actions with uncertain outcomes: an agent is rational if and only if it chooses the action that yields the highest *expected* utility, averaged over all

possible outcomes of that action. Therefore, a decision-theoretic agent evaluates an action by weighing the utility of each possible outcome by the probability that it occurs.

In the context of the EA-Coach, the decision under consideration corresponds to which example to present to the student. Finding an example with the highest expected utility involves a two-phase process for each candidate example:

[*Simulation Phase*] First, the EA-Coach uses its student model to simulate how a student will solve a given problem in the presence of the candidate example and what the student will learn from doing so. Thus, the outcome of this simulation is a probabilistic prediction of how a candidate example impacts problem-solving success and learning.

[*Expected Utility Calculation Phase*] Given this prediction, the EA-Coach uses a utility function to quantify the candidate example's expected utility in terms of meeting the learning and problem-solving success objectives.

The simulation and utility calculation phases are repeated for each example in the EA-Coach's example pool, and the example with the highest expected utility for learning and problem-solving success is presented to the student. We now describe the simulation and utility calculation phases.

5.4 Simulation Phase

During the simulation phase, the EA-Coach uses its student model to generate a prediction of learning and problem-solving success for a candidate example. Doing so involves first adding a simulation slice to the dynamic Bayesian network, and then incorporating nodes into the slice to model a student's APS behaviors. We will illustrate these steps by relying on a concrete scenario. Suppose a student opens the problem in figure 2-3 and asks for an example. Opening the problem results in the construction of the dynamic Bayesian network from the target problem's solution graph (a fragment of the network is shown in slice t , figure 5-4, left). Initially, this network contains all of the

solution graph rule and proposition nodes ('*R*', '*G*', '*F*' nodes in slice *t*, figure 5-4), as well as two binary-valued nodes to model meta-cognitive tendencies, as follows:

- *MinAnalogyTend* node, where $P(\text{MinAnalogyTend} = \text{True})$ represents the probability a student has a tendency for min-analogy (see '*MinAnalogyTend*' node in slice *t*, figure 5-4)
- *EBLCTend* node, where $P(\text{EBLCTend} = \text{True})$ represents the probability a student has a tendency for EBLC (see '*EBLCTend*' node in slice *t*, figure 5-4)

The prior probabilities for the rule and tendency nodes come from the long-term model (see EA-Coach architecture, figure 4-1). The conditional probabilities for the proposition nodes are specified using the canonical interactions described in section 5.2.2. We now illustrate the simulation phase with this network, assuming the candidate example is that shown in figure 2-3.

5.4.1 Simulation Slice

As the first step in the simulation phase, a special '*simulation*' slice is added to the network (figure 5-4, slice *t+1*, simulation slice). This slice will be used to generate a prediction of how a student will solve the problem in the presence of the candidate example, and how the student's knowledge will evolve from doing so. Initially, the simulation slice is directly based on the slice directly before it (pre-simulation slice from now on), in that it contains all the same rule and proposition nodes (figure 5-4, pre-simulation and simulation slices, *R* and *F/G* nodes). The simulation slice does not, however, include the two APS tendency nodes ('*EBLCTend*' and '*MinAnalogyTend*' nodes in slice *t*, figure 5-4). This is because the simulation does not predict the impact of an example on a student's APS tendencies. Although doing so is an interesting possibility, it adds additional complexity to the simulation, and so we decided to leave this possibility for future work, after we have validated the approach on the simplified model.

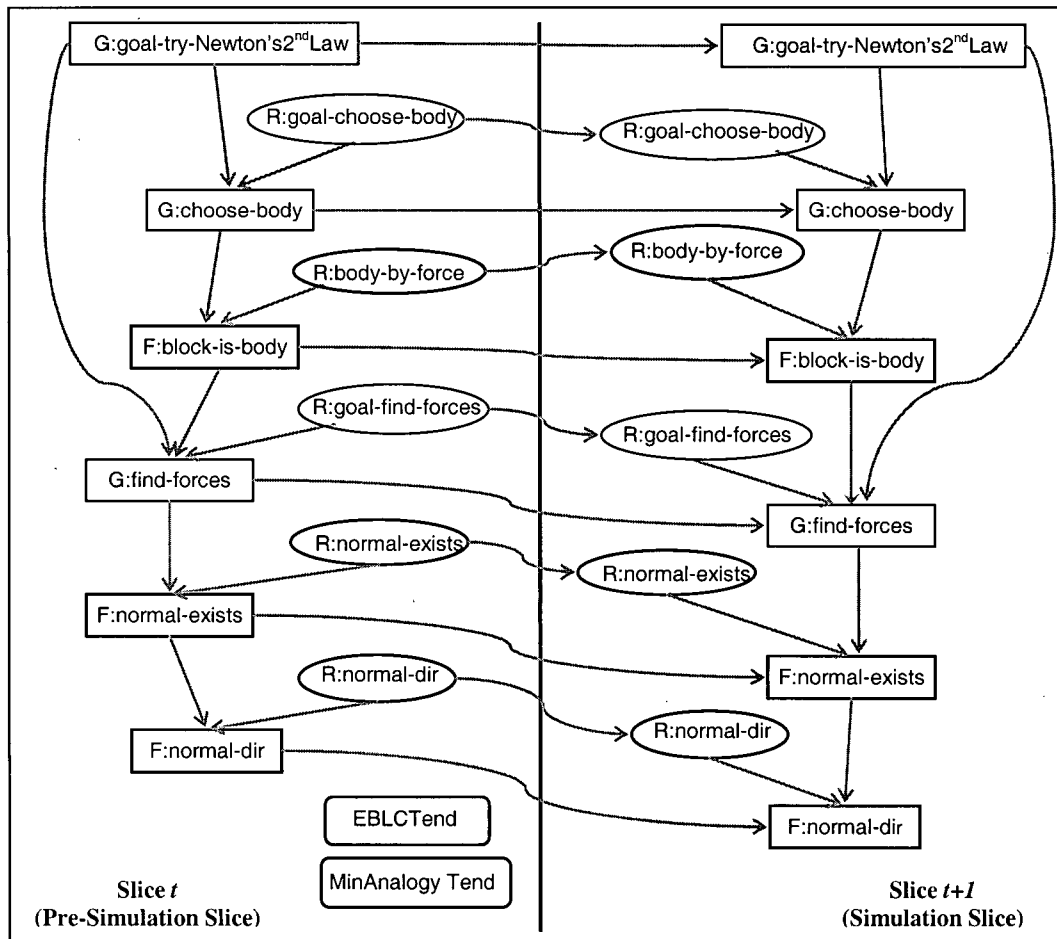


Figure 5-4: Fragment of the EA-Coach dynamic Bayesian network

5.4.2 Nodes to Model APS Behaviors

As a second step in the simulation phase, to model students' APS behaviors, additional nodes are incorporated into the simulation slice (figure 5-5 shows a fragment of the network from figure 5-4 with the additional nodes in bold), as follows:

- *Similarity* nodes encode the similarity between the problem solution step and the corresponding example step, if any (e.g., figure 5-5, simulation slice, '*Similarity_{n-exists}*' and '*Similarity_{n-dir}*' nodes).

- *Copy* nodes encode the probability that the student will generate a step in the problem solution by copying from the example (e.g., figure 5-5, simulation slice, '*Copy_{n-exists}*' and '*Copy_{n-dir}*' nodes).
- *EBLC* nodes encode the probability that the student will infer a given rule via EBLC by explaining an example⁹ step (e.g., figure 5-5, simulation slice, '*EBLC_{n-exists}*' and '*EBLC_{n-dir}*' nodes).

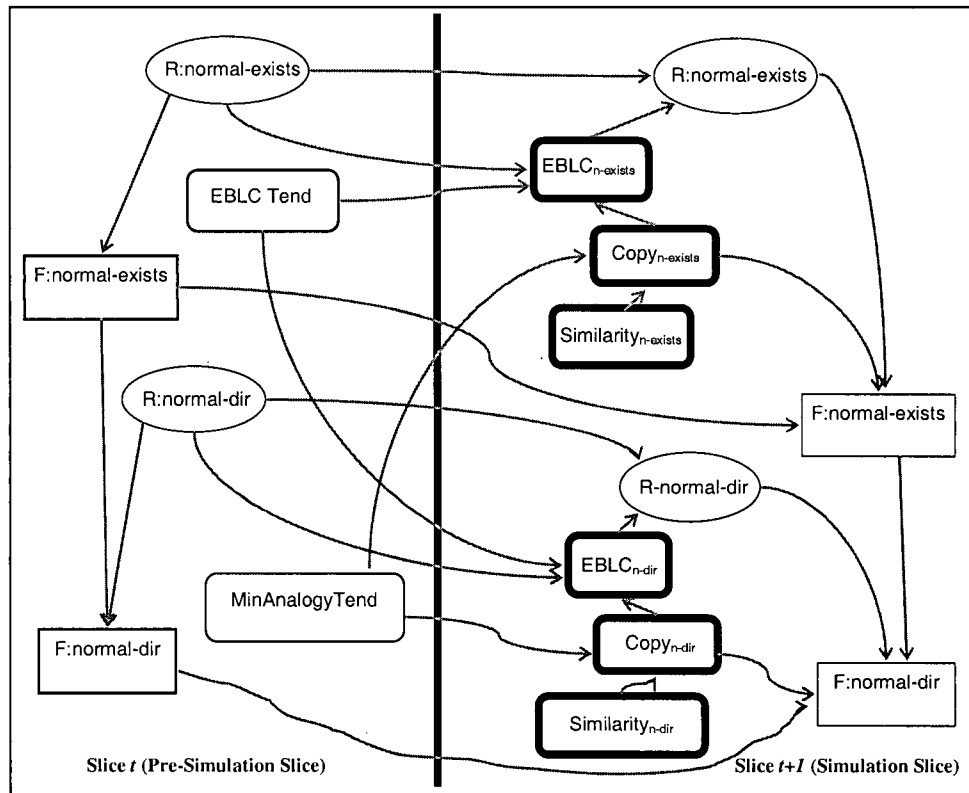


Figure 5-5: Additional nodes in simulation slice to predict APS behaviors

⁹ Although in [VanLehn, Ball et al., 1990], the authors suggest that students *could* also learn via EBLC during pure problem solving (i.e., without an example), this is not currently modeled because very little detail is provided on how it occurs.

The general procedure is that a *Copy* node and a *Similarity* node is added for each *fact* node corresponding to a solution step in the network ('*F:*' nodes in the dynamic Bayesian network), while an *EBLC* node is added for each *rule* node ('*R:*' nodes in the dynamic Bayesian network). Goal nodes ('*G:*' nodes in the dynamic Bayesian network) and corresponding rules represent a special case in the simulation, which will be described in section 5.4.4.

5.4.2.1 Similarity Nodes

During the simulation phase, the only form of direct evidence for the student model corresponds to the similarity between the problem and candidate example, encoded by the *Similarity* nodes (e.g., '*Similarity_{n-exists}*' and '*Similarity_{n-dir}*' nodes in the simulation slice from the network in figure 5-5, shown in figure 5-6 with some additional information on node values). A similarity node is added for each fact node in the simulation slice (referred to as *step* in subsequent discussion).

The similarity node's value is based on the definitions presented in section 2.2.1.1 and is either *None*, *Trivial* or *NonTrivial*. To set a similarity node's value, the EA-Coach compares the solution graphs of the problem and the candidate example, as well as their specifications stored in the EA-Coach's knowledge base, using the following algorithm:

- [Step 1] locate the problem step node *sp* that the similarity node is linked to the problem's solution graph,
- [Step 2] Identify the rule *R* that generated *sp*,
- [Step 3] Check if rule *R* is in the example's solution graph:
 - [Case 3a: *Structurally different; Value = None*] If *R* is not in the example's solution graph, then a structural difference exists between the problem and example with respect to *sp*. Set the value of the similarity node to *None*.
 - [Case 3b: *Structurally identical; Value = Trivial or NonTrivial*] If *R* is in the example's solution graph, then the problem/example are structurally identical with respect to *sp*. Identify the type of superficial relation (trivial, non-trivial) between the problem and example with respect to *sp*:
- [Step 4] Identify the set of nodes $S = \{ se_1, \dots, se_n \}$ in the example's solution graph that were derived by *R*.

[Step 5] Identify which node se in S to compare with sp :

[Case 5a] If $|S|=1$, this is straightforward, i.e., $se = se_1$.

[Case 5b] If $|S|>1$, then use heuristics to identify se (see Appendix 1).

[Step 6] Assess the superficial similarity between sp and se and set the similarity node value accordingly. Since sp and se encode the solution step using the formal vector representation we described in chapter 2, determining the superficial similarity involves comparing the corresponding slots in sp/se 's vectors, and setting the similarity node according to our definitions presented in section 2.2.1.1. In particular, the value of the similarity node is set to *NonTrivial* if sp and se include at least one non-trivial difference and to *Trivial* otherwise. Given a superficial difference between sp and se , to determine if it is *trivial* or *non-trivial*, the system checks if (1) the example constant corresponding to the difference appears in the example specification, and (2) has a corresponding constant in the problem specification; if (1) and (2) are true, then the difference is assessed as *trivial*, and otherwise, as *non-trivial*.

In figure 5-6, the value of the *Similarity_{n-exists}* node is *Trivial* because the problem step it is linked to, which encodes that a normal force exists, only includes trivial differences with the corresponding example solution step (i.e., the constants corresponding to the differences appear in both the problem and example specifications; see section 2.2.1.1 for full details). On the other hand, the value of the *Similarity_{n-dir}* node is *NonTrivial* because the problem step it is linked to, which encodes the normal force direction, includes a non-trivial difference with the corresponding example solution step (i.e., the constant corresponding to the difference is missing from the example specification; see section 2.2.1.1 for full details).

Similarity nodes play a fundamental role in predicting APS behaviors according to our hypotheses presented in chapter 2, as we will see shortly.

5.4.2.2 Copy Nodes

Copy nodes allow the model to predict whether the student will generate the corresponding problem solution step by copying from the example (e.g., '*Copy_{n-exists}*' and '*Copy_{n-dir}*' nodes in the simulation slice from the network in figure 5-5 shown here in figure 5-6). A copy node is added for each problem solution step in the simulation slice.

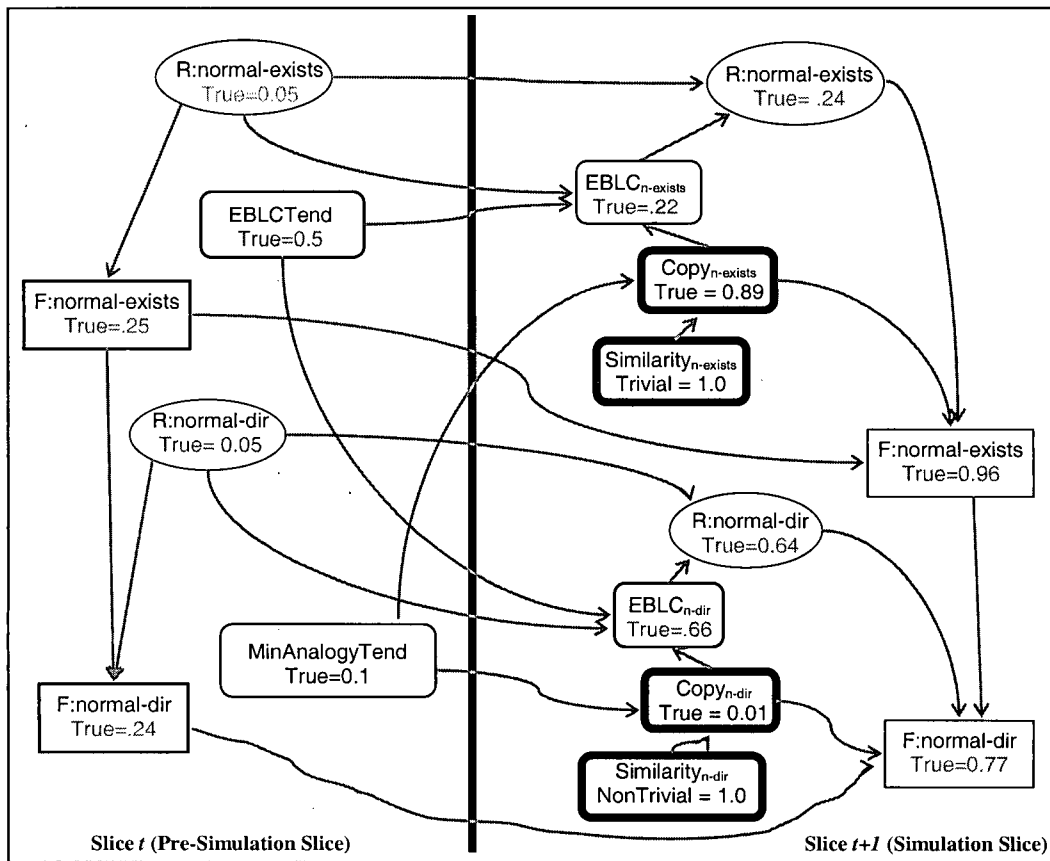


Figure 5-6: Copy nodes in simulation slice

Table 5-1: Copy node CPT

MinAnalogyTend	True			False		
	NonTrivial	None	Trivial	NonTrivial	None	Trivial
Copy=False	0.99	0.99	0.8	0.99	0.99	0.05
Copy=True	0.01	0.01	0.2	0.01	0.01	0.95

Copy nodes have binary values *True* and *False*, where $P(\text{Copy}=\text{True})$ is the probability P the student will generate the corresponding problem step by copying. The copy node's *Conditional Probability Table* (CPT, see table 5-1) expresses how its value depends on the following factors:

- The similarity between the problem step the copy node refers to and the corresponding step in the example solution, captured by the parent similarity node (e.g., figure 5-6, '*Similarity_{n-dir}*' node linked to the '*Copy_{n-dir}*' node in the simulation slice)
- The student's tendency for min-analogy, captured by the parent min-analogy tendency node (e.g., figure 5-6, '*MinAnalogyTend*' node linked to the '*Copy_{n-dir}*' node in the simulation slice)

The probabilities in the CPT encode that if the value of the parent similarity node is *NonTrivial* or *None*, then the probability of the corresponding problem solution step being generated by copying is virtually zero, because the example does not afford students the opportunity to do so. If the value of the parent similarity node is *Trivial*, then the example allows the student to generate the problem solution step by copying. In this case, the probability of copying depends on the student's tendency for min-analogy. If the student does not have a tendency for min-analogy, indicating that she prefers to generate the problem solution by copying, the probability of copying is high. On the other hand, if the student does have a tendency for min-analogy, then the probability of copying is low.

The impact of these factors is shown in figure 5-6. The probability the student will generate the problem step corresponding to node '*F:normal-exists*' by copying is high (simulation slice, '*Copy_{n-exists}*', *True*=.89). This is because the problem/example similarity allows for it (simulation slice, '*Similarity_{n-exists}*'=*Trivial*,) and the student has a low tendency for min-analogy (pre-simulation slice, '*MinAnalogyTend*', *True*=.1). In contrast, the probability the student will generate the problem step corresponding to node '*F:normal-dir*' by copying is very low (simulation slice, '*Copy_{n-dir}*', *True*=.01). This is because the non-trivial difference (simulation slice, '*Similarity_{n-dir}*'=*NonTrivial*) between the problem step and corresponding example step blocks copying.

We should point out that the values in the copy node's CPT, as well as the CPTs described in subsequent sections, represent our best first estimate of the impact of factors such as student characteristics/similarity on the variable under consideration, and may need to be refined in the future, for instance via sensitivity analysis. Future work could involve using machine learning, i.e., learning the CPT parameters from data, for instance via the Expectation Maximization (EM) algorithm [Dempster, Laird et al., 1977]. A key challenge with taking this approach for setting the CPT values is obtaining the data needed for the learning task.

5.4.2.3 EBLC Nodes

EBLC nodes allow the simulation to predict whether the student will learn the corresponding rule through EBLC from the example (e.g., '*EBLC_{n-exists}*' and '*EBLC_{n-dir}*' nodes in the simulation slice from the network in figure 5-5 shown here in figure 5-7). During the simulation phase, the framework adds one EBLC node for each rule node in the simulation slice that (1) can be derived via this type of reasoning, based on [VanLehn, Jones et al., 1992] that describes how *domain* rules but not *planning* rules can be learned through EBLC (the difference between EBLC on planning vs. domain rules will be discussed in section 5.4.4), and (2) is included in the example solution (i.e., the example includes a step that is structurally identical to the problem step derived by the rule under consideration).

EBLC nodes have binary values *True* and *False*, where $P(EBLC=True)$ is the probability P the student will reason via EBLC on the corresponding example solution step in an attempt to learn the rule deriving the step. The EBLC node's CPT (see table 5-2) expresses how its value depends on the following factors:

- The probability the student knows the rule, captured by corresponding rule node in the pre-simulation slice (e.g., figure 5-7, pre-simulation slice '*R:normal-dir*' node, linked to the '*EBLC_{n-dir}*' node in the simulation slice)
- The probability the student will generate the problem solution step derived by this rule by copying, captured by the corresponding copy node (e.g., figure 5-7, simulation slice, '*Copy_{n-dir}*' node, linked to the '*EBLC_{n-dir}*' node)

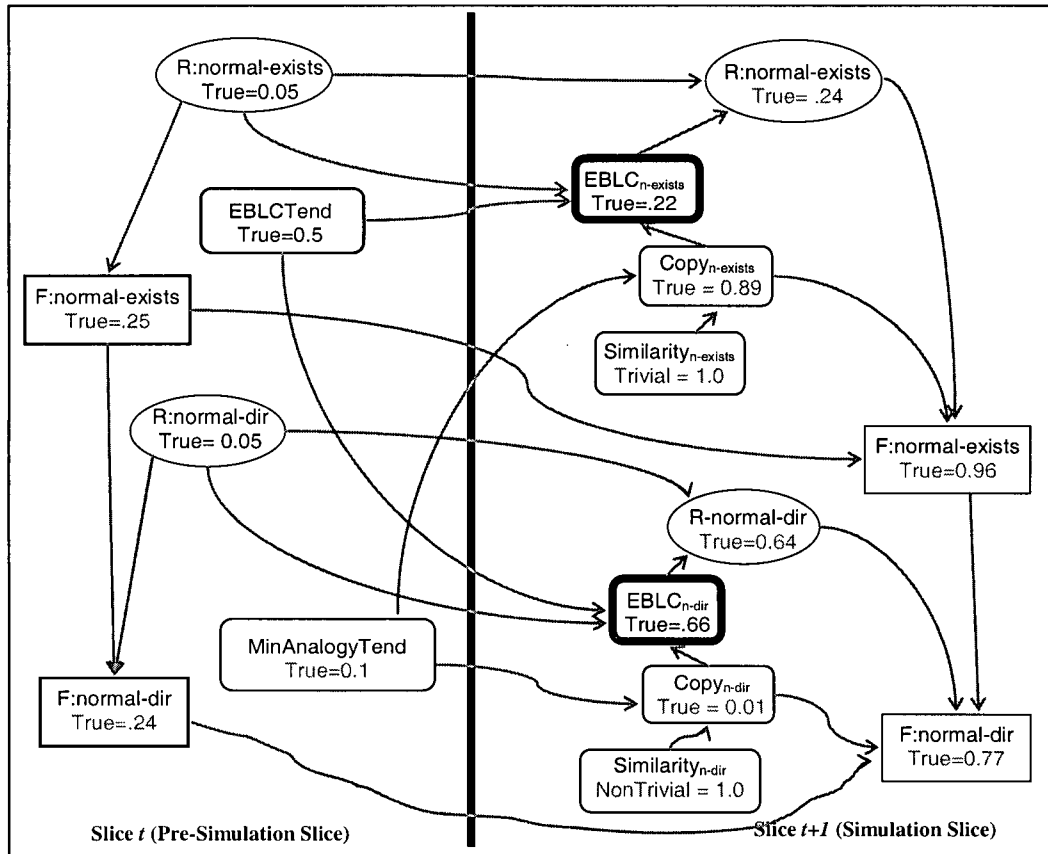


Figure 5-7: EBLC nodes in simulation

Table 5-2: EBLC node CPT

Rule, (Pre-Simulation Slice)	False				True			
	False		True		False		True	
Copy								
EBLC Tend (Pre-Simulation Slice)	False	True	False	True	False	True	False	True
EBLC=False	0.55	0.05	0.99	0.7	0.99	0.99	0.99	0.99
EBLC=True	0.45	0.95	0.01	0.3	0.01	0.01	0.01	0.01

- The student's tendency for EBLC, captured by the parent EBLC-tendency node (e.g., figure 5-7, pre-simulation slice '*EBLCTend*' node, linked to the '*EBLC_{n-dir}*' node in the simulation slice)

The CPT encodes the assumption that if the student already knows the rule, then the probability of EBLC is low, since the student does not need to learn the rule via EBLC. If the student does not know the rule, then the probability that he will reason via EBLC to derive the rule depends on two factors: his tendency for this type of reasoning, and the probability that he will generate the corresponding problem step by copying from the example. The CPT expresses that if there is a high probability the student will not copy, then he has more incentive to reason via EBLC than if he does copy, but this probability is mediated by the model's belief in the student's EBLC tendency. In particular, if the student has a low tendency for EBLC then the probability that he will engage in EBLC is low, even if he does not copy.

The impact of these factors is shown in figure 5-7. The model predicts the student is not likely to reason via EBLC to learn the rule corresponding to the '*R:normal-exists*' node in the simulation slice (simulation slice, '*EBLC_{n-exists}*', *True*=.22). This is because of the high probability that the student will copy the corresponding solution step (simulation slice, '*Copy_{n-exists}*', *True*=.89), and the moderate probability of her having tendency for EBLC (pre-simulation slice, '*EBLCTend*', *True*=.5). In contrast, a low probability of copying (simulation slice, '*Copy_{n-dir}*', *True*=0.01) combined with a moderate tendency for EBLC (pre-simulation slice, '*EBLCTend*', *True*=.5) increases the probability for EBLC reasoning (simulation slice, '*EBLC_{n-dir}*', *True*=.66).

Note that the network's structure with respect to EBLC and copying means that the two are not mutually exclusive, i.e., one can reason via EBLC and still copy. Although this seems unlikely, cognitive science research does not provide clear answers on how EBLC and copying interact, and so we wanted to account for this possibility in our network.

There is one special case pertaining to EBLC nodes, when a rule derives more than one step in the problem solution. In this case, *multiple* copy nodes will exist (i.e., one for each step). In order to avoid having all these nodes as direct parents of the EBLC node, which

would increase the size of the CPT, we use a “*filter*” node to link the copy nodes to the EBLC node (see figure 5-8). The relationship between a filter node and the copy nodes is expressed through a canonical AND-interaction, encoding that its value is *True* if all its inputs are true and *False* otherwise. This allows the model to encode that if one of the steps will not be generated by copying, then there may be incentive for the student to learn via EBLC (but this is mediated by knowledge and tendency).

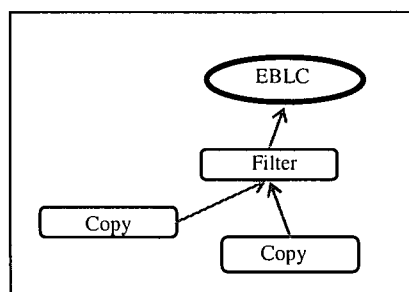


Figure 5-8: Special case for EBLC node

5.4.3 Prediction of Learning and Problem-Solving Success

The student model’s prediction of EBLC and copying influences its prediction of learning and problem-solving success, as follows. The probabilities for the rule nodes in the simulation slice represent the model’s prediction of whether the student will learn the corresponding rule in the presence of the candidate example (e.g., ‘*R:normal-exists*’, ‘*R:normal-dir*’ nodes in the simulation slice from the network in figure 5-5 shown here in figure 5-9). A rule node’s CPT (see table 5-3) encodes how its value depends on the following factors:

- The probability that the student already knows the rule, captured by the corresponding rule node in the pre-simulation slice (e.g., figure 5-9, pre-simulation slice ‘*R:normal-dir*’ node, linked to ‘*R:normal-dir*’ node in the simulation slice)

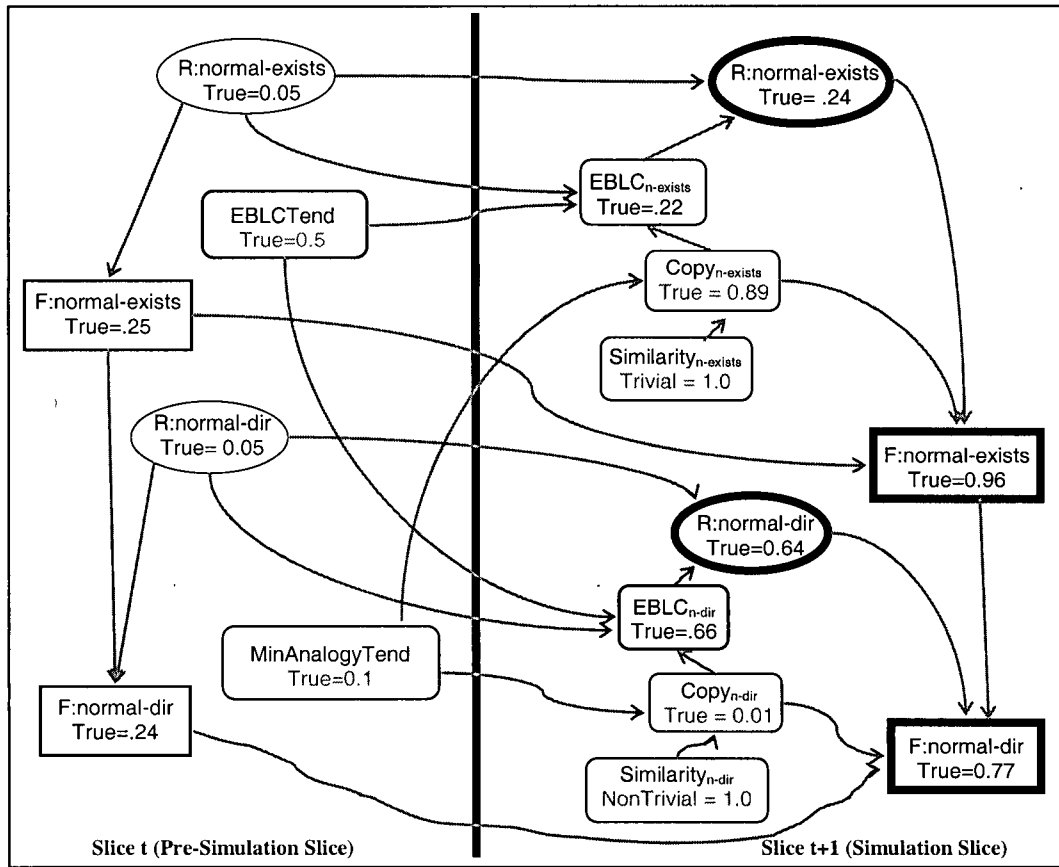


Figure 5-9: Prediction of learning & problem-solving success

Table 5-3: Rule node CPT

Rule (Pre-Simulation Slice)	False		True	
	False	True	False	True
Rule=False	0.99	0.1	0.01	0.01
Rule=True	0.01	0.9	0.99	0.99

- The probability that the student will learn the rule via EBLC, captured by the corresponding EBLC node (e.g., figure 5-9, simulation slice, '*EBLC_{n-dir}*' node linked to the '*R:normal-dir*' node)

Learning of a rule is predicted to occur if the probability of a rule being known is low in the pre-simulation slice and the simulation predicts the student will reason via EBLC to learn the rule (e.g., as is the case for rule '*R:normal-dir*' in the simulation slice in figure 5-9). However even in this case learning is not guaranteed to occur, since cognitive science findings show that students may require several attempts before correct principles are inferred [Chi, 2000].

The probabilities for the problem steps in the simulation slice represents the model's prediction of whether the student will generate a given step in the interface (e.g., '*F:normal-exists*', '*F:normal-dir*' nodes in the simulation slice from figure 5-5 shown here in figure 5-9). A step node's CPT (see table 5-4) encodes how its value depends on the following factors:

- The probability of the corresponding step in the pre-simulation slice, encoding that a step's value depends on its past value (e.g., figure 5-9, pre-simulation slice '*F:normal-dir*' node linked to '*F:normal-dir*' node in the simulation slice)
- The probability that the student will generate the step by copying, captured by the corresponding copy node (e.g., figure 5-9, simulation slice, '*Copy_{n-dir}*' node linked to '*F:normal-exists*' node)
- The probability that the student has the necessary prerequisites to generate the step on her own (i.e., knows the rule for deriving the step and can generate all prerequisite steps). In table 5-4, this is summarized by the '*Parents*' entry (e.g., figure 5-4, simulation slice, '*R:normal-dir*' and '*G:find-forces*' nodes linked to '*F:normal-exists*' node)

Table 5-4: Step (fact) node CPT

Fact (Pre-Simulation Slice)	False				True			
Parents	False		True		False		True	
Copy	False	True	False	True	False	True	False	True
Fact=False	0.75	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Fact=True	0.25	0.99	0.99	0.99	0.99	0.99	0.99	0.99

If a problem step is not already generated in the pre-simulation slice¹⁰, the simulation predicts that the student will generate it either if she copies from the example (as is the case for '*F:normal-exists*' node in the simulation slice in figure 5-9) or by the application of her own knowledge (as is the case for '*F:normal-dir*' node in the simulation slice in figure 5-9).

5.4.4 Accounting for Goals in the Simulation

Planning solution steps encoded in the dynamic Bayesian network using goal ('G:') nodes and corresponding rules are a special case in the simulation, for two reasons. First, the simulation tries to predict if a student will generate the target problem steps and how he will do it (on his own vs. by copying). However, students can never actually generate goal steps in the EA-Coach interface because it does not support the entry of goals (e.g., there is no way for students to specify that they have the goal to, for instance, apply Newton's Second Law). Second, the simulation does not model learning of this class of rules (i.e., ones corresponding to planning knowledge). A key reason for this is that the only mechanism for learning embedded into the student model's operation is EBLC. Although we could have assumed that planning rules are learned via ELBC in the same

¹⁰ This occurs if the student generates a solution step(s) and then asks for an example. Accounting for this means that the model does not need to consider whether an example will help the student generate the step, since she already achieved problem-solving success on that step.

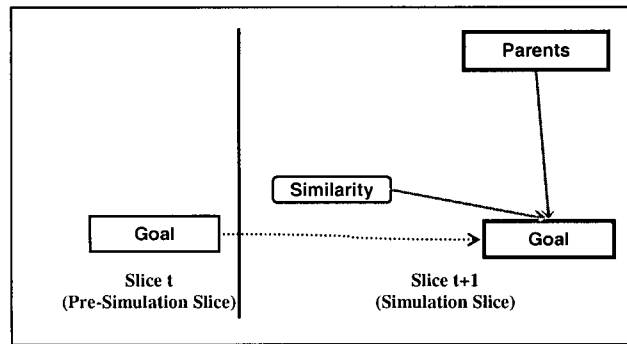


Figure 5-10: Special case in simulation: goal nodes

manner as domain rules, cognitive science research does not describe if or how students learn planning knowledge via EBLC. In addition, the model infers EBLC by taking into account copying, but as we already pointed out, goal steps cannot actually be copied in the EA-Coach interface. Thus, the absence of information on copying makes it even less clear how the model should infer EBLC.

Given these factors, the simulation does not try to predict students' APS behaviors (copying, EBLC) with respect to goals, and so copy/EBLC nodes are not added to the simulation slice for these nodes. However, the example solutions seen by the student in the EA-Coach interface do include solution steps related to goals (e.g., "*We answer this question by using Newton's Second Law*", see figure 4-4b in chapter 4). Consequently, we wanted the simulation to account for the student being able to rely on the example to help him generate the problem solution. To model this, a simplified network structure is used, shown in figure 5-10. Specifically, a similarity node is added for each goal node in the network and directly linked to it (without an intermediate copy node). A goal node's CPT is almost identical to the fact node's CPT, with one exception: the *Similarity* node replaces the *Copy* node in the table (see table 5-5; the *Parents* entry still corresponds to the rule that derived the goal and any pre-requisite proposition nodes in that slice). The CPT encodes the assumption that *Goal=True* if the value of the corresponding similarity node is *Trivial* in the simulation slice, because this type of similarity allows for 'copying'. This is a simplification, since it does not account for a student's tendency and how it will influence copying. However, this allows the simulation to have a very basic

notion of how the example may be helpful for generating the problem solution, without explicitly modeling APS behaviors related to copying and EBLC.

Table 5-5: Goal node CPT

Goal (Pre-Simulation Slice)	False				True			
Parents	False		True		False		True	
Similarity	Trivial	NonTrivial/ None	Trivial	NonTrivial/ None	Trivial	NonTrivial/ None	Trivial	NonTrivial/ None
Goal=False	0.01	0.75	0.01	0.01	0.01	0.01	0.01	0.01
Goal=True	0.99	0.25	0.99	0.99	0.99	0.99	0.99	0.99

5.5 Expected Utility Calculation Phase

Once the simulation phase is completed for a candidate example, the second phase of the example-selection process involves quantifying the student model's prediction in terms of the candidate example's expected utility.

Recall that the EA-Coach has two objectives when choosing an example: (1) learning and (2) problem-solving success. Therefore, the outcome from the example-selection process is characterized by two attributes encoding how these objectives are achieved. For instance, an outcome could be that an example supports problem-solving success, but not learning¹¹. When an outcome is characterized by more than one attribute, a factor that needs to be taken into account during the decision-making process is how the attributes

¹¹ Note that as we pointed out in section 5.1, an example may support *one* of learning or problem-solving success objectives, without necessarily supporting *both*. For instance, an example may help a student achieve problem-solving success, because it allows its solution to be copied; however, because its solution may be copied, the example fails to foster learning.

compare in terms of importance. Such decision problems are handled by multi-attribute utility theory (MAUT) [Keeney and Raifa, 1976; Clement, 1996]. MAUT provides a specification for how to represent a decision maker's preferences between several attributes using various multi-attribute utility functions. The EA-Coach relies on a particular kind of multi-attribute utility function, a linearly additive one, which has the following form:

$$U(x_1, x_2, \dots, x_n) = \sum_{i=1}^n w_i U(x_i)$$

where (1) x_i are the attributes; (2) $U(x_1, x_2, \dots, x_n)$ is the utility of the outcome described by the conjunction of the attributes x_1, x_2, \dots, x_n ; (3) $U(x_i)$ is the utility of an attribute x_i ; (4) w_i is the weight assigned to that attribute encoding its importance. We discuss considerations related to using this kind of a function after we describe how it is used in the EA-Coach framework to calculate an example's expected utility.

As is illustrated graphically in figure 5-11, the EA-Coach obtains an overall expected utility for a candidate example by combining utilities for learning and problem-solving success. These are in turn obtained by combining utilities for the individual rules and problem steps (facts).

To calculate an example's expected utility for learning, first, the expected utility (*EU*) for learning each individual rule in the problem solution is calculated (captured by the nodes '*Rule₁ Utility*', ..., '*Rule_n Utility*' in figure 5-11). This is the sum of the probability P of each possible learning outcome for that rule (i.e., value of a rule node), multiplied by the utility U of that outcome:

$$EU(Rule_i) = P(\text{known}(Rule_i)) \cdot U(\text{known}(Rule_i)) + \\ P(\neg \text{known}(Rule_i)) \cdot U(\neg \text{known}(Rule_i))$$

Since we define $U(\text{known}(Rule_i))=1$ and $U(\neg \text{known}(Rule_i))=0$, the expected utility of a rule corresponds to the probability that the rule is known.

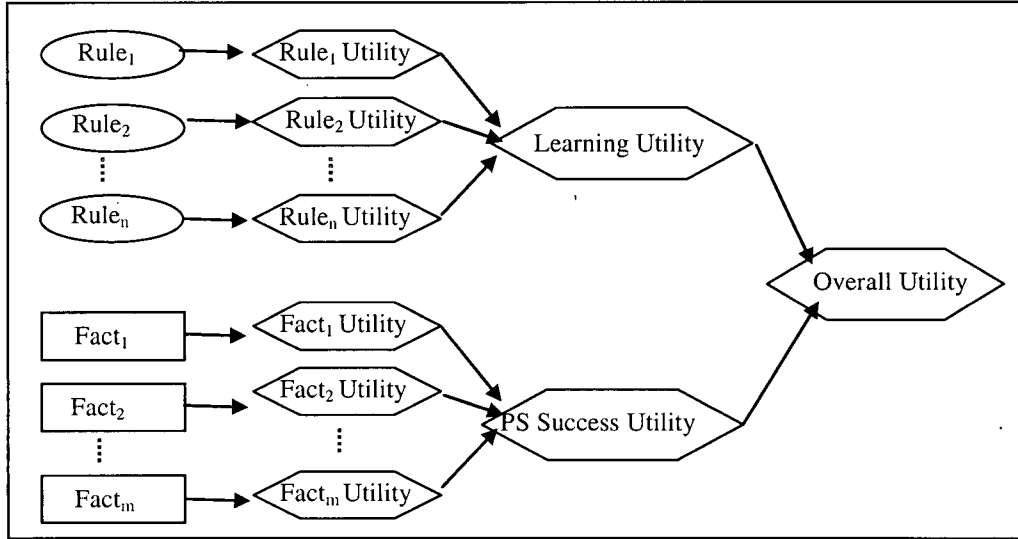


Figure 5-11: The EA-Coach utility model

Next, an example's overall expected utility for learning is calculated (captured by the node '*Learning Utility*' node in figure 5-11). This is the weighted sum of the expected utilities for learning the individual rules:

$$\sum_i^n EU(Rule_i) \cdot w_i$$

Since we consider all the rules to have equal importance, all the weights w are assigned an equal value (i.e., $1/n$, where n is the number of rules in the network that can be derived via EBLC).

The same approach is used to obtain the measure for the problem-solving success objective. First, the expected utility (EU) of each fact being generated is calculated (captured by the nodes '*Fact₁ Utility*', '*Fact₂ Utility*' in figure 5-11). This is the sum of the probability P of each possible problem-solving outcome for that fact (i.e., value of a fact node), multiplied by the utility U of that outcome:

$$EU(\text{Fact}_i) = P(\text{generated}(\text{Fact}_i)) \cdot U(\text{generated}(\text{Fact}_i)) + \\ P(\neg \text{generated}(\text{Fact}_i)) \cdot U(\neg \text{generated}(\text{Fact}_i))$$

As was the case with expected utility for learning an individual rule, we define $U(\text{generated}(\text{Fact}_i))=1$ and $U(\neg \text{generated}(\text{Fact}_i))=0$.

An example's overall expected utility for problem-solving success is the weighted sum of the expected utilities for the individual facts (captured by the node '*PS Success Utility*' node in figure 5-11):

$$\sum_i^m EU(\text{Fact}_i) \cdot w_i$$

As was the case for rule nodes, we assume that all the facts are equally important, and so the weights are assigned an equal value (i.e., $1/m$, where m is the number of facts in the network).

The expected utilities for learning and problem-solving success are combined in a weighted sum to obtain an example's overall expected utility (captured by the node '*Overall Utility*' node in figure 5-11):

$$\text{Overall } EU(\text{example}) = EU(\text{Learning}) \cdot w_1 + EU(\text{PS Success}) \cdot w_2$$

The weights w_1 and w_2 are currently set to the same value (i.e., $1/2$), since we assume learning and problem-solving success to be equally important.

5.5.1 Using a Linearly Additive Multi-Attribute Utility Function: Implications

Using a linearly additive multi-attribute function greatly simplifies the utility computation, since the utilities can be specified and calculated independently of each other. However, a linearly additive function assumes additive independence among the agent's preferences. To illustrate additive independence, suppose we have two scenarios A and B, where a scenario corresponds to two possible outcomes, each with a 0.5 probability of occurring:

- A (low Learning, low PS Success) with probability 0.5
 (high Learning, high PS Success) with probability 0.5
- B (low Learning, high PS Success) with probability 0.5
 (high Learning, low PS Success) with probability 0.5

For instance, in scenario A, the two outcomes are (1) both learning and problem-solving success are low and (2) both learning and problem-solving success are high. If the decision maker is indifferent between the two scenarios then additive independence holds. However, if the decision-maker prefers scenario B over A because she prefers that at least one of *PS Success* or *Learning* is realized and does not want to risk neither happening, then additive independence does not hold. More formally, additive independence is defined as “*changes in lotteries in one attribute do not affect preferences for lotteries in the other attribute*” [Clement, 1996], where a lottery (i.e., scenario in our example above) is a probability distribution over outcomes. In the context of the EA-Coach, whether additive independence holds depends on the agent whose preferences the tutor is emulating (an instructor’s, the researcher’s, etc.) However, assuming additive independence even when it does not hold has been shown to be acceptable in many practical applications [Clement, 1996].

5.6 Example-Selection Process: Summary

The EA-Coach example-selection process involves the following two phases for a candidate example:

1. *Simulation phase*: generating a prediction of how the student will solve a problem in the presence of the candidate example and what she will learn from it
2. *Expected utility calculation phase*: quantifying this prediction via a linearly additive multi-attribute utility function to obtain the candidate example’s expected utility

When a candidate example’s expected utility is calculated, its simulation slice is discarded from the network. The simulation and utility calculation phases are repeated for

each example in the framework's pool and the example with the highest expected utility is presented to the student.

5.7 Role of the Example in the Student Model's Assessment

Once an example is selected and presented to the student, the student model *assesses* how the student actually uses the example to solve the target problem and how her knowledge and meta-cognitive tendencies evolve as a consequence. Although the same network and parameters are used in assessment mode as during simulation mode, there are two key differences between the two modes. First, during assessment the model generates its appraisal as a *response* to student actions. Specifically, a new slice is added to the network each time a student produces a correct problem-solving entry¹². To illustrate this, let's consider the following scenario. Suppose a student opens the problem in figure 2-3, which results in the construction of a dynamic Bayesian network based on that problem's solution graph (see slice t in figure 5-12). Next, the student asks for an example and the EA-Coach presents her with the one shown in figure 2-3. The student then generates a correct problem-solving entry corresponding to specifying the normal force. In response, a new slice is added to the dynamic Bayesian network (e.g., see slice $t+1$, figure 5-12). This slice initially contains the same solution graph and APS tendencies nodes as the slice before it. In the newly added slice (see slice $t+1$, figure 5-12), the following occurs:

- A similarity, a copy and an EBLC node is added and linked to the fact and rule nodes corresponding to the problem-solving entry (see '*Similarity_{n-exists}*', '*Copy_{n-exists}*', '*EBLC_{n-exists}*' nodes in slice $t+1$, figure 5-12). The value of the similarity node is set using the same algorithm as during simulation. Recall that to do so, the

¹² Slices are only added for correct entries because the model only contains a representation of the correct problem solution. Although incorrect entries could be used to infer that a student does not know a particular rule or that she is incorrectly copying from an example, modeling this would require determining which entry the student was trying to generate. Doing so is not straightforward and so for the time being is not implemented.

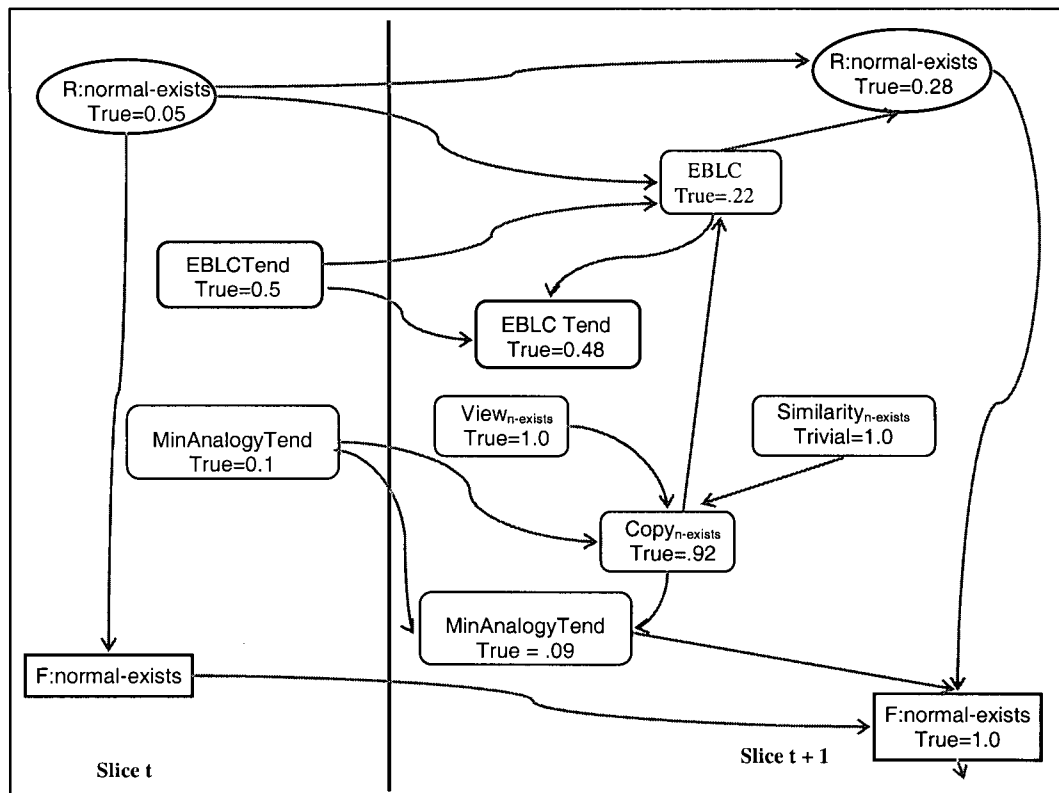


Figure 5-12 : Fragment of the EA-Coach dynamic Bayesian network used during assessment mode

framework (1) identifies which example step to compare with the problem step corresponding to the student's entry and (2) assesses the similarity between the problem/example steps. During assessment mode, the framework stores the 'id' of the example step that is compared with the problem step, referred to as '*exampleStepId*' below (we explain what purpose this serves shortly).

- The student's problem-solving entry is entered as evidence into the network by setting the value of the corresponding fact node in the network to 'True' (see '*F:normal-exists*' node in slice *t+1*, figure 5-12).

- A ‘View’ node is added and linked to the fact node corresponding to the student’s problem-solving entry (see ‘ $View_{n-exists}$ ’ node in slice $t+1$, figure 5-12). The view node encodes information on students’ example viewing, as we describe below.

Therefore, a second key difference between simulation and assessment is that during assessment the model has evidence on student actions, including (1) problem-solving entries and (2) example-viewing actions in the masking interface (stored in the *viewing history*, i.e., the sequence of steps a student viewed in the example). Problem-solving entries help the model infer that a student knows a given rule. In particular, if the probability of copying is low, then evidence from a student’s entry results in belief propagation to the parent rule node, increasing the probability that the student knows the rule. If, on the other hand, the probability of copying is high, then the copy node ‘explains away’ much of the evidence so that it does not propagate to the parent rule node. This reflects the assumption that copied entries do not provide evidence that a student knows the rule needed to generate the problem step. To help assess copying, the model relies on information coming from a student’s example-viewing actions in the masking interface. Specifically, the model takes into account whether prior to generating a problem-solving entry, the student actually viewed the corresponding step in the example solution. This information is encoded in the network by the view node. The node’s value is binary (*True/False*) and is set to:

- *True* if prior to generating her entry, the student viewed a corresponding step in the example solution, i.e., ‘*exampleStepId*’ is in her viewing history. The search for a step in the viewing history is constrained to the last *three* steps viewed since the last problem-solving entry. This encodes the assumption a student would not remember example steps she viewed further back.
- *False* otherwise.

Table 5-6: Copy node CPT used during assessment

View	False				True			
MinAnalogyTend	False		True		False		True	
Similarity	Trivial	NonTrivial/ None	Trivial	NonTrivial/ None	Trivial	NonTrivial/ None	Trivial	NonTrivial/ None
Copy=False	0.99	0.99	0.99	0.99	0.01	0.01	0.4	0.01
Copy=True	0.01	0.01	0.01	0.01	0.99	0.01	0.6	0.01

The View node is linked to the copy node (see figure 5-12). Therefore, the probability a student generated the entry by copying depends on (see table 5-6) the following factors:

- The similarity between the problem step the copy node refers to and the corresponding step in the example solution, captured by the parent similarity node
- The student's tendency for min-analogy, captured by the parent MinAnalogyTend node in the previous slice
- Whether the student viewed the corresponding step in the example, encoded by the view node

As is the case during simulation mode, if the value of the parent similarity node is *NonTrivial*, then the probability of copying is virtually zero. On the other hand, if the value of the similarity node is *Trivial*, then the example allows a student to generate the problem solution step by copying. In this case, the probability of copying depends on the student's existing tendency for min-analogy and whether she viewed the corresponding step in the example solution, encoded by the view node. If the value of the view node is *False*, then the probability of copying is very low, even if the student has a low tendency for min-analogy. If, on the other hand, the value of the view node is *True*, then probability of copying is increased (although this increase is mediated by the student's tendency for min-analogy). For instance, if a student viewed a step in the example solution and then generated the corresponding step in the problem solution and the

similarity between the problem step and example step was trivial, then the probability of copying is increased (even if the student has a high tendency for min-analogy).

A student's example-viewing actions are also used to determine whether to add an EBLC node to the network. Specifically, an EBLC node is added if (1) the rule can be derived EBLC and the example affords the student the opportunity to learn the rule, i.e., a structurally identical step exists in the example and (2) the student viewed the corresponding step in the example (i.e., '*exampleStepId*' is in her viewing history). If the EBLC node is added, its value and influence on the rule node it is linked to depends on the same factors as during simulation, described in section 5.4.2.3 (i.e., identical CPT is used during simulation and assessment modes). If the EBLC node is not added, then the rule corresponding to the student's problem-solving entry only has one parent: the rule in the previous slice. In this case, the value of the parent rule node is simply transferred over to the child rule node.

The model uses information on copying and EBLC to assess a student's corresponding meta-cognitive tendencies in the new slice. To assess tendency for min-analogy, the model uses its appraisal of the student's copying behaviors. For instance, belief in the student having copied (e.g., '*Copy_{n-exists}*', *True*=.93 in figure 5-12) decreases the model's belief in the student's tendency for min-analogy ('*MinAnalogyTend*', *True*= .1 in slice *t-1* decreases to *True* =.09 in slice *t*). To assess tendency for EBLC, the model uses its appraisal of EBLC episodes. For instance, if the probability of EBLC is low, then belief in EBLC tendency to decrease ('*EBLCTend*', *True*=.5 in slice *t-1* decreases to *True* =.48 in slice *t*). Note that if an EBLC node is not added, then the EBLC tendency node has as its only parent the EBLC tendency node in the previous slice. In this case, the value of the parent tendency node is simply transferred over to the child tendency node.

5.8 Summary

In this chapter, we provided details on the computational mechanisms enabling the EA-Coach to provide adaptive support for APS through its example-selection mechanism,

corresponding the framework's student model and expected utility calculation components.

In the next chapter, we describe the evaluation we conducted to evaluate the pedagogical effectiveness of EA-Coach and its example-selection mechanism.

Chapter 6

Evaluation of the EA-Coach

In this chapter, we describe the controlled laboratory study we conducted to formally evaluate the effectiveness of the EA-Coach [Muldner and Conati, 2007]. Since the primary form of support delivered by the EA-Coach corresponds to its example-selection mechanism, the study focused on this component. We begin by specifying our study objectives and the approach we adopted to achieve them. After presenting the particulars of the study, we describe the dependent measures we considered in our data analysis, and then devote the remainder of the chapter to describing the study results.

6.1 Objectives & Approach

As we mentioned in earlier chapters, the challenge for the EA-Coach example-selection mechanism is to choose examples that are different enough to trigger learning by encouraging effective APS behaviors (*learning goal*), but at the same time similar

enough to help the student generate the problem solution (*problem-solving success goal*). To verify how well the EA-Coach example-selection process meets these goals, we compared it with a selection approach corresponding to choosing examples that are as similar as possible to the target problem. The latter is the standard approach advocated by existing ITS that perform example selection [Weber, 1996b; Aleven and Ashley, 1997; Nogry, Jean-Daubias et al., 2004]. In contrast, the EA-Coach selection mechanism may choose examples that include some differences with the target problem.

In order to compare how the two example-selection approaches influence learning, we focused on analyzing how each approach influences APS *behaviors* that impact learning outcomes, i.e., min-analogy and EBLC. Our hypothesis was the following:

the EA-Coach's example-selection strategy will be more effective than the strategy of selecting maximally similar examples in terms of encouraging EBLC and min-analogy.

The approach of assessing learning by performing a fine-grained analysis of behaviors that impact learning outcomes is advocated in [Chi, Bassok et al., 1989], where it is used to assess students' domain understanding by analyzing their self-explanations. Although this approach makes the analysis challenging because it requires that students' reasoning is captured and analyzed, it does have the advantage of providing in-depth insight into how each example-selection strategy impacts learning via the relevant meta-cognitive processes. Another way to measure learning is via pre/post test differences (i.e., if students show a gain from pre to post test, then this shows they learned). Although we did conduct some analysis using this measure, this was not the focus of our evaluation, for two reasons. First, it is hard to use this approach with the within-subjects design we adopted (as we describe in 6.6.1.3). Second, this approach does not provide *fine-grained* information on the process of learning, i.e., it does not tell us *why* learning did or did not occur.

6.2 Study Participants

The study participants were university students who

- were either (1) in the process of taking the Physics 100 course at UBC (equivalent to a grade twelve physics course) or (2) had completed either Physics 100 or a grade twelve physics course at any point prior to the study;
- had not taken any higher-level physics courses.

Subjects were paid ten dollars per hour for their participation. Since we needed subjects who provided information on APS behaviors, and our pilot studies showed that students who have a high level of expertise are less likely to use examples while problem solving, we pre-screened subjects based on knowledge. Specifically, subjects were given a pre-test (see section 6.3), and those who answered all the questions correctly did not participate in the remainder of the study. In addition, remaining subjects who did not use *any* examples during the study were not included in the analysis.

A total of thirty-two subjects signed-up for the experiment and completed the pre-test. Eleven subjects answered all the pre-test questions correctly and did not participate in the remainder of the study. Of the remaining twenty-one subjects, two were discarded. One subject was discarded because of system instability (the subject accidentally received an older unstable version of the system which compromised her interaction with the interface). The second subject was discarded because she did not follow the experimental procedure: she took a prolonged phone break during the pre-test phase and did not diligently answer the pre-test questions, which could have confounded the results. Three subjects did not access any examples during the experimental phase of the study and so they did not complete the remainder of the experiment (i.e., the post-test), nor were they included in the data analysis.

This left us with data from 16 subjects (14 were female). We describe how this data was used in the analysis in section 6.6, when we present the results.

6.3 Study Methodology

The methodology for the study evolved from the pilot evaluations we described in chapter 3. In particular, the study methodology was based on the primary pilot's and was refined using the two follow-up pilots. The key refinement to the methodology corresponded to the addition of a 'training phase' that introduced subjects to the EA-Coach, added after the first follow-up pilot. Recall that the first follow-up pilot was used to evaluate the addition of feedback for correctness to the system. As a consequence of this addition, participants were required to use a more rigorous solution-entry format than they did in the primary pilot. The first follow-up pilot revealed that some participants had difficulty with the rigorous format. To address this issue, in addition to modifying the format of the example window (see chapter 3), we added a training phase to the study that introduced participants to the EA-Coach interface. We piloted this addition with the second follow-up pilot.

As was the case with the pilots, we used a within-subject design for the evaluation of the EA-Coach, because it increases an experiment's power to detect differences between conditions. This is accomplished by exposing each subject to all the conditions, thus accounting for the variability between subjects arising from individual differences in, for instance, expertise, APS tendencies and verbosity (which impacts verbal expression of self-explanation/EBLC). Each participant

1. completed a pencil and paper pre-test (subjects were given 30 minutes to complete the pre-test);
2. was given a 10 minute break during which her pre-test was graded and the pre-test data was used to initialize the system parameters (*initialization phase*);
3. was introduced to the EA-Coach Interface, which took approximately 10 minutes (*training phase*);
4. solved two physics problems using the EA-Coach (*experimental phase*);
5. was given a 5 minute break;

6. completed a pencil and paper post-test (subjects were given 30 minutes to complete the post-test).

We begin with the descriptions of the training and experimental phases.

6.3.1 Training Phase

During the training phase, subjects were introduced to the functionalities in the EA-Coach interface. Subjects were first given a high-level overview of how to enter a problem solution and access an example in the EA-Coach. To solidify the process in subjects' memory, they were then asked to (1) use each of the tools needed to enter the problems solution and were guided through this process; (2) access an example. To avoid any carry-over effects from training, we used a problem and an example that contained 'place-holders' instead of actual physics content. For instance, instead of actual physics problem/example specifications, the problem/examples had text that specified "*here you will see a physics problem*". The example solution contained just enough generic text (e.g., "*equation 1*", "*Force F*" etc.) to provide a reference point during the training phase, for instance, to allow us to point out that the variable pane contained the definitions of the variables that appeared in the example solution.

6.3.2 Experimental Phase

During the experimental phase, subjects used the EA-Coach interface (figure 6-1) to solve two physics problems (p_1 and p_2 , section 6.4). Subjects were instructed on which problem to open first, and used the '*File Open*' menu to do so. Once subjects finished a problem and closed it, they were not allowed to re-open it. To control the overall experiment time while providing adequate opportunity to generate the problem solution, subjects were given 60 minutes per problem. Although all the pilot subjects required less time to generate the solutions (on average, 26 minutes during the two follow-up pilots), we set the threshold higher because there was considerable variability between the pilot subjects' task time (the fastest being 15 minutes, while the slowest was 41 minutes).

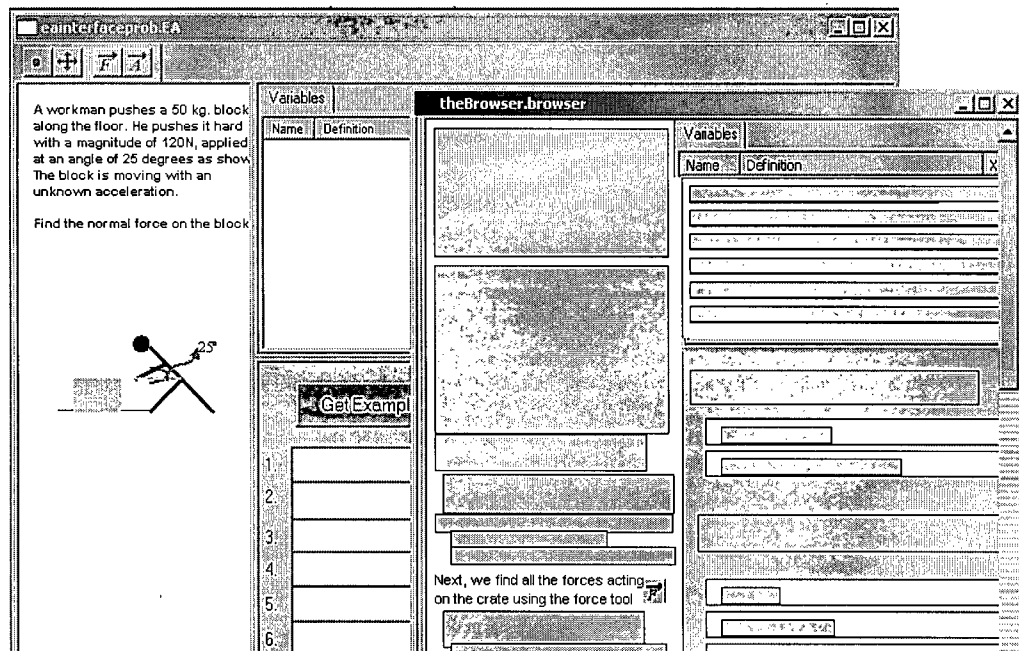


Figure 6-1: The EA-Coach interface

For each problem, subjects had the option to refer to one example (opened by clicking on the 'Get Example' button in figure 6-1). Subjects were instructed to treat this phase of the study as a homework situation, where they had some problems to solve, had access to a worked-out example, and were trying to both do their homework and prepare for an upcoming test. Subjects were told that it was up to them to use the examples and if they did, how to use them.

To compare the EA-Coach's example-selection approach to the standard approach, the example-selection strategy was manipulated as follows:

[static-selection condition] For one of the problems, subjects received an example that was highly superficially and structurally similar to the problem (as will be shown

in section 6.4)¹². We refer to this selection type as *static* because the choice of example did not depend on a subject's characteristics and so was constant for all students for a given problem.

[*adaptive-selection condition*] For the other problem, the example was adaptively selected by the EA-Coach at run-time from the pool of examples (presented in section 6.4). In order to maximize learning and problem-solving outcomes, the EA-Coach adaptively tailors the choice of the example for each student (as we describe in section 6.3.3, for the study the example-selection mechanism was initialized using subjects' pre-test data). Thus, in contrast to the statically selected example, the adaptively selected example is not necessarily the most similar to the target problem.

Therefore, the evaluation included two conditions: a static-selection condition and an adaptive-selection condition; to clarify the following discussion, we will also refer to the two conditions simply as static or adaptive. To account for carry-over effects, the presentation order of the problems and the conditions was fully counterbalanced, with subjects assigned to the problem/condition combinations in a round-robin fashion. Since the study involved two problems (p_1 and p_2) and two conditions (static and adaptive selection), counterbalancing resulted in four problem/condition combinations:

- p_1 /static selection, p_2 /adaptive selection
- p_1 /adaptive selection, p_2 /static selection
- p_2 /static selection, p_1 /adaptive selection
- p_2 /adaptive selection, p_1 /static selection

Both conditions used the version of the EA-Coach interface described in chapter 4. In particular, the EA-Coach provided feedback for correctness on subjects' problem-solving entries, realized by coloring the entries red or green for correct and incorrect entries,

¹² This pairing was accomplished by hand before run-time – but the framework could have performed this selection at run-time by selecting the example that had the fewest differences with the corresponding problem.

respectively. The coach also informed subjects when it could not *interpret* their entries because of syntactic problems related, for instance, to undefined variable names. The coach did not, however, provide any other hints or interventions related to domain-specific help or APS strategies.

During the experimental phase, we used the same data collection techniques as during the primary pilot (see chapter 3), which included (1) using the talk-aloud technique [Ericsson and Simon, 1980] to capture students' reasoning, and videotaping and subsequently transcribing all sessions and (2) logging all student actions in the problem and example windows of the EA-Coach interface.

6.3.3 Initialization Phase

During the initialization phase that preceded the training/experimental phases described above, we initialized the EA-Coach example mechanism. As we described in chapter 5, the mechanism takes into account the similarity between the target problem and candidate example, as well as information about a student's knowledge and meta-cognitive tendencies. The mechanism still functions without being initialized with student-specific information, but in this case its operation is not tailored to a given student's characteristics at the onset of the interaction. Initializing the selection mechanism corresponds to setting the priors for knowledge and APS tendency nodes in the Bayesian network student model.

To initialize the knowledge node priors we relied on data from the pre-test, which was designed to provide information on students' physics knowledge. Specifically, the pre-test contained several questions corresponding to each relevant physics rule involved in the problems/example solutions in the experimental phase (i.e., knowledge node in the Bayesian network). To illustrate this, figure 6-2 shows the three pre-test questions and corresponding marking scheme used to initialize the priors for the *normal-exists/normal-dir* rules. If a student

- answered all questions correctly, the corresponding rule was assigned a prior of 0.95 to reflect the probability that the student most likely knew the rule;

- answered all questions incorrectly, the corresponding rule was assigned a prior of 0.05, to reflect the probability that the student likely did not know the rule;
- answered some questions correctly, the corresponding rule was assigned a prior based on the percentage of correct responses, to reflect the probability that the student had *some* knowledge of the rule.

For instance, if the student drew the normal force for all three corresponding questions in figure 6-2, but only drew the force's direction correctly in the first two questions, then the prior for the *normal-exists* rule was set to 0.95, while the prior for the *normal-dir* rule was set to 0.67.

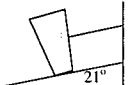
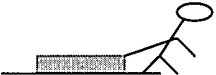
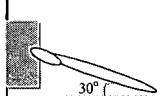
<p>Problem 1 A vase with a mass of 2 kg sits on a frictionless shelf that is inclined 21 degrees from the horizontal. The vase is held in place by a taut string that runs parallel to the incline. Draw all the forces/ their direction that are acting on the vase.</p>  <p>Normal Force drawn: 1 point Direction correct: 1 point</p>
<p>Problem 2 Bob pulls a large package along the ground with a force of $F=100\text{N}$, applied at 34 degrees to the horizontal. Draw the acceleration vector and all the forces that are acting on the package.</p>  <p>Normal Force drawn: 1 point Direction correct: 1 point</p>
<p>Word Problems: Problem 2 Jane pushes a 4 kg book into the wall. She applies the push at an angle of 30 degrees with the horizontal, and a magnitude of 105 N. The book is sliding up the wall with some acceleration. What is the acceleration of the block ? (assume no friction!) Please draw free-body diagram as part of your answer, include the acceleration vector</p>  <p>Normal Force drawn: 1 point Direction correct: 1 point</p>

Figure 6-2: Pre-test questions/corresponding marking scheme used to initialize the *normal-exists/normal-dir* rules

Obtaining information to initialize the APS tendency priors proved more challenging. We are not aware of the existence of a cognitive test to assess a student's APS tendencies. One possibility we considered to obtain this information involved adding a third problem-solving phase to our experiment. Specifically, we would use the above-described design, but prior to the experimental phase, we would introduce a 'baseline' phase. In this phase, students would solve a problem with access to a statically selected example, and the model would assess the students' tendency by monitoring their APS behaviors. Recall that statically selected examples are highly similar to the target problem, allowing students to copy as much or as little as they desire, thus providing an opportunity to observe their inherent tendencies. Unfortunately, adding this phase would mean introducing another static condition that *always* appears first. Consequently, our experiment would no longer be counterbalanced, since the adaptive condition would never appear as the first trial. This would make it difficult to draw conclusions about the adaptive condition *in isolation* from the baseline phase. Given this consideration we decided against this option and instead chose to initialize the tendency node priors to 0.5 for all the subjects.

As we indicated above, the initialization phase occurred prior to the training/experimental phases. The model did not update this assessment of knowledge/meta-cognitive tendencies during the study (i.e., it was not used in assessment mode). This decision was based on the following two factors:

- Due to the need to counterbalance the study conditions, allowing the assessment to be updated during the study would mean that the EA-Coach example-selection mechanism had varying amounts of information about a student's tendency (i.e., depending on whether the adaptive condition was first, or followed the static condition), which could bias the results.
- It allowed us to evaluate the example-selection mechanism in isolation from the model's assessment.

6.4 Study Materials

The evaluation involved two types of materials: pencil and paper pre and post tests, as well as problem and examples students solved and referred to during the experimental phase of the study.

6.4.1 Pre and Post Tests

The pre and post tests included the same number of problems (7 problems, some with sub-parts, see Appendix 3). The pre and post tests were based on the ones used for the primary pilot study described in chapter 3, but expanded to allow for a fine-grained assessment of students' physics knowledge (this refinement was piloted in the two follow-up pilots). Specifically, the problems were designed to provide information on subjects' knowledge of individual rules in the EA-Coach Knowledge Base, used to initialize the example-selection mechanism. To improve accuracy, the tests included several questions per concept to account for the possibility of slips, etc. To motivate students to answer the post-test questions, we designed the pre and post tests to have equivalent questions, but we varied the constants and objects that appeared in the problem specifications.

6.4.2 The EA-Coach Problems and Examples

The problems (see table 6-1) and the examples (see table 6-2) used during the experimental phase of the study are based on typical "Newton's Second Law" problems used in physics courses (identified through on-line searches / textbooks (e.g., [Halliday and Resnick, 1988])). However, we did not take all of the problems and examples directly from a textbook, because we needed ones that satisfied particular constraints in terms of their similarity to each other (for instance, we needed two examples that included only trivial superficial differences with the problem). Since such problems/examples were not in present textbooks, we had to design them ourselves.

Table 6-1: Problems used during the experimental phase

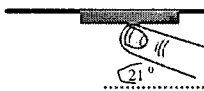
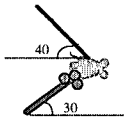
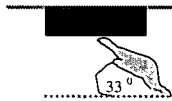
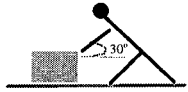
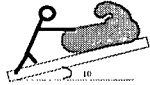
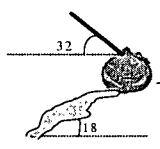
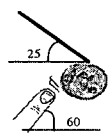
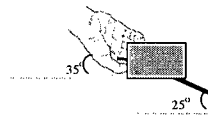
Problem label	Problem specification
p ₁	<p>Barb has decided to attach a picture to her ceiling. To see what it would look like, she holds it against the ceiling. She is applying a pushing force of 190N at an angle of 21 degrees to the horizontal. The picture has a mass of 3 kg and is moving along the ceiling with some unknown acceleration. What is the magnitude of the normal force on the picture?</p> 
p ₂	<p>A toy mouse of some unknown mass hangs from the ceiling on a string. A playful cat pushes the mouse with a force of 50N, at an angle of 30 degrees to the horizontal, as shown. At this point, the mouse is not moving and the string makes an angle of 40 degrees with the horizontal. What is the tension in the string?</p> 

Table 6-2: Examples included the EA-Coach example pool during the experimental phase

Example label	Example specification
e ₁	<p>Jake is trying to install a light fixture of mass = 2 kg. He is pushing it into the ceiling with a magnitude of 180 N. He is applying his force at an angle of 33 degrees to the horizontal and the fixture is moving along the ceiling, with some unknown acceleration. What is the magnitude of the normal force on the fixture?</p> 
e ₂	<p>Greg is pushing a crate of mass 40 kg along the floor. He pushes with a magnitude of 65 N. This force is directed downwards and applied at an angle of 30 degrees from the horizontal. The crate is moving with some unknown acceleration. Find the magnitude of the normal force on the crate.</p> 
e ₃	<p>A child is pushing a sled of mass 4 kg up a hill inclined at 10 degrees to the horizontal. This pushing force is applied parallel to the horizontal (i.e., at 0 degrees), but we don't know its magnitude. The crate is moving with an acceleration of 1.2 m/s². Find the magnitude of the normal force on the sled.</p> 

Example label	Example specification
e ₄	<p>A pumpkin of some unknown mass is suspended by a cord and pushed by Ann with a 45 N force until the cord forms an angle of 32 degrees with the horizontal (as shown). At this point, the pumpkin is at rest. The pushing force is applied at an angle of 18 degrees with the horizontal. What is the tension in the cord?</p> 
e ₅	<p>A yoyo (fully unwound) of mass 2 kg is hanging from a string. Jane is pushing the yoyo so that its string makes an angle of 25 degrees with the horizontal – at this point, the yoyo is at rest. Jane applies the pushing force at an angle of 60 degrees to the horizontal, as shown. We don't know the magnitude of the pushing force. What is the tension in the string?</p> 
e ₆	<p>A block is attached to a string, which is attached to the ground. Bob pulls on the block with a force that has a magnitude of 40 N, applied at an angle of 35 degrees to the horizontal. At this point, the block is at rest. The angle the string makes with the horizontal is 25 degrees. We don't know the mass of the block... What is the tension in the string?</p> 

The two problems were based on the ones involved in the primary pilot. However, we replaced problem_{trivial} (see table 3-1) with a similar problem. This was prompted by one subject's comment during the primary pilot, who said that "*this problem [problem_{trivial}] is a little different ...*" and then when she opened problem_{non-trivial} "*this is the problem I'm really familiar with*". Based on an informal search through on-line physics problem sets, as well as a standard physics text book [Halloun and Hestenes, 1985], problem_{trivial} did indeed appear to have a less-common problem situation than problem_{non-trivial}. Thus, to make the two problems more equivalent in terms of familiarity, we replaced problem_{non-trivial} with problem p₁, one that required the same domain knowledge to generate the solution but that appeared to involve a less-common problem situation.

As we pointed out in chapter 3, our goal was to choose problems equivalent in terms of difficulty but different enough so that it made sense to ask students to solve them.

Table 6-3: Structural Differences between problems p_1 and p_2

Problem P1	Problem P2
Normal force exists, acceleration vector exists, inclined-vector component equation (y-axis), parallel-vector component equation.	Tension force exists, acceleration is null, inclined-vector component equation (x-axis).

However, satisfying these two constraints meant that the two problems included some structural differences (summarized in table 6-3). These differences are the same as were involved in the primary pilot (with one exception discussed shortly). Since the primary pilot suggested that the structural differences did not impact subjects' behaviors, we felt they would satisfy the two constraints listed above. The one exception is that the primary pilot problems did not include a structural difference relating to vector $\{x/y\}$ -component equations inclined with respect to an axis. In the primary pilot, these were represented using the same form and so did not correspond to a structural difference, i.e., were derived by the same rule. Recall that after the primary pilot, we modified the representation of vector component equations, which meant that the equations were derived by two rules (i.e., one to generate equations for the x-component and one to generate equations for the y-component, $F_x = F * \cos\{angle\}$ and $F_y = F * \sin\{angle\}$, respectively).

In the static condition, problem p_1 was paired with example e_1 and problem p_2 was paired with example e_4 (for convenience, the pairs are shown in table 6-4). We summarize the superficial differences between the two problem/example pairs in the static condition in table 6-5 and table 6-6.

Table 6-4 : Problem/example pairs used in the static condition



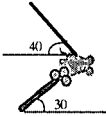
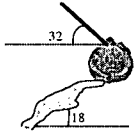
Problem p_1		Corresponding Example e_1	
<p>Barb has decided to attach a picture to her ceiling. To see what it would look like, she holds it against the ceiling. She is applying a pushing force of 190N at an angle of 21 degrees to the horizontal. The picture has a mass of 3 kg and is moving along the ceiling with some unknown acceleration. What is the magnitude of the normal force on the picture?</p>		<p>Jake is trying to install a light fixture of mass = 2 kg. He is pushing it into the ceiling with a magnitude of 180 N. He is applying his force at an angle of 33 degrees to the horizontal and the fixture is moving along the ceiling, with some unknown acceleration. What is the magnitude of the normal force on the fixture?</p>	
<p>A toy mouse of some unknown mass hangs from the ceiling on a string. A playful cat pushes the mouse with a force of 50N, at an angle of 30 degrees to the horizontal, as shown. At this point, the mouse is not moving and the string makes an angle of 40 degrees with the horizontal. What is the tension in the string?</p>		<p>A pumpkin of some unknown mass is suspended by a cord and pushed by Ann with a 45 N force until the cord forms an angle of 32 degrees with the horizontal (as shown). At this point, the pumpkin is at rest. The pushing force is applied at an angle of 18 degrees with the horizontal. What is the tension in the cord?</p>	

Table 6-5: Superficial differences between problem p_1 & corresponding example e_1 in the static condition

Simplified fragment of solution step corresponding to the difference	Problem Solution Step	Corresponding Example Solution Step	Type of Difference
Body to apply Newton's 2 nd Law	picture	light fixture	Trivial
The mass of the ... is ...	picture / 3kg	light fixture / 2kg	Trivial/ Trivial
There is a pushing force on ... due to ...	picture / Barb	light fixture / Jake	Trivial/ Trivial
The pushing force direction is drawn ... CCW ¹³ from the horizontal	159°	147°	Non- Trivial ¹⁴
The pushing force magnitude is ...	190N	180N	Trivial
There is a weight force on the ...	picture	light fixture	Trivial
The magnitude of weight force is 9.8 * ...	3kg	2 kg	Trivial
There is a normal force on ... due to the ceiling	picture	light fixture	Trivial
The y-component equation of the pushing force is ...	$P \sin (21)$	$P \sin (33)$	Trivial

¹³ counter-clockwise

¹⁴ Note that this difference is classified as non-trivial because it cannot be resolved by transformational analogy. This is because the constant corresponding to the angle of the pushing force does not appear in both the example specification *and* solution. In particular, the example solution states that the pushing force is drawn 147 degrees from the horizontal; in the problem solution, students are required to specify the angle *counterclockwise* from the origin in the EA-Coach interface. However, the constant 147 does not appear in the example specification, since the specification provides the pushing force angle clockwise from the horizontal (33 degrees, see e_1). Therefore, transformational analogy is not enabled, and the difference is classified as non-trivial.

Table 6-6: Superficial differences between problem p_2 & corresponding example e_4 in the static condition

Simplified fragment of solution step corresponding to the difference	Problem Solution Step	Corresponding Example Solution Step	Type of Difference
Body to apply Newton's 2 nd Law	mouse	pumpkin	Trivial
There is a pushing force on the ... due to ...	mouse/cat	pumpkin/Ann	Trivial/ Trivial
The pushing force direction is ... CCW from horizontal	30°	18°	Trivial
The pushing force magnitude is...	50 N	45 N	Trivial
There is a weight force on the ...	mouse	pumpkin	Trivial
There is a tension force on the... due to ...	mouse/string	pumpkin/cord	Trivial/ Trivial
The direction of the tension force is ...	140°	148°	Non-Trivial ¹⁵
The x-component equation for the tension force is ...	$-T \cdot \cos(40)$	$-T \cdot \cos(32)$	Trivial
The x-component equation for the pushing force is ...	$P \cdot \cos(30)$	$P \cdot \cos(18)$	Trivial

¹⁵ Note that the difference between the directions of the tension force the problem/example is non-trivial, while the difference between the directions of the pushing force is trivial. This is because the difference corresponding to the pushing force direction can be resolved by transformational analogy, while the difference corresponding to the tension force direction cannot. The pushing force difference can be resolved by substituting example constant 18 by the problem constant 30 (since the constant 18 appears in both the example solution and specification and has a corresponding constant in the problem specification, enabling transformational analogy). On the other hand, the constant corresponding to the angle of the tension force does not appear in both the example specification and solution. In particular, the example solution specifies that the tension force angle is 148 degrees counterclockwise from the horizontal. However, this constant does not appear in the example specification, because it gives the cord angle *clockwise* from the horizontal (i.e., 32 degrees, see e_4 specification). Therefore, transformational analogy is not enabled, since 148 does not appear in both the example solution and specification, and so the difference is classified as non-trivial.

Table 6-7: Superficial differences: mean number of trivially and non-trivially different problem/example steps in the two conditions

	Static selection	Adaptive selection
Mean # trivial differences (<i>st. dev.</i>)	8 (0)	8 (.81)
Mean # non-trivial differences (<i>st. dev.</i>)	1 (0)	4.75 (.95)

To help clarify the discussion of the results, we report on the level of similarity between the statically vs. adaptively selected examples and corresponding problems. Recall that as we indicated in section 6.4, in the static condition examples e_1 and e_4 were paired with problem p_1 and p_2 , respectively. In the adaptive condition, the EA-Coach never chose to make this pairing¹⁶. Consequently, as is summarized in table 6-7, the adaptively selected examples were always less superficially similar to the target problem than the statically selected ones, since (1) the total number of trivial and non-trivial differences between the problem/example was always higher in the adaptive condition than in the static condition and (2) the adaptive condition always had more non-trivial differences. The EA-Coach example mechanism chose less similar examples because it predicted that such examples had better potential to trigger learning, while helping students to achieve problem-solving success. We will see how this prediction was realized once the dependent measures have been described.

¹⁶ There are some circumstances that may lead the EA-Coach to select highly similar examples. For instance, if a student has a very low tendency for EBLC hindering her ability to learn rules, then by taking this into account the system *may* select highly similar examples that at least will help the student to generate the problem solution.

6.5 Dependent Measures

Recall that the objective of the evaluation was to analyze how the two example-selection approaches (adaptive vs. static) impacted learning and problem-solving success. To assess the impact on learning, we analyzed:

- *Effect of example selection on min-analogy and EBLC.* To determine example selection's impact on min-analogy, we analyzed *copy events*, i.e., the number of copy episodes for a problem solution. To determine example selection's impact on how students self-explained, we analyzed *Self-explanation/EBLC events*, i.e., the number of self-explanations expressed while generating a problem solution, including EBLC explanations.
- *Learning gains from pre to post test:* the percentage of rules learned in each example-selection condition.

To assess the impact on problem-solving success, we determined how many students generated a correct problem solution in each condition. We also analyzed the impact of example selection on students' performance during the problem-solving process by measuring:

- *Task time:* the time to generate a problem solution.
- *Errors:* the number of errors made while generating a problem solution.

6.6 Results

To obtain results for the dependent measures of interest, the key analysis techniques we used were (1) univariate repeated-measures ANOVA and (2) Wilcoxon signed ranks test. The former is a standard technique used for statistical analysis of data, but it involves averaging over participants [Howell, 1997]. Consequently, it has the potential to obscure how a particular treatment impacts an individual [Vicente and Torenvliet, 2000]. Thus, we also report this information, and analyze it using the Wilcoxon signed ranks test.

As far as the ANOVA analysis is concerned, only one within-subject factor may be included in the analysis at a time. Although in our experiment, the primary within-subject factor of interest was example-selection type (also referred to as *selection* below, static vs. adaptive), our study design included another within-subject factor, namely *problem* (i.e., p_1 vs. p_2). Even though we designed the problems to be equivalent in terms of difficulty, we first confirmed that this was the case by running the ANOVA with problem as the primary within-subjects factor. The ANOVA revealed that there were no significant main, interaction or order effects for problem for any of the dependent variables. Therefore, we performed the remainder of the analysis with selection as the within-subject factor. In this analysis, we considered selection in combination with the between-subject factors resulting from the counterbalancing of selection and problem types, referred to as *selection order* and *problem order*¹⁷, respectively. Therefore, the ANOVA used was a 2 (*example-selection type*) x 2 (*selection order*) x 2 (*problem order*), where selection order and problem order were control variables.

We first verified that there was no significant difference in subjects' performance on pre-test between the four groups arising from the counterbalancing of problem/condition combinations, which was the case ($F(3,15)=0.23$, $p=0.874$). Of the sixteen subjects that completed all stages of the experiment, two accessed an example in only one of the two selection conditions, but did not even open the example in the other condition. Specifically, one subject did not access the adaptively selected example, but did access the statically selected one; a second subject did not access the statically selected example, but did access the adaptively selected one. We originally intended to include these subjects' data in the within-subjects statistical analysis. After further consideration,

¹⁷ We kept problem order in this analysis because although we did not find it to have an impact in isolation of selection (i.e., when the analysis included problem as the within-subject factor) we wanted to account for the possibility that it did play a role when the analysis was performed with selection as the within-subject factor. It turned out that this was not the case: we did not find any significant problem order interactions or effects.

however, we decided not to, since it is hard to argue that selection has an impact on APS behaviors if the student does not open the available example. Thus, all of the within-subjects statistical analysis is based on the data from the fourteen subjects who used examples in both the static and adaptive selection conditions. We did perform some analysis that did not involve within-subjects comparisons, for which we considered data from all 16 participants – when this is the case, it is clearly indicated in the text.

6.6.1 Results: Learning

We first report on how example selection impacted APS behaviors that impact learning, namely min-analogy and EBLC. We then provide findings from the analysis on the pre/post test differences.

6.6.1.1 APS Behaviors: Min-Analogy

As we indicated in section 6.5, to analyze example selection's impact on min-analogy, we analyzed students' copy events. To identify copy events, we used the same method as in the pilot study, which we described in section 3.1.3 and summarize here. This involved identifying instances when students (1) accessed a step in the example solution and (2) generated the corresponding step in their problem solution. If the student's entry to a step accessed in the example was identical or included minor differences of the type listed in section 2.2.1, then it was classified as 'copied'. Note that this analysis included both correct and incorrect copy events.

The ANOVA results for copy events are shown in table 6-8. There was a significant main effect of example selection on copying ($F(1,10) = 7.978, p=0.018$). On average, students copied less from the adaptively than from the statically selected examples (on average, 5.9 vs. 8.1, respectively)¹⁸. This was true for 10 of the 14 participants involved in the within subject analysis, while 2 participants copied more in the adaptive condition and 2

¹⁸ No other effects or interactions were significant.

Table 6-8: Summary of ANOVA analysis for copy events (N=14)

Source	SS	Df	MS	F	Sig.
Selection	34.074	1	34.074	7.978	.018
Selection * Problem _{order}	3.646	1	3.646	0.854	.377
Selection * Selection _{order}	0.074	1	0.074	0.017	.898
Selection * Problem _{order} * Selection _{order}	2.503	1	2.503	0.586	.462
Error	42.708	10	4.271		

participants had an equal number of copy events in both conditions. These individual differences between the two conditions are statistically significant according to the Wilcoxon signed ranks test ($p=0.011$), confirming the ANOVA results that students copied significantly less in the adaptive condition.

6.6.1.2 APS Behaviors: Self-explanation/EBLC Events

To identify EBLC, we analyzed the verbal protocols. EBLC is a form of self-explanation that involves learning a rule by using overly general or common sense reasoning, as opposed to explaining a solution step using a domain rule that is already known. We refer to the latter as '*domain-based*' explanations. Although identifying self-explanation is fairly straightforward, further distinguishing between EBLC and domain-based explanation is more challenging. This is because students' explanations are often fragmented or incomplete (other attempts to classify EBLC faced similar issues, e.g., [VanLehn, 1999]). To make this more concrete, figure 6-3 shows some examples of self-explanations expressed during the study, including a discussion of why it is hard to identify the type of explanation (the classification for the explanations in figure 6-3 will be provided shortly). Given the challenge of identifying EBLC, we took a two-tier approach to analyze example selection's impact on self-explanation. We first classified utterances as self-explanation, without trying to distinguish between domain and EBLC explanations. As mentioned above, this is an easier classification, and it already provides

Figure 6-3 : Sample self-explanations

<p>[1] <i>"a_x is zero - not moving at this point..."</i></p> <p>Explanation expressed while reading the example solution step "$a_x = 0$" for example e_4. Not enough information is provided to determine how this inference is made; no indication of EBLC-style reasoning is provided.</p>
<p>[2] <i>"... it's zero because it is on the y axis (points to the example free body diagram) but this (points to her problem) can't be because...it's at an angle"</i></p> <p>Explanation expressed while looking at free body diagram in example e_5 after reading example solution step "$P_x=0$". The subject correctly infers why the x component of the pushing force is zero (i.e., because its component is perpendicular to the x-axis) but it is not clear what reasoning she uses to make this inference.</p>
<p>[3] <i>"I think force W will be force N because both are pointing down – will that be right?"</i></p> <p>Explanation expressed while solving problem p_1. Subject then writes incorrect equation "$F_n = F_w$", where F_n and F_w are the variables representing the magnitude of the normal and weight force, respectively, which the subject has drawn correctly in her free-body diagram. Since both the forces related to the explanation are drawn straight down, the explanation appears to relate to a overly general rule that the magnitude of two forces is equal if they point in the same direction.</p>
<p>[4] <i>"should be negative – it is in the negative quadrant (gestures towards her free body)"</i></p> <p>Explanation expressed while solving problem p_1. No indication of EBLC provided.</p>
<p>[5] <i>"... It's negative because it's under the x axis (referring to the tension vector while looking at free body diagram of example e5) my tension is above the x axis so it should be positive and it will be cos... the angle it makes with it will be 40"</i></p> <p>Explanation expressed after reading the example solution step "$T_x = -T\cos(5)$" in example e_4. Subject appears to be relying on an overly general mathematics rule to infer that components for all vectors below the x-axis are negative.</p>
<p>[6] <i>"It is 21 degrees to the horizontal and the picture is going that way (gestures left) so maybe it is the other way around... so it would be 180 – 21"</i></p> <p>Explanation expressed after an incorrect attempt to draw the pushing vector while solving problem p_1. Specifically, subject tried several times to enter the direction of the pushing force as 21 degrees, which is the value given in the problem description. Subject appears to be making a link between the direction of motion and the direction of the pushing force, possibly relying on common sense reasoning that when a push is applied to an object causing the object to move in a particular direction then the pushing force is also oriented in the direction of motion.</p>
<p>[7] <i>"py equals zero... why are you zero ... oohh it is because it is zero degrees"</i></p> <p>First portion of utterance expressed after reading example solution step "$P_y=0$" in e_3 ; second portion expressed while looking at the example's free body diagram, which shows that the pushing force is parallel to the x-axis. The subject's inference is not clear: a literal translation is that she infers the component is zero because the force direction is zero degrees from the x-axis, which is slanted with respect to the horizontal; an alternative is that she infers that the component is zero because the force is oriented zero degrees to the horizontal.</p>

Table 6-9: Summary of ANOVA analysis for self-explanation events (N=14)

Source	SS	Df	MS	F	Sig.
Selection	23.574	1	23.574	6.422	0.030
Selection * Problem _{order}	.003	1	.003	.001	.978
Selection * Selection _{order}	1.574	1	1.574	.429	.527
Selection * Problem _{order} * Selection _{order}	2.860	1	2.860	.779	.398
Error	36.708	10	3.671		

some insight into how example selection impacts how students reason during APS. We then classified self-explanations as EBLC or domain based, as we describe below.

To identify self-explanation, the verbal protocols were analyzed by the author following the same method as was used during the pilot study (see section 3.1.3). The ANOVA results are shown in table 6-9. There was a significant main effect of example selection on self-explanation ($F(1, 10) = 6.42, p = 0.03$): students expressed significantly more self-explanations in the adaptive condition than in the static condition (on average, 4.07 vs. 2.57, respectively) ¹⁹. This was true for 9 of the 14 participants involved in the within-subject analysis, while 3 students self-explained more in the static condition, and 2 students generated the same number of self-explanations in both conditions. These individual differences are statistically significant according to the Wilcoxon signed ranks test ($p = 0.023$), confirming the ANOVA results that students self-explained significantly more in the adaptive condition.

To get a sense of when self-explanation took place, we identified explanations that occurred during (1) example studying, when students explained from an example vs. (2) problem solving, when students explained their own solution entries without referring to the example. Figure 6-3 shows examples of explanations generated in each context:

¹⁹ No other effects or interactions were significant.

explanations [1], [2], [5] and [7] were expressed during example studying; explanations [3], [4] and [6] were expressed during problem solving. In the adaptive condition, about half of the self-explanations generated occurred in each context (on average per student, 53% during example studying vs. 48% during problem solving). In the static condition, a higher percentage of self-explanations was generated during example studying (on average per student, 79% during example studying vs. 21% during problem solving). We discuss the implications of this finding in terms of the EA-Coach in section 6.6.1.4.

To identify EBLC-based self-explanations, we followed the approach in [VanLehn, 1999] and looked for instances where either (1) students appeared to rely on common sense or overly general reasoning as indicated via the verbal protocols or (2) when students managed to generate explanations for domain principles that they did not have good domain understanding of (as indicated by pre-test and/or confusion expressed during the study), suggesting that knowledge gaps existed which required EBLC style reasoning to generate the explanation. In figure 6-3, explanations [2-3] and [5-7] were classified as EBLC²⁰, while the remaining two explanations, namely [1] and [4], were classified as domain-based explanations²¹.

The ANOVA results for EBLC events are shown in table 6-10. There was a significant main effect of example selection on EBLC ($F(1,10) = 12.8, p = 0.005$). On average, students generated significantly more EBLC-based explanations in the adaptive condition than in the static condition (2.92 vs. 1.14, respectively). This was true for 9 of the 14 students involved in the within-subject analysis, while 5 students generated the same

²⁰ For explanation [6], the classification is based on the subject's confusion regarding the corresponding domain principle; for explanation [3], the classification is based on the fact that the subject appears to be using overly general reasoning; for the remaining explanations, the classification is based on the fact that the explanations correspond to principles subjects did not have pre-existing domain knowledge, as indicated by pre-test data.

²¹ The classification is based on the fact that these two explanations did not appear to involve common sense or general reasoning and related to principles for which subjects answered corresponding pre-test questions correctly.

Table 6-10 : Summary of ANOVA analysis for EBLC events (N=14)

Source	SS	Df	MS	F	Sig.
Selection	22.012	1	22.012	6.422	0.005
Selection * Problem _{order}	2.679	1	2.679	4.560	.240
Selection * Selection _{order}	.012	1	.012	.007	.935
Selection * Problem _{order} * Selection _{order}	.107	1	.107	.062	.808
Error	17.167	10	1.717		

number of EBLC explanations in both conditions. These individual differences are statistically significant according to the Wilcoxon signed ranks test ($p=0.007$), confirming the ANOVA results. In contrast, there was no difference in the number of domain-based self-explanations between the two conditions (on average, 1.143 in the adaptive condition vs. 1.12 in the static condition, $F(1, 10) = 0.28$, $p = 0.87$).

6.6.1.3 Pre/Post Test Gains

With the analysis presented above, we evaluated how example selection influences learning by analyzing how it triggers APS behaviors that are known to foster it. Another way to measure learning is via pre/post test differences: if a student shows gains on questions corresponding to a given rule from pre to post test, then this provides evidence that the student has solidified knowledge of that rule. We found that in general, students significantly improved from pre to post test (2-tailed $t(13)=7.49$, $p<0.001$; table 6-11 provides information on subjects' pre and post test performance). Given the within-subject design, however, to analyze how each selection condition impacted learning it is necessary to consider rules that only appeared in one of the two conditions. This is because if a rule was involved in both conditions, it is impossible to unambiguously determine the condition that triggered learning (if any).

Table 6-11: Subjects' pre and post test performance (N=14)

	Pre-Test	Post Test
Mean (/34)	20.6	29.3
Standard Deviation	8.2	6.8

To gain insight into student learning based on individual rules, we first identified for each student the set of rules that met the following criteria:

- Were involved in only *one* of the two conditions and the student used the example in that condition
- Were not known or partially known as indicated by the pre-test (since the test included several questions per rule, a student could have 'partial' knowledge by answering some but not all questions correctly)

We refer to the rules satisfying these conditions as '*learning opportunities*'. Over all 16 students, six rules²² met these criteria and so were learning opportunities. We then identified, for each student and each rule under consideration, the condition in which the student experienced the rule. Recall that because we counterbalanced both the problems and the selection conditions, some subjects experienced a given rule in the adaptive condition, while other subjects experienced this rule in the static condition. If we consider all 16 subjects, this left us with 11 students who had learning opportunities in the static condition, and 11 who had learning opportunities in the adaptive condition (see table 6-12, which also shows the average number of learning opportunities per student in each condition). Because many subjects had learning opportunities in only *one* condition, we chose to conduct a between subjects analysis with all 11 subjects in each condition.

²² The fact that the number of rules considered as learning opportunities is small is a function of the low number of structural differences between the two problems.

Table 6-12: Pre/Post test gains

	Static Selection	Adaptive Selection
Number of students who had learning opportunities (over all 16 students)	11	11
Average number of learning opportunities per student	2	2.18
Percentage of rules learned on average per student	52%	77%

To identify '*learning events*' (i.e., rules learned) given a students' learning opportunities, we analyzed, for each student, the number of rules that the student showed gains²³ on from pre to post-test corresponding questions. Table 6-12 shows the average percentage of rules learned by each student in the two selection conditions. For a given student, this percentage is calculated by dividing that student's '*learning events*' by her '*learning opportunities*'.

As the data in table 6-12 indicates, we found a trend that the percentage of learning events was higher for rules students experienced in the adaptive condition than rules experienced in the static condition ($t(20)=1.72$, $p=0.056$).

6.6.1.4 Learning: Summary of Key Results and Discussion

We found that the EA-Coach's adaptively selected examples encouraged students to engage in the effective APS behaviors better than the statically selected examples. Specifically, in the adaptive condition, students copied significantly less and generated significantly more EBLC-based explanations. We also found encouraging trends that students learned more rules in the adaptive condition than in the static condition.

²³ A gain occurs if, for a given rule, a student performs better on corresponding questions on the post-test than she did for the pre-test.

The results show that in general, the EA-Coach meets its learning goal, thus supporting our assumption that certain superficial differences encourage effective APS behaviors. The statically selected examples were highly similar to the target problem and thus made it possible to generate the problem solution by copying, of which students took advantage. As one subject stated during an informal discussion after the study, *"1st example [statically selected] is most useful, more straightforward ...you don't have to think that much because it is same thing - step by step"*. Conversely, the adaptively selected examples were less superficially similar to the target problem. This provided incentive for students to generate the problem solution without copying, i.e., to engage in min-analogy, when they realized that copying was not an effective strategy. For some students this realization happened after copy attempts resulted in errors, which discouraged subsequent copying. One subject articulated this by saying *"I should just ignore this and solve the problem on my own"*, after several incorrect copy attempts. To see if there was a difference in terms of how successful students were in copying in the two conditions, we labeled each copy event as correct or incorrect. In the static condition, 4.8% (20/416) of the total errors were the result of incorrectly copied steps, compared to 6.7% (28/416) in the adaptive condition (table 6-17 in section 6.6.2.3 summarizes the source of all errors, which are discussed in that section). On average per student, 66% of copying was correct in the adaptive condition, vs. 86% in the static condition. All of the correct copy events in both conditions corresponded to example solution steps that had trivial superficial differences or no differences with the problem.

Now let's discuss why some steps were incorrectly copied. In the static condition, all of the copy errors corresponded to example solution steps that either had no differences or only trivial differences with the problem, and were the result of typos or poor attempts at substituting examples constants via ones needed for the problem's solution. Students tended to easily fix these errors when the EA-Coach signaled them, making it unlikely to discourage copying. In the adaptive condition, on average of the 11 students who had copy errors, 86% of the errors corresponded to students attempting to copy example solution steps that had non-trivial differences with the problem, and so could not easily be resolved by fixing slips. There were 23 such errors out of a total of 28 copy errors; all

but 4 were resolved (as will be described in section 6.6.2.1, two students did not generate a full problem solution). The remaining copy errors in the adaptive condition (5 out of a total of 28 copy errors) were the result of students copying example solution steps that either had no differences or only trivial differences with the problem (and so could easily be resolved).

Thus, for the most part students managed to resolve errors resulting from incorrect copying in both the selection conditions. However, what differentiates the two conditions is how easily the errors could be resolved. In the static condition, all the copy errors could easily be resolved because they corresponded to either trivial differences or slips. In contrast, in the adaptive condition, the majority of copy errors could *not* easily be resolved by fixing slips because they corresponded to non-trivial differences. This, in conjunction with the finding that students were more successful at copying correctly in the static condition, suggests that the type of superficial similarity was responsible for facilitating/ discouraging copying in the static and adaptive selection conditions, respectively.

Although adaptively selected examples effectively discouraged copying, we should point out that we saw some instances where subjects appeared to switch from one bad behavior to another. In particular, sometimes when students were not successful at copying a step from the corresponding example, they would switch to rapidly entering what appeared to be alternative guesses to generate the problem step (a behavior that may be referred to as '*gaming the system*' [Baker, Corbett et al., 2006a]). Since the system provided feedback for correctness, students could resort to gaming instead of deriving the step by learning the corresponding rule. For instance, one student produced forty attempts to generate a problem step, after unsuccessfully trying to copy it from the example (the example and problem shared a non-trivial difference at this point that blocked copying). This student, who generated very few self-explanations, continued to produce subsequent solution attempts quite quickly, leaving little or no time for reasoning of any kind. Gaming behavior was sometimes apparent in the protocols – for instance, another student said "*I'm just trying things, I don't feel like thinking about it*".

Table 6-13: Results of correlation analysis (learning; N=14²⁴)

		Copy Events	Errors
EBLC Events	Pearson Correlation	-.207	.468
	Significance (1 tailed)	.145	0.006

In addition to influencing copying, example selection influenced *when* and *how* much students self-explained. As far as when self-explanations occurred, recall that in the adaptive condition, about half of the self-explanations occurred during problem solving and about half during example studying. In contrast, in the static condition students generated more self-explanations during example studying than during problem solving. One possible explanation for this finding is that because students were copying less in the adaptive condition, they were more proactive during the problem-solving process. This resulted in more self-explanations *during* problem solving. Although students could have referred to the example to generate the self-explanations, the data suggests that they tried to avoid switching contexts (from the problem solution that they were generating over to the example solution). Recall that currently the EA-Coach student model only represents self-explanation from an example. The fact that students also explained during problem solving without referring to an example suggests that the model could take this into account (discussed in chapter 8).

Now let's discuss why students generated more EBLC-style self-explanations in the adaptive condition. According to one of the assumptions embedded in the EA-Coach's example-selection mechanism, blocking copying when a student is lacking knowledge of a principle relevant for generating a problem step encourages the student to learn via

²⁴ For the correlation analysis, we used data from 14 students who used examples in both conditions (i.e., N=14) but we collapsed across conditions: we used data from both conditions in the analysis. Therefore, the number of data points involved in the analysis was actually 28, i.e., two per subject.

EBLC. Since students copied less in the adaptive condition, this may explain the higher number of EBLC explanations in that condition. To see if this was indeed the case, we checked whether a correlation existed between EBLC and copying. For this and subsequent correlation analysis, since we were interested *in general* about whether a given relationship existed, we collapsed across conditions (i.e., used data from both conditions in the analysis).

The results of the correlation analysis are summarized in table 6-13. We found a trend indicating that less copying was associated with more EBLC, although this correlation did not reach significance (pearson correlation = $-.207$, 1 tailed $p=.145$). Another possibility for why students had more EBLC explanations in the adaptive condition is related to the 'error' dependent variable. In particular, errors may have triggered EBLC reasoning to fill the gap revealed by the error, as is proposed by some cognitive theories of learning [VanLehn, 1996]. As will be discussed in section 6.6.2.2, students made significantly more errors in the adaptive condition than in the static condition. Consequently, the adaptive condition provided more triggers for EBLC to resolve the errors. We found that errors were indeed associated with EBLC (pearson correlation = $.468$, 1 tailed $p=0.006$).

Instances when the learning goal was not satisfied. Although in general the adaptively selected examples encouraged our target APS behaviors and thus satisfied the learning goal, this was not always the case. We begin by discussing when this happened with respect to the copy findings. As reported in section 0, two students copied more in the adaptive condition than the static condition. One of these students had an above average number of copy events in both conditions (10 vs. 12 copy events in the adaptive and static conditions, respectively, compared to the average of 5.9 vs. 8.1, respectively), suggesting that she had a max-analogy tendency. Thus, it appears that more explicit scaffolding than the EA-Coach's may be needed to encourage a shift over to min-analogy for students with a strong max-analogy tendency. The second student had an average number of copy episodes in the adaptive condition, and about half of the average number of copy episodes in the static condition (6 vs. 4 copy events in the adaptive and static conditions, respectively). One explanation as to why this student copied more in the

adaptive condition may be related to the order in which this student experienced the conditions. Specifically, the student solved the problem in the adaptive condition first. Although we did not find overall that selection order had an effect, for this student the adaptively selected example may have discouraged copying, but this did not become apparent until the subsequent (static) condition. Two other students had the same number of copy events in the adaptive and static conditions; both number of copy events were below average (one student only copied once in each condition, while the other copied four times in each condition). This suggests that they already had a tendency for min-analogy, and so adaptively selected examples did not influence how much they copied.

As far as the EBLC events are concerned, although none of the students expressed fewer explanations in the adaptive condition than in the static condition, five students generated an equal number of explanations in the two conditions. Three of these students expressed a below-average number of explanations, and so appeared to have a low EBLC tendency (two students generated only one explanation in each condition, while one student generated no explanations, compared to the average 2.92 vs. 1.14 in the adaptive and static conditions, respectively). This suggests that additional scaffolding may be beneficial to encourage EBLC for students who have a particularly low tendency for EBLC (discussed in chapter 8). As far as the other two students are concerned, who generated two EBLC explanations in each condition, it is not clear why the adaptively selected examples did not encourage them to self-explain more than the statically selected examples. In addition to the cases discussed above, two other students in the adaptive condition were not able to learn the rules that were required to generate the problem solution, which hindered their problem-solving success; we discuss these two cases in section 6.6.2.3.

6.6.2 Results: Problem-Solving Performance

We first report on how example selection influenced students' success in terms of the final problem solution. We then provide findings on selection's impact on performance during the problem-solving process.

6.6.2.1 Problem-Solving Success

Problem-solving success is achieved if students generate a correct problem solution. For the 14 students considered in the within-subjects analysis, problem-solving success was realized as follows:

- In the static condition, all 14 subjects generated a correct problem solution.
- In the adaptive condition, 12 subjects generated a correct problem solution. The other two subjects produced a partial problem solution (both first solved problem p_1 in the adaptive condition and both referred to the available example in both conditions).

This difference between the two conditions, however, is not statistically significant (sign test, $p=0.5$)²⁵, indicating that overall, both statically and adaptively selected examples helped students generate the problem solution.²⁶

We should point out that feedback for correctness delivered by the EA-Coach was likely a factor in helping students achieve a fully correct solution. This feedback led students to either try and fix errors or erase them, and consequently *none* of the students' final solutions contained errors (two students had incomplete solutions, but even these did not contain errors).

6.6.2.2 Task Time & Errors

To analyze how example selection affected the problem-solving process, we analyzed students' task time and errors in each condition. The ANOVA results for the task time dependent variable are shown in table 6-14. There was a significant main effect of example selection on task time ($F(1, 10) = 31.59, p<0.001$): students took significantly

²⁵ The sign test was used instead of the Wilcoxon test because the data does not include magnitude information, making the Wilcoxon test inappropriate.

²⁶ As far as the two students who used an example in only one condition are concerned, both generated a correct problem solution in the static and the adaptive condition.

longer to generate the problem solution in the adaptive condition than in the static condition (on average, 42min, 23sec vs. 25min, 35sec, respectively). This was true for 13 of the 14 students involved in the within subject analysis (these individual differences between the two conditions are statistically significant according to the Wilcoxon signed ranks test, $p=0.004$).

Table 6-14: Summary of ANOVA analysis for task time (N=14)

Source	SS	Df	MS	F	Sig.
Selection	2250.25	1	2250.245	31.598	<.001
Selection * Problem _{order}	70.245	1	70.245	0.986	.344
Selection * Selection _{order}	576.164	1	576.164	8.090	.017
Selection * Problem _{order} *Selection _{order}	28.94	1	28.94	.406	.538
Error	712.19	10	71.22		

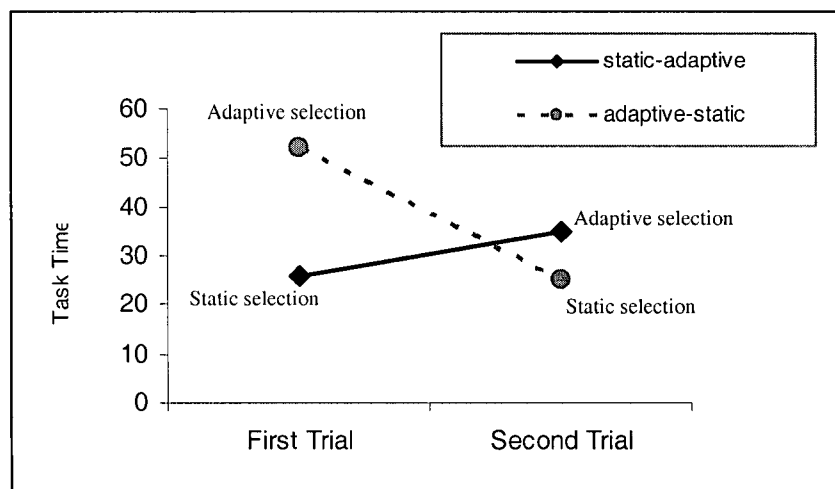


Figure 6-4: Interaction between selection and selection order for task time

In addition, there was a significant interaction between example selection and selection order, indicating that the presentation order of the conditions disproportionately affected task time ($F(1,10)=8.09$, $p=0.017$)²⁷. As the graph of this interaction in figure 6-4 shows, the magnitude of the difference in task time between the two conditions was greater if the adaptive condition was first in the selection order than if it followed the static condition. This suggests that students took longer to solve the problem in the adaptive condition if they saw it first instead of second. To gain more insight into this result, we analyzed the difference in task time between the two conditions for each selection order, *adaptive-static* and *static-adaptive*. Using a Bonferroni adjustment [Howell, 1997] to re-set the significance threshold to $p=0.05/2 = 0.025$, this analysis revealed a significant simple effect of example selection on task time in the *adaptive-static* selection order ($t(5)$ ²⁸=4.99, $p=0.004$). Students in this selection order took longer to generate the problem solution in the adaptive condition than in the static condition (on average, 51min, 55sec vs. 24min, 38sec, respectively). Although task time was also higher for the adaptive condition in the *static-adaptive* selection order (on average, 35min, 15sec vs. 26min, 18sec for the static condition), this simple effect ($t(7)$ ²⁹=2.52, $p=0.04$) was only marginally significant after the Bonferroni adjustment. Thus, although task time was higher in the adaptive condition, this effect was strongest for students who experienced the adaptive condition first in the selection order.

For the error analysis, we counted all errors a student made while generating a problem solution in each selection condition; the ANOVA results are shown in table 6-15. There was a significant main effect of example selection on error ($F(1, 10) = 11.53$, $p=0.007$): students produced significantly more errors while generating the problem solution in the

²⁷ No other effects or interactions were significant.

²⁸ Of the 14 students whose data we used in the within-subjects analysis, 6 experienced the adaptive-static selection order.

²⁹ Of the 14 students whose data we used in the within-subjects analysis, 8 experienced the static-adaptive selection order.

adaptive condition than in the static condition (on average, 22.35 vs. 7.57, respectively). This was true for 11 of the 14 students involved in the within subject analysis (these individual differences between the two conditions are statistically significant according to the Wilcoxon signed ranks test, $p=0.017$).

Table 6-15: Summary of the ANOVA analysis for error (N=14)

Source	SS	Df	MS	F	Sig.
Selection	1876.298	1	1876.298	11.532	.007
Selection * Problem _{order}	37.333	1	37.333	.229	.642
Selection * Selection _{order}	1122.012	1	1122.012	6.896	.025
Selection * Problem _{order} * Selection _{order}	76.190	1	76.190	.468	.509
Error	1627.083	10	162.708		

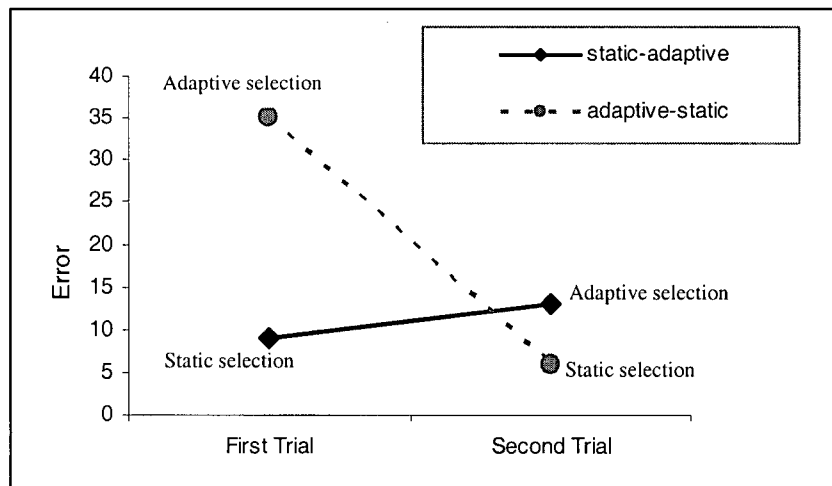


Figure 6-5: Interaction between selection and selection order for error

In addition, there was a significant interaction between selection and selection order, indicating that the presentation order of the conditions disproportionately affected error ($F(1,10)=6.89$, $p=0.025$)³⁰. As was the case for the time dependent variable, the magnitude of the difference in number of errors between the two conditions was greater if the adaptive condition was first in the selection order than if it followed the static condition (the graph of this interaction is shown in figure 6-5). We then analyzed the difference in number of errors between the two conditions for each selection order, *adaptive-static* and *static-adaptive*. This analysis revealed a marginally significant simple effect of selection on error in the *adaptive-static* selection order after the Bonferroni adjustment re-set the significance threshold to $p=0.05/2 = 0.025$ ($t(5)$ ³¹ $=3.07$, $p=0.028$). This effect indicates that students in the adaptive-static selection order generated more errors in the adaptive condition than in the static condition (on average, 35 vs. 5.67, respectively). Although on average, students in the *static-adaptive* selection order also generated more errors in the adaptive condition than in the static condition, this was not significant (12.63 in the adaptive condition vs. 8.88 in the static condition, $t(7)$ ³² $=1.04$, $p=.331$). As was the case with task time results, these findings demonstrate that although in general the number of errors was higher in the adaptive condition, this effect was strongest for students who experienced the adaptive condition first in the selection order.

6.6.2.3 Problem-Solving Performance: Summary of Key Results and Discussion

As stated above, problem-solving success is achieved if the student generates the problem solution, and is not a function of performance (task time, number of errors) *while* doing so. We found that in general students were successful in generating the problem solution in both the static and adaptive selection conditions.

³⁰ No other effects or interactions were significant.

³¹ Of the 14 students whose data we used in the within-subjects analysis, 6 experienced the adaptive-static selection order.

³² Of the 14 students whose data we used in the within-subjects analysis, 8 experienced the static-adaptive selection order.

Instances when the problem-solving success goal was not satisfied. Two students generated a correct but incomplete solution in the adaptive condition. Both of these students received an example with non-trivial superficial differences that blocked copying of some of its solution steps, because the EA-Coach student model predicted that this would trigger learning via EBLC. This prediction is mediated by the model's assessment of the student's EBLC tendency, to which we had assigned a generic prior probability of 0.5 for both students due to lack of more accurate information. This appeared to have been inaccurate for one of these students, who showed no desire to engage in any in-depth reasoning during the study and so likely had a very low EBLC tendency. The other student, however, generated a number of EBLC self-explanations, indicating that inaccurate prior on EBLC tendency was not the reason for suboptimal example selection in terms of problem-solving success. This student invested considerable effort and did learn *some* of the rules needed to solve the problem (as we found by comparing her pre and post-test answers on related questions). However, although the student model simulation predicted she would learn all the necessary rules and thus generate the full problem solution, she was unable to do so within the allotted 60 minutes. We can't predict whether this student would have eventually generated a full solution if more time was available or whether she would have become overwhelmed and frustrated by the process. There is a fine line between taking extra time to learn from one's errors, and entering what is referred to as *floundering*, i.e., engaging in too many trial and error attempts that obstruct learning. Thus, even if students have good APS tendencies there is no guarantee that they will learn all the rules needed to generate a full problem solution. This suggests that the system could be improved by the addition of more explicit scaffolding to help students learn rules via EBLC when they are floundering, as we discuss further in chapter 8.

Task time & errors. Students took longer/made more errors in the adaptive condition than in the static condition. This was particularly the case in the *adaptive-static* selection order (i.e., task time and number of errors were lower if the adaptive condition followed the static condition). The most plausible explanation for this finding is that solving the problem in the static condition first provided a scaffold that better prepared students to

Table 6-16: Results of correlation analysis (problem-solving performance; N=14³³)

		Copy Events	EBLC Events
Task Time	Pearson Correlation	-.362	.508
	Significance (1 tailed)	0.028	0.003
Errors	Pearson Correlation	-.271	.468
	Significance (1 tailed)	.082	.006

solve the problem in the adaptive condition. As one of the subjects in the *static-adaptive* order indicated, *“I liked the transition between the two... one was very similar, exactly the same [i.e., statically selected example] – this one [i.e., adaptively selected example] you had to manipulate the set up... It’s not hard to manipulate but you’d have to think about the question and how to relate them - the first one showed you the basics”*.

In general, however, the fact that students had a higher task time and more errors in the adaptive condition is not a negative result from a pedagogical standpoint, because these are by-products of learning. Specifically, cognitive science research shows that learning takes time and may require multiple attempts before the relevant pieces of knowledge are inferred/correctly applied (e.g., [VanLehn, 1990; Chi, 2000]). In particular, it is reasonable to assume that not copying and trying to reason via EBLC can take longer and result in more errors before a solution is found. To gain more insight into the relationships between APS behaviors and problem-solving performance during APS in our study, we checked for correlations between the corresponding variables (time/errors

³³ For the correlation analysis, we used data from 14 students who used examples in both conditions (i.e., N=14) but we collapsed across conditions: we used data from both conditions in the analysis. Therefore, the number of data points was actually 28, i.e., two per subject

Table 6-17: Source of errors (N=14)

	Static Selection	Adaptive Selection	Total
% of errors from incorrectly copied steps	4.8% (20/416)	6.7% (28/416)	11.5% (48/416)
% of errors from problem-solving without copying	20.4% (85/416)	68.1% (283/416)	88.5% (368/416)
Total	25.2% (105/416)	74.8% (311/416)	100% (416/416)

and copying/EBLC)³⁴. As table 6-16 shows, as far as errors are concerned, this analysis suggests that

- generating a problem solution without copying is associated with more errors during the problem-solving process;
- errors are associated with increased EBLC – this result was discussed in section 6.6.1.4.

Since generating a solution without copying is associated with more errors, the analysis also provides insight as to why students had more errors in the adaptive condition than in the static condition: because they generated more of the problem solution on their own, without copying. Table 6-17 summarizes the overall percentage of errors that came from copying vs. problem solving without copying in the two conditions.

As far as task time is concerned, the correlation analysis suggests that

- copying is associated with decreased task time;
- EBLC is associated with increased task time.

³⁴ As was the case for the correlation-related analysis in section 6.6.1.4, the analysis included data from both conditions.

6.7 Summary

In this chapter, we described a controlled evaluation of the EA-Coach. The evaluation focused on the key form of support delivered by the framework, its adaptive example-selection mechanism. To verify how well the EA-Coach example-selection process met its learning and problem-solving success goals, we compared it with a selection strategy that involves choosing the maximally similar example, which is the approach is advocated by existing ITS that perform example selection.

The evaluation of the EA-Coach demonstrated that its example-selection mechanism meets the two selection goals. As far as learning is concerned, the EA-Coach's example-selection mechanism is more effective at encouraging EBLC and min-analogy than statically selected examples that are maximally similar to the target problem. Specifically, on average, students generated significantly more EBLC-based self-explanations and copied significantly less when presented with the EA-Coach's examples, as compared to statically selected examples. Since cognitive science research shows that copying is detrimental to learning and EBLC fosters learning, these findings provide support regarding the EA-Coach's pedagogical effectiveness. In addition to APS behaviors, we also analyzed learning gains from pre to post test. Although this analysis was limited by our within-subject design, we did find encouraging trends that students had higher pre to post test gains on rules they experienced in the adaptive condition.

As far as problem-solving success is concerned, we showed that overall, students were successful in generating the problem solution in the presence of statically *and* adaptively selected examples. Since the adaptively selected examples were less similar to the corresponding target problem than the statically selected examples, they required students to rely more on their own reasoning during APS. This resulted in more errors and higher task time during the problem-solving process. While this prevented two students from generating a problem solution, we have evidence that one of these students did learn from the process. In general, the fact that most students were able to achieve problem-solving success in the adaptive condition given the minimal scaffolding offered by the EA-Coach is a very positive result, because it shows that minor additions to the EA-Coach

scaffolding may be sufficient to enable problem-solving success for all students, in addition to learning.

Chapter 7

Related Work

In the past decade, substantial advances have been made in improving students' learning outcomes by incorporating ITS into classroom curricula [Koedinger, Anderson et al., 1997; Morgan and Ritter, 2002], demonstrating that ITS research can make and has made a practical impact on education. Here, we review a representative sample of related work from the ITS community, starting with ITS that support APS. An alternative instructional activity related to APS is problem solving without examples, i.e., pure problem solving. Once we describe ITS that support pure problem solving, we present ITS that provide support for meta-cognitive skills. Finally, we describe ITS that rely on a decision-theoretic approach for action selection.

7.1 ITS Supporting APS

Since cognitive science research shows that students have difficulty choosing appropriate examples, ITS that support APS all perform example selection for students. However, none of these tutors reason about how the selected example will impact a given student's

learning/problem-solving outcomes from APS or take the approach of assuming some problem/example differences are actually beneficial to learning while enabling problem-solving success when selecting an example.

We begin by describing ITS that support APS in domains where there isn't a clear-cut notion of correct and incorrect behavior, also referred to as *ill-structured* domains [Lynch, Ashley et al., 2006]. One such ITS is CATO [Aleven and Ashley, 1997], which helps students build legal arguments by dynamically generating examples of how an expert would argue about a particular legal case. In the law domain, a legal case typically has some features favoring the plaintiff and some the defendant. The key to successful argumentation is the ability to compare and contrast the 'current' case with precedent ones, either downplaying or emphasizing the connection between them.

In CATO, a legal case is represented by a set of *factors*, which are facts that make a case stronger or weaker for the plaintiff or the defendant. The factors, along with related abstract legal knowledge, are stored in CATO's Factor Hierarchy. Students use the hierarchy to build legal arguments and can ask CATO to provide argumentation examples. To generate an example of a legal argument, CATO first finds precedent cases that have factors in common with the current case. CATO then generates the example by emphasizing or downplaying the connection between the case and precedent cases. The example corresponds to an 'expert solution' of how a lawyer would generate an argument. Since CATO does not attempt to model students' progress or understanding as they interact with the system, the examples it generates are not tailored to individual learners.

Another ITS that provides support in an ill-structured domain is SPIEL [Burke and Kass, 1995]. SPIEL is embedded in a simulation environment that students use to practice skills needed to be an effective salesperson. SPIEL helps students acquire these skills by monitoring their interaction with the system and presenting 'stories' when a story-telling opportunity presents itself. The stories are video clips that illustrate first-person narratives about actual experiences. The relevance of a story for SPIEL is not based on a notion of correct or incorrect way to perform a task, but rather whether the story presents an

interesting story-telling opportunity. To recognize when to present a story, SPIEL relies on a set of rules. The rules are automatically created by SPIEL before run-time from the stories in SPIEL's story base and general story-telling strategies embodying how a story could be interesting. Like CATO, SPIEL does not try to model a students' social skills as they interact with the system and so does not adapt its support to individual learners.

In contrast to the ITS described in this section thus far, some ITS support APS via example selection in *well-structured domains*, where there is a clear notion of correct and incorrect behavior. AMBRE [Guin-Duclosson, Jean-Duabias et al., 2002; Nogry, Jean-Daubias et al., 2004] aims to help students generate solutions to algebra word problems by structuring the interaction according to three phases. First, students reformulate the problem in terms of its structural features (*reformulation* phase). Second, students select an example that is "nearest to the target problem" (*selection* phase). Third, students rely on this example to generate the problem solution (*solution* phase). The authors state that AMBRE provides 'guidance' for all three stages, but no detail is given on how this occurs.

ELM [Weber, 1996a; Weber, 1996b] is an ITS for LISP programming that incorporates a case base of worked-out examples. As students are solving a problem, they can ask for an example, which ELM selects for them. ELM's selection criteria is to find "the most similar example to the target problem" [Weber, 1996b]. In actuality, ELM aims to select the example that is the most structurally similar to the target problem, i.e., whose solution is derived by the same rules as the problem's solution (as far as superficial similarity is concerned, unfortunately, it is not clear if or how ELM takes it into account). To find examples that are maximally similar, ELM first generates the solution to the target problem. ELM then compares the problem solution to the example solutions stored in the case base, quantifying each example's similarity to the problem based on the formula:

$$\text{Sim}(E, P) = f(E, P) - g(E, P) - h(P, E)$$

[where P, E are the problem / example solutions]

The function f returns a weighted sum of the number of steps P and E have *in common*. The function g returns the number of solution steps that are in P but not in E , while the

function h returns the number of solution steps that are in E but not in P . Once each example's similarity has been quantified, ELM selects the example with the highest similarity score.

ELM incorporates a very basic student model, corresponding to maintaining information on which rule a student is likely to use when generating the problem solution. In ELM, a given solution step can often be derived by several rules. Initially, the rules all have equal weights. As a student solves problems, ELM updates the weights so that rules the student uses have a higher weight than unused rules.

Of the ITS described in this section, ELM and AMBRE target learning in problem-solving domains that are most similar to the EA-Coach's, because they have a clear notion of correct and incorrect behavior. As we already indicated, no detail is provided on AMBRE's example-selection mechanism, making it impossible to compare it to the EA-Coach's. On the other hand, ELM's example-selection mechanism does share some similarities with the EA-Coach's. Both ELM and the EA-Coach compare problem and example solutions when selecting an example. However, ELM and the EA-Coach do not treat problem/example differences in the same way. First, the EA-Coach assumes that some differences are actually beneficial in helping students learn, while in ELM all differences diminish an example's possibility of being selected. Second, ELM takes into account when the example solution includes *extra* rules not needed for the target problem's solution, which the EA-Coach does not do. Although this is something we may incorporate into the EA-Coach's selection process in the future, further investigation is needed to understand how this type of difference impacts APS. Third, ELM quantifies an example's 'utility' by relying on a formula that only takes into account the similarity between the problem/example solutions. In contrast, the EA-Coach quantifies an example's utility by relying on a decision-theoretic approach. This approach involves taking into account how a student's knowledge and meta-cognitive tendencies *in combination* with problem/example similarity will impact the problem-solving success and learning.

7.2 ITS Supporting Pure Problem Solving

Some ITS support problem solving by providing tutor-generated help instead of examples, i.e., support pure problem solving. In this section, we describe ITS that support pure problem solving but that do not target meta-cognitive skills or rely on a decision-theoretic approach (ITS that support pure problem solving and target meta-cognitive skills/take a decision theoretic-approach will be described in subsequent sections). Andes [Conati, Gertner et al., 1997] is an ITS for Newtonian physics that we introduced in chapter 3. Andes lets student take the initiative in the problem-solving process by affording freedom of interaction: students are not required to follow specific solution paths or show all problem-solving steps. During problem solving, Andes provides two forms of guidance: (1) immediate feedback for correctness on students' problem-solving entries and (2) tailored hints, presented upon a student's request for help or when the system detects the student is floundering. The key to Andes' ability to tailor its support is its student model, which we described in chapter 5. Another ITS that aims to maintain freedom of interaction during problem solving is Ms. Lindquist, which helps students solve algebra word problems [Heffernan and Koedinger, 2002]. As is the case with Andes, students interacting with Ms. Lindquist are free to enter their solution steps in the order/manner of their choice. However, Ms. Lindquist incorporates a more complex tutor model than Andes, which is based on how human tutors support learning of algebra. In particular, the model encodes a set of tutorial strategies, represented via rules, that Ms. Lindquist relies on to respond to students' problem-solving entries. To tailor its support, Ms. Lindquist relies on its student model, which is also represented by a set of rules encoding correct and incorrect domain knowledge.

On the other hand, some ITS maintain stricter control over students' interaction. An example of such an ITS is the Lisp Tutor [Anderson, Conrad et al., 1989], which supports learning of the LISP programming language. The Lisp Tutor constrains students' problem-solving entries by asking them to enter their code top-down and left-to right, and requiring them to repair any errors before moving on. The Lisp tutor provides immediate feedback for correctness on students' problem-solving entries, as well as tailored hints

when students make mistakes or ask for help. Like Andes and Ms. Lindquist, the LISP Tutor relies on its student model to tailor its feedback. In the Lisp Tutor, the student model corresponds to a set of production rules that encode correct and incorrect ways to generate LISP code.

7.3 ITS Supporting Meta-Cognitive Skills

In recent years, there has been increasing interest in developing ITS that target general meta-cognitive skills rather than domain-specific skills. To date, existing tutors have not targeted meta-cognitive skills during APS, and thus none have supported min-analogy (which is specific to APS). On the other hand, the meta-cognitive skill of self-explanation has been targeted in a variety of instructional activities. Instead of selecting examples, however, the standard approach for supporting self-explanation has been to provide students with tools that they can use to derive the explanations and/or provide prompts encouraging self-explanation. With one exception, described in the next section, none of the existing tutors that aim to support meta-cognitive skills rely on a decision-theoretic approach for action selection.

Some ITS support self-explanation during pure example studying, i.e., when students only study examples, without a problem-solving component. The Self-Explanation Assistant (SEA) [Kashihara, Hirashima et al., 1995] supports learning of operating system concepts by presenting worked-out examples that have *gaps* in their solutions. Students are asked to fill in the gaps by self-explaining them. This is accomplished by selecting the explanation from a list in the SEA interface (a process referred to as *gap-filling* self-explanation [VanLehn and Jones, 1993b]).

SEA tailors its support to a given student, realized by including solution gap(s) only if doing so does not incur too high a cognitive load for the student. The cognitive load calculation is based on a student's knowledge of the domain principles to be self-explained, in that the lower a student's knowledge of a principle, the higher the cognitive load incurred by explaining it. Once the gap selection process is complete, SEA

dynamically generates the example content by relying on Natural Language Generation (NLG) techniques. SEA incorporates a basic student model, corresponding to a vector of numeric values representing students' knowledge of domain principles.

Like SEA, the SE-Coach [Conati and VanLehn, 2000] provides tailored support for gap-filling self-explanation, in the domain of Newtonian physics. Specifically, the SE-Coach selects solution gaps only if they correspond to domain principles the student has mastered after studying fully detailed examples. Once the gaps are selected, the SE-Coach relies on NLG techniques to dynamically generate the example content. The SE-Coach also targets two other types of self-explanations

- *step correctness* self-explanation, i.e., justifying a solution step in terms of the domain theory (e.g., [Conati, Larkin et al., 1997; Conati and VanLehn, 2000]);
- *step utility* self-explanation, i.e., relating solution steps to goals in the abstract plan underlying the example solution [Conati, Larkin et al., 1997; Conati and VanLehn, 2000].

The SE-Coach provides interface tools that students can use to generate all three types of self-explanations. It also provides feedback for correctness on the explanations and generates interventions encouraging students to self-explain. The interventions are tailored based on the SE-Coach's student model. The model is a Bayesian network, used to assess how a student's self-explanations reflect her understanding of the example solution. The network is automatically created from the currently open example's solution graph (following the Andes approach that we described in chapter 5). To generate its assessment of the student, the SE-Coach student model integrates information on (1) the time a student spent viewing the example solution steps, (2) self-explanations a student generated via the provided tools and (3) a student's knowledge of the principles needed to derive the corresponding example solution steps.

In addition to supporting self-explanation during pure example studying, there is some work on incorporating support for self-explanation into open learning environments. These environments place less emphasis on supporting learning through structured explicit instruction and more on allowing the learner to freely explore the available

instructional material [Collins and Brown, 1990; White, Shimoda et al., 1999; White, 1993]. The Adaptive Coach for Exploration (ACE) is an environment that helps students to learn about mathematical functions [Bunt, Conati et al., 2001] by supporting self-explanation during the exploratory process [Bunt, Conati et al., 2004; Conati, Merten et al., 2005]. ACE includes a student model corresponding to a dynamic Bayesian network that is used to assess how effectively students are exploring. This involves detecting whether students (1) explore the salient domain concepts and (2) self-explain their exploratory actions, either *explicitly* via the provided tools or *implicitly* (i.e., in their head, without tool usage). Detecting implicit self-explanation is particularly challenging due to lack of hard evidence of its occurrence. The ACE student model meets this challenge by incorporating information on (1) the length of time a student devoted to a given exploratory action, 2) a student's knowledge and self-explanation tendency and (3) her attention patterns in the ACE interface, based on information provided by an eye-tracker. Although some preliminary work has been done on using the student model's assessment to provide tailored support for self-explanation, such as prompts encouraging students to self-explain, this work is still in the developmental phases.

Thus far, we have described ITS that support the meta-cognitive skill of self-explanation during example studying and exploration. Some ITS support self-explanation during pure problem solving, without making examples available. The Geometry Explanation Tutor [Aleven and Koedinger, 2002] targets self-explanation during geometry theorem proving, while Normit-SE's [Mitrovic, 2003] support is provided during database normalization. Both tutors

- target self-explanation for step correctness (i.e., justifying a solution step in terms of the domain theory);
- provide interface tools for generating the explanations, as well as feedback for correctness on the explanations.

Normit-SE and the Geometry Explanation Tutor do not tailor their support according to specific shortcomings in students' self-explanation behaviours. The Geometry

Explanation Tutor prompts students to self-explain *every* problem-solving step, while Normit-SE prompts students to self-explain every *new* or *incorrect* problem-solving step.

The Geometry Explanation Tutor has been extended via a *Help-Seeking (HS)* Tutor that provides support for the meta-cognitive skill of help seeking [Aleven, McLaren et al., 2004]. For instance, two aspects of effective help seeking involve knowing when to ask for help, and not abusing available help by asking for hints too frequently. The HS Tutor relies on a production rule model to detect ineffective help-seeking behaviors and generates prompts to correct them. Note that help abuse is a form of ‘gaming the system’, i.e., attempting to succeed in an educational environment by exploiting properties of the system rather than by learning the material and using that knowledge to generate the solution [Baker, Corbett et al., 2006a]. Since gaming can interfere with learning, there is interest in investigating ways of detecting gaming [Baker, Corbett et al., 2006b] and preventing it [Baker, Corbett et al., 2006a; Roll, Aleven et al., 2006]. All of this work involves ITS that support pure problem solving by providing detailed hints instead of examples. Consequently, although effective help-seeking is related to min-analogy since both skills involve not abusing available help, the manner in which students can abuse help in these tutors involves asking for detailed hints too frequently, instead of by indiscriminately copying from examples.

7.4 ITS Relying on a Decision-Theoretic Approach for Action Selection

In contrast to the ITS described thus far, some ITS rely on a decision-theoretic approach for action selection, quantifying the expected utility of tutorial actions via a user model’s prediction. However, none of these tutors target APS. Thus, to date, no ITS has (1) formalized the utility of an example in terms of its ability to facilitate learning and problem-solving success outcomes from APS or (2) included a probabilistic user model for inferring how APS-specific factors, such as problem/example similarity, impact these outcomes.

The INQPRO [Ting, Zadeh et al., 2006] ITS aims to help students acquire scientific inquiry skills by targeting hypothesis generation in the domain of introductory science. To do so, INQPRO provides learners with an interface that they can use to generate hypotheses. Once a student enters a hypothesis, INQPRO generates a simulation to help the student visualize the outcomes of her predictions. To support the inquiry process, INQPRO provides tailored meta-cognitive interventions that are intended to assist students in generating effective hypotheses.

To decide which intervention to generate, INQPRO relies on a Bayesian network supplemented with utility and decision nodes, i.e., a decision network. The Bayesian network is used to model students' reasoning during the inquiry process, and in particular, to assess students' hypothesis generation. The assessment, in conjunction with the utility and decision nodes embedded in the dynamic network, allows INQPRO to select interventions that have best potential to target specific shortcomings in a given student's exploratory behavior.

Like INQPRO, some tutors take a decision-theoretic approach for action selection, but do not aim to support meta-cognitive skills. All these tutors provide their support during pure problem solving. The CAPIT [Mayo and Mitrovic, 2001] tutor is designed to help students acquire punctuation skills. CAPIT provides two forms of support: (1) it selects problems for students to work on that fall into a given student's *zone of proximal development* (i.e., that are not too hard but also not too easy); (2) it provides students with hints that are generated when students make errors.

CAPIT includes a student model corresponding to a dynamic Bayesian network. The network's nodes represent *constraints*, where a constraint specifies conditions that must be satisfied by all correct solutions to the target problem. When a student's problem-solving entry violates a constraint, this is used to infer information about the student's knowledge. In order to decide which problem or error message to select, CAPIT first uses the Bayesian network to predict the impact of a candidate tutorial action (i.e., a problem or error message). The network's prediction is then quantified via a single-objective utility function, allowing CAPIT to select the tutorial action with the maximum expected

utility. For instance, to find problems within a student's zone of proximal development, CAPIT aims to find ones that are likely to trigger one or two errors over problems triggering none or a large number of errors.

Another tutor that supports learning by selecting problems for students to work on is iTutor [Pek and Poh, 2000]. Like CAPIT, iTutor aims to choose problems that are not too easy but not overly challenging either. iTutor relies on a dynamic Bayesian network to predict how a given problem will impact a student's knowledge, quantifies this prediction via a utility function, and presents the problem with the highest expected utility in terms of helping the student learn as a result of solving it.

Instead of selecting problems for students, the Decision Theoretic (DT)-Tutor [Murray, VanLehn et al., 2004] selects tutorial actions. Examples of tutorial actions include *provide a hint*, *do the step for the student*, *provide positive feedback*, *do nothing*, etc. Although DT-tutor has yet to be integrated into a full ITS, its action-selection approach has been simulated in two domains: mathematical functions and elementary reading skills.

As is the case for CAPIT and iTutor, DT-Tutor's student model corresponds to a dynamic Bayesian network. DT-Tutor follows the Andes approach [Conati, Gertner et al., 2002] by basing the backbone of the network's structure on the target problem's solution graph, which is extended to also model the student's affective state. To model tutorial actions and their utility, *decision* and *utility* nodes are incorporated into the network, thereby converting it into a dynamic Decision network. DT-Tutor uses the network to calculate a candidate tutorial action's expected utility in terms of its impact on a student's knowledge, problem-solving progress and affective state. This process allows DT-Tutor to select the tutorial action with the highest expected utility in terms of helping the student learn, make problem-solving progress and maintain a positive affective state.

Chapter 8

Conclusions and Future Work

The main objectives of this dissertation work have been to design, implement and evaluate an Intelligent Tutoring System for supporting APS. The underlying motivation is that examples are both a natural and effective means of learning: students tend to spontaneously rely on examples during problem-solving activities, and examples are more beneficial aids to problem solving than general procedures alone. However, research shows that learning gains from APS strongly depend on how effectively students use examples, and in particular, what meta-cognitive skills students bring to bear. Given this finding, it seems highly valuable to provide computer-based support for meta-cognitive skills relevant to APS, to complement the extensive availability of computer-based support targeting meta-cognitive skills during other instructional activities.

The dissertation work has fulfilled all three objectives: an ITS for supporting APS was designed, implemented and evaluated. Throughout the course of the thesis work, we have adopted an interdisciplinary approach, encompassing research from Cognitive Science, Artificial Intelligence and Human-Computer-Interaction. In particular, the design of the EA-Coach's support is directly based in cognitive science research on APS and relies on Artificial Intelligence techniques to provide students with tailored support. The evaluation of the system follows established Human-Computer-Interaction methodologies, which we relied on to analyze its pedagogical effectiveness.

The dissertation research has three key contributions [Conati, Muldner et al., 2006; Muldner and Conati, 2005; Muldner and Conati, 2005; Muldner and Conati, 2007], as follows:

1. We bring a contribution to Intelligent Tutoring Systems. In particular, the EA-Coach is the first Intelligent Tutoring System that supports meta-cognitive skills during APS and does so by adaptively selecting examples with various levels of similarity to the target problem. Although there has been growing interest in recent years in devising ITS to support meta-cognitive skills, none of this support has targeted APS or EBLC, making the EA-Coach the only tutor to support the meta-cognitive skills of min-analogy and EBLC. The EA-Coach also distinguishes itself from ITS targeting meta-cognitive skills in the style of support it delivers. Specifically, other ITS provide more explicit support than the EA-Coach. This is because they provide interface tools and/or explicit prompts to encourage the target meta-cognitive skills, which the EA-Coach does not do. Instead, the EA-Coach's key form of scaffolding for meta-cognition corresponds to its example-selection mechanism. While some work has been done on supporting APS via example selection, as we already pointed out none of this work targets meta-cognitive skills. In addition to this distinction, the EA-Coach is unique in its example-selection approach because it is the only tutor that (1) tailors the choice of example to a given student, (2) considers some differences between a problem/example to actually be beneficial in terms of triggering learning while supporting successful

problem solving and (3) adopts a decision-theoretic approach for finding examples that enable both outcomes (learning and problem solving).

2. We bring a contribution to User Modeling. The EA-Coach student (user) model relies on our formal definition of similarity to infer the impact of problem/example similarity, in conjunction with student characteristics, on APS outcomes for a given student. To date, ITS supporting APS incorporate basic student models that assess a student's knowledge based on her problem-solving actions alone, without taking into account her meta-cognitive skills or problem/example similarity. Although some ITS incorporate richer models that like the EA-Coach rely on Bayesian networks for the modeling task, none of these tutors support APS. Consequently, the EA-Coach student model differentiates itself from existing models by explicitly representing (1) APS behaviors corresponding to copying and EBLC, (2) students' min-analogy and EBLC meta-cognitive tendencies, (3) problem/example similarity and by (4) reasoning about how these factors, in conjunction with a student's domain knowledge, impact learning/problem-solving APS outcomes.
3. We bring a contribution to Cognitive Science through the evaluation of the EA-Coach on the influence of problem/example similarity on APS behaviors and subsequent learning and problem-solving outcomes. Work in the cognitive science community shows that (1) examples that are too different from the target problem can hinder learning and problem-solving success, particularly for novice students and (2) very similar examples may help students achieve problem-solving success but do not trigger learning. However, full understanding has yet to be reached on how examples can help students achieve problem-solving success *and* trigger learning. Our evaluation has provided insight into this issue, and in particular, showed that non-trivial superficial differences can help students achieve problem-solving success, while triggering EBLC and min-analogy needed for effective learning.

In the course of fulfilling the thesis objectives, we have opened up interesting avenues for future work, which we now describe.

8.1 Future Work

8.1.1 Additional Scaffolding

The EA-Coach encourages min-analogy and EBLC through fairly implicit means, putting much of the responsibility for learning during APS on the student. As we stated in chapter 4, this is intended to encourage students to take initiative in the learning process. However, a possible drawback is that the system's support may not be sufficiently strong to trigger min-analogy or EBLC for some students. Our evaluation confirmed that some students did indeed require more explicit scaffolding during APS. There are several possibilities regarding how this scaffolding could be realized.

One form of scaffolding could correspond to providing students with tools to help them infer the appropriate domain principles via EBLC. To date, ITS that provide tools for self-explanation have scaffolded only domain-based reasoning (e.g., [Kashihara, Hirashima et al., 1995; Conati and VanLehn, 2000; Aleven and Koedinger, 2002]). For instance, figure 8-1 shows a tool provided by the SE-Coach, where the student generates the self-explanation by selecting from drop-down lists. Notice that the explanation process only involves domain-based reasoning, and the student's input of the explanation is constrained by the system, which allows the SE-Coach to monitor and provide feedback on students' self-explanations. Constraining student's input of explanations was a standard approach until recently, when researchers investigated allowing students to generate free-form explanations. Specifically, the Geometry Explanation Tutor now also allows students to enter the explanation by typing [Aleven, Popescu et al, 2004]. Since with this latter approach students are not constrained in how they generate explanations, they could express EBLC-reasoning to the system. However, the Geometry Tutor can not provide feedback on EBLC, since it does not have the capabilities to recognize overly general/common sense reasoning. Thus, in general, it is an open question with respect to how tools for EBLC should be designed to provide the necessary scaffolding, or how to incorporate the complex domain and student models needed to allow the system to capture and provide feedback on EBLC.

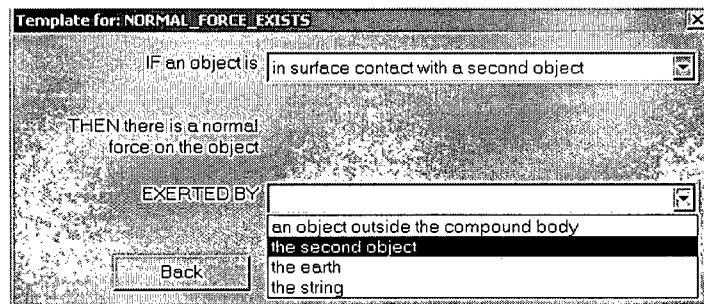


Figure 8-1: The SE-Coach tool used to generate a self-explanation on the existence of a normal force

Cognitive science research shows that students use EBLC reasoning to derive new domain principles when learning *on their own*, i.e., in the absence of any tools. However, if an ITS provides tools supporting self-explanation without scaffolding common sense/overly general reasoning, e.g., of the type shown in figure 8-1, then students could still learn the rule, but without any guidance for EBLC from the system. An advantage of guiding EBLC is that some students have a very low tendency for this kind of reasoning and so require assistance with this process, which may in turn help them gain experience with EBLC so that they can apply it on their own, in the absence of tools. To see if and how this occurs, it would be interesting to compare if and how scaffolding commonsense/overly general reasoning impacts learning outcomes.

In addition to tools, another form of scaffolding could correspond to system-generated meta-cognitive interventions for min-analogy and EBLC, which our evaluation suggested could be beneficial for some students. Since our goal is to maintain freedom of interaction with the EA-Coach, the system should intervene only if a given student's APS behaviors are hindering her learning outcomes. This can be accomplished by relying on the student model's assessment of a given student's capabilities, thereby tailoring the interventions to individual students' needs. A common approach for realizing meta-cognitive interventions involves generating prompts, e.g., to encourage self-explanation [Conati and VanLehn, 2000], or effective help-seeking [Aleven, McLaren et al., 2004]. Another approach involves using animated pedagogical agents to express approval or

lack of it, based on whether students are gaming the system in ways that interfere with learning [Baker, Corbett et al., 2006a]). Yet a third possibility involves making the use of tools, such as the ones to scaffold EBLC, mandatory if the system detects that students are not reasoning effectively.

8.1.2 Scaffolding the Progression of Skill Acquisition

The EA-Coach provides its support within a general ITS framework that incorporates two other tutors: the SE-Coach and Andes. These target different types of instructional activities, i.e., pure example studying without making problems available (SE-Coach) and pure problem solving without the aid of examples (Andes). Cognitive science research suggests that a natural progression in cognitive skill acquisition exists, which starts with example studying, moves to APS and then finally proceeds to pure problem solving [VanLehn, 1996]. Having all three tutors present in one coherent framework opens up the opportunity for investigating this progression. For instance, a question of general interest is whether there are optimal trajectories that students can follow to go from pure example studying (SE-Coach), to APS (EA-Coach), to pure problem solving (Andes). If so, how can they be defined and supported?

8.1.3 Further Analysis

The evaluation of the EA-Coach focused on its adaptive example-selection mechanism. Since the EA-Coach's student model is a key component of the mechanism, the evaluation demonstrated the value of the model, and in particular, that our assumptions embedded into the model are generally appropriate. For our evaluation, however, the model was used for simulation but not for assessment, for reasons stated in section 6.3.3. Although the same model structure and parameters are used during simulation as assessment, direct insight on the model's accuracy in assessment mode could be gained by analyzing its performance in this mode. This analysis could take several forms, as follows:

- Evaluating the accuracy of the model's *intermediate* assessment, i.e., during a student's interaction with the EA-Coach. For instance, it would be interesting to determine if the model's assessment of EBLC at a given point during APS is accurate by correlating its appraisal at this point with data from the protocol analysis.
- Evaluating the accuracy of the model's *final* assessment, i.e., after a student's interaction with the EA-Coach. This could be done, for instance, by correlating the model's final assessment of learning of domain principles with post-test scores on corresponding questions.

8.1.4 Novel Technologies

An interesting extension to the EA-Coach would involve the incorporation of eye-tracking technology. Currently, the EA-Coach's ability to track students' example usage is due to the masking interface. The interface, however, provides less information than an eye tracker. For instance, since the masking interface only covers the example window, it does not provide information on eye-gaze patterns in the problem window (although this could be implemented, it has the potential for being intrusive). The eye tracker is also able to provide fine-grained information on which problem/example constants students are attending to, which could help the model assess whether students are reasoning via transformational analogy. An open question is how to incorporate this information into the model's assessment and how useful it would be in terms of improving the model's accuracy.

An eye tracker also makes it possible to evaluate the masking interface's impact on APS. Recall that in addition to providing information on students' visual attention, the masking interface is intended to provide scaffolding to discourage copying. Since an eye tracker provides information on visual attention without the need to cover the example solution, it would facilitate evaluating if and how the masking interface scaffolds APS.

8.1.5 Refinements to the Student Model

We already discussed one possible refinement to the model above in section 8.1.4, corresponding to incorporating information provided by an eye tracker into the model's operation. Another refinement could correspond to how the model assesses self-explanation. The EA-Coach student model currently aims to infer if a student is self-explaining via EBLC from an example. However, our evaluation indicated that students also self-explain during pure problem solving without relying on examples. The challenge in having the model assess explanation during pure problem solving is that given the current design of the system, there is very little information on whether it is occurring. On the other hand, if tools are incorporated into the system as we propose in section 8.1.1, then tool usage during problem solving would provide information on whether students are reasoning via EBLC.

8.1.6 Subsequent Evaluations

Since the controlled laboratory study described in chapter 6 was the first evaluation of the EA-Coach, its design was geared at providing as much insight as possible into the system's overall pedagogical effectiveness. While we achieved this goal, there are several obvious possibilities related to future evaluations. We have already provided several suggestions above, including

- evaluating the EA-Coach's role in the progression of skill acquisition;
- comparing domain-based vs. EBLC-based tools as scaffolds for self-explanation, as well as evaluating the masking interface's impact on copying.

Another possibility for a future laboratory experiment is investigating the impact of the EA-Coach on learning by measuring pre to post test differences via a standard between-subjects design. Specifically, the study would involve *two* groups of students: the control group would received statically selected examples and the treatment group would receive adaptively selected examples. This design does not afford as much experimental power as the within-subjects design we chose, but does have the advantage of making it easier to measure pre to post test differences between the groups. Instead of a laboratory

experiment, another option for a future evaluation is a longitudinal field study, where students would use the EA-Coach for a longer period of time than is afforded by a laboratory experiment, for instance while enrolled in a physics course. A longitudinal study has the advantages of affording greater ecological validity than a laboratory experiment, and making it possible to evaluate the EA-Coach's long-term impact on students.

References

- [Aleven and Ashley, 1997] Aleven, V. and Ashley, K. D. 1997. Teaching case-based argumentation through a model and examples: Empirical evaluation of an intelligent learning environment. In *Proceedings of the Eighth World Conference of Artificial Intelligence in Education (AIED'07)*, (Kobe Japan), pages 87-94.
- [Aleven and Koedinger, 2002] Aleven, V. and Koedinger, K. 2002. An effective meta-cognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, 26(2), pages 147-179.
- [Aleven, McLaren et al., 2004] Aleven, V., McLaren, B. and Koedinger, K. 2004. Toward Tutoring Help Seeking: Applying Cognitive Modeling to Meta-Cognitive Skills. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems (ITS'04)*, (Maceo, Brazil), pages 227-239.
- [Aleven, Popescu et al., 2004] Aleven, V., Popescu, O., Torrey, C. and Koedinger, K. 2004. Evaluating the Effectiveness of a Tutorial Dialogue System for Self-Explanation. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems, (ITS'04)*, (Maceo, Brazil), pages 443-454.

- [Anderson, Farrell et al., 1984] Anderson, J. R., Farrell, R. and Saurers, R. 1984. Learning to program in Lisp. *Cognitive Science*, 8, pages 87-129.
- [Anderson, Conrad et al., 1989] Anderson, J. R., Conrad, F. G. and Corbett, A. T. 1989. Skill acquisition and the {LISP} tutor. *Cognitive Science*, 14(4), pages 467-505.
- [Anderson and Thomson, 1989] Anderson, J. and Thomson, R. 1989. Use of analogy in a production system architecture. *Similarity and Analogical Reasoning*, Cambridge University press, New York, pages 367-397.
- [Anderson, 1993] Anderson, J. R. 1993. *Rules of the Mind*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- [Baker, Corbett et al., 2006a] Baker, R. S. J. d., Corbett, A. T., Koedinger, K. R., Evenson, E., Roll, I., Wagner, A. Z., Naim, M., Raspat, J., Baker, D. J. and Beck, J. 2006. Adapting to When Students Game an Intelligent Tutoring System. In *Proceedings of the Eight International Conference on Intelligent Tutoring Systems (ITS'06)*, (Jhongli Taiwan), pages 392-401.
- [Baker, Corbett et al., 2006b] Baker, R. S. J. d., Corbett, A. T., Koedinger, K. and Roll, I. 2006. Generalizing Detection of Gaming the System Across a Tutoring Curriculum. In *Proceedings of Eight International Conference on Intelligent Tutoring Systems (ITS'06)*, (Jhongli, Taiwan), pages 402-411.
- [Bassok, Wu et al., 1995] Bassok, M., Wu, L. and Olseth, K. 1995. Judging a book by its cover: Interpretive effects of content on problem-solving transfer. *Memory & Cognition*, 23(3), pages 221-252.
- [Bielaczyc, Pirolli et al., 1995] Bielaczyc, K., Pirolli, P. and Brown, A. L. 1995. Training in self-explanation and self-regulation strategies: Investigating the effects of

knowledge acquisition activities on problem-solving. *Cognition and Instruction* 13(2), pages 221-252.

[Bloom, 1984] Bloom, B. S. 1984. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13, pages 4-16.

[Bunt, Conati et al., 2001] Bunt, A., Conati, C., Hugget, M. and Muldner, K. 2001. On Improving the Effectiveness of Open Learning Environments through Tailored Support for Exploration. In *Proceedings of the Tenth World Conference on Artificial Intelligence and Education*, (San Antonio, USA), pages 365-376.

[Bunt, Conati et al., 2004] Bunt, A., Conati, C. and Muldner, K. 2004. Scaffolding self-explanation to improve learning in exploratory learning environments. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems (ITS'04)*, (Maceo, Brazil), pages 656-667.

[Burke and Kass, 1995] Burke, R. and Kass, A. 1995. Supporting Learning through Active Retrieval of Video Stories. *Expert Systems with Applications*, 9(3), pages 361-378.

[Chi, Feltovich et al., 1981] Chi, M. T. H., Feltovich, P. and Glaser, R. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, pages 121-152.

[Chi, Bassok et al., 1989] Chi, M., Bassok, M., Lewis, M., Reimann, P. and Glaser, R. 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 15, pages 145-182.

[Chi and VanLehn, 1991] Chi, M. and VanLehn, K. 1991. The content of physics self-explanations. *The Journal of the Learning Sciences*, 1, pages 69-105.

- [Chi, 2000] Chi, M. (2000). Self-Explaining: The dual process of generating inferences and repairing mental models. *Advances in Instructional Psychology*. R. Glaser, Lawrence Erlbaum Associates, pages 161-238.
- [Clement, 1996] Clement, R. 1996. *Making Hard Decisions*, Pacific Grove: Duxberry Press.
- [Collins and Brown, 1990] Collins, A. and Brown, J. S. 1990. The computer as a tool for learning through reflection. *Learning issues for intelligent tutoring systems*, New York, Springer, pages 1-18.
- [Conati, Gertner et al., 1997] Conati, C., Gertner, A., VanLehn, K. and Druzdzel, M. 1997. On-line student modeling for coached problem solving using Bayesian networks. In *Proceedings of the Sixth International Conference on User Modeling (UM'97)*, (Sardinia, Italy), pages 231-242.
- [Conati, Larkin et al., 1997] Conati, C., Larkin, J. and VanLehn, K. 1997. A computer framework to support self-explanation. In *Proceedings of the Eighth World Conference of Artificial Intelligence in Education (AIED'07)*, (Kobe Japan), pages 279-276.
- [Conati and VanLehn, 2000] Conati, C. and VanLehn, K. 2000. Toward Computer-based Support of Meta-cognitive Skills: A Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education*, 11, pages 389-415.
- [Conati, Gertner et al., 2002] Conati, C., Gertner, A. and VanLehn, K. 2002. Using Bayesian Networks to Manage Uncertainty in Student Modeling. *Journal of User Modeling and User-Adapted Interaction*, 12(4), pages 371-417.
- [Conati, Merten et al., 2005] Conati, C., Merten, C., Muldner, K. and Ternes, D. 2005. Exploring Eye Tracking to Increase Bandwidth in User Modeling. In

Proceedings of the Tenth International Conference on User Modeling (UM'05), (Edinburgh, Scotland), pages 357-366.

[Conati, Muldner et al., 2006] Conati, C., Muldner, K. and Carenini, G. 2006. From Example Studying to Problem Solving via Tailored Computer-Based Meta-Cognitive Scaffolding: Hypotheses and Design. *Technology, Instruction, Cognition and Learning (TICL): Special Issue on Problem Solving Support in Intelligent Tutoring Systems*. 4(2), pages 139-190.

[Dean and Kanazawa, 1989] Dean, T. and Kanazawa, K. 1989. A Model for Reasoning about Persistence and Causation. *Computational Intelligence* 5(3), pages 142-150.

[Dempster, Laird et al., 1977] Dempster, A., Laird, N. and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1-38.

[Ericsson and Simon, 1980] Ericsson, K. and Simon, H. 1980. Verbal Reports as Data. *Psychological Review*, 87(3), 215-250.

[Flavell, 1976] Flavell, J. H. 1976. Metacognitive Aspects of Problem Solving. *The Nature of Intelligence*. Hillsdale, New Jersey, Lawrence Erlbaum, pages 231-235.

[Guin-Duclosson, Jean-Duabias et al., 2002] Guin-Duclosson, N., Jean-Duabias, S. and Nogry, S. 2002. The Ampre-Ile: How to Use Case-Based Reasoning to Teach Methods. In *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems (ITS'02)*, (Biarritz, France), pages 783-791.

[Halliday and Resnick, 1988] Halliday, D. and Resnick, R. 1988. *Fundamentals of Physics (Third Edition)*. New York, NY, Wiley and Sons.

- [Halloun and Hestenes, 1985] Halloun, I. and Hestenes, D. 1985. The Initial Knowledge State of College Physics Students. *American Journal of Physics*, 53, pages 1043-1055.
- [Heffernan and Koedinger, 2002] Heffernan, N. and Koedinger, K. 2002. An Intelligent Tutoring System Incorporating a Model of an Experienced Human Tutor. In *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems (ITS'02)*, (Biarritz, France), pages 596-608.
- [Henrion, 1989] Henrion, M. Some practical issues in constructing belief networks. 1989. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence (UAI'89)*, (Wisconsin, USA), pages 161-173.
- [Holyoak and Koh, 1987] Holyoak, K. J. and Koh, K. 1997. Surface and structural similarity in analogical transfer. *Memory and Cognition*, 15(4), pages 332-340.
- [Howell, 1997] Howell, D. 1997. *Statistical Methods for Psychology*. London, Wadsworth Publishing Company.
- [Keeney and Raiffa, 1976] Keeney, R. and Raiffa, H. 1976. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York, Wiley.
- [Kashihara, Hirashima et al., 1995] Kashihara, A., Hirashima, T. and Toyoda, J. 1995. A Cognitive Load Application in Tutoring. *User Modeling and User-Adapted Interaction*, 4, pages 279-303.
- [Koedinger, Anderson et al., 1997] Koedinger, K. R., Anderson, J. R., Hadley, W. H. and Mark, M. 1997. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8, pages 30-43.
- [LeFevre and Dixon, 1986] LeFevre, J. and Dixon, P. 1986. Do written instructions need examples? *Cognition and Instruction*, 3, pages 1-30.

- [Lynch, Ashley et al., 2006] Lynch, C., Ashley, K., Aleven, V. and Pinkwart, P. 2006. Defining Ill-Defined Domains: A literature survey. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains (ITS'07)*, (Jhongli, Taiwan), pages 1-10.
- [Mayo and Mitrovic, 2001] Mayo, M. and Mitrovic, A. 2001. Optimizing ITS behaviour with Bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education*, 12, pages 124-153.
- [Mitrovic, 2003] Mitrovic, A. 2003. Supporting Self-Explanation in a Data Normalization Tutor. In *Supplementary proceedings of the Eleventh International Conference of Artificial Intelligence in Education (AIED'03)*, pages 565-577.
- [Morgan and Ritter, 2002] Morgan, P. and Ritter, S. 2002. An experimental study of the effects of Cognitive Tutor Algebra 1 on student knowledge and attitude. *Carnegie Learning*,
http://www.carnegielearning.com/web_docs/morgan_ritter_2002.pdf.
- [Muldner, 2002] Muldner, K. 2002. *Supporting Analogical Problem Solving in an Adaptive Learning Environment*. Ph.D. Thesis Proposal, University of British Columbia, Canada.
- [Muldner and Conati, 2005a] Muldner, K. and Conati, C. Providing adaptive support for meta-cognitive skills to improve learning. 2005. In *Metacognition in Computation: 2005 Spring AAAI Spring Symposium*, (California, USA), pages 86-92.
- [Muldner and Conati, 2005b] Muldner, K. and Conati, C. 2005. Using Similarity to Infer Meta-Cognitive Behaviors During Analogical Problem Solving. In *Proceedings of the Tenth International Conference on User Modeling (UM'05)*, (Edinburgh, UK), pages 134-143.

- [Muldner and Conati, 2007] Muldner, K. and Conati, C. 2007. Evaluating a Decision-Theoretic Approach to Tailored Example Selection. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, (Hyderabad, India), pages 483-489.
- [Murray, VanLehn et al., 2004] Murray, R. C., VanLehn, K. and Mostow, J. 2004. Looking ahead to select tutorial actions: A decision-theoretic approach. *Journal of Artificial Intelligence in Education*, 14(3), pages 235-278.
- [Nogry, Jean-Daubias et al., 2004] Nogry, S., Jean-Daubias, S. and Duclosson, N. 2004. ITS Evaluation in Classroom: The Case of Ambre-AWP. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems (ITS'04)*, (Maceo, Brazil), pages 511-520.
- [Novick, 1988] Novick, L. R. 1988. Analogical transfer, problem similarity and expertise. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 14, pages 510-520.
- [Novick and Holyoak, 1991] Novick, L. R. and Holyoak, K. J. 1991. Mathematical problem solving by analogy. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 17(3), pages 398-415.
- [Novick, 1995] Novick, L. 1995. Some Determinants of Successful Analogical Transfer in the Solution of Algebra Word Problems. *Thinking and Reasoning*, 1(1), pages 1-30.
- [Pearl, 1988] Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, Morgan-Kaufmann.
- [Pek and Poh, 2000] Pek, P. K. and Poh, K. L. 2000. Using Decision Networks for Adaptive Tutoring. In *Proceedings of the International Conference on*

Computers in Education / International Conference on Computer-Assisted Instruction, (Taiwan), pages 1076-1084.

[Pirolli and Anderson, 1985] Pirolli, P. L. and Anderson, J. R. 1985. The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39, pages 240-272.

[Quilici and Mayer, 1996] Quilici, J. L. and Mayer, R. E. 1996. Role of examples in how students learn to categorize statistics word problems. *Journal of Educational Psychology*, 88(1), pages 144-161.

[Reed, Dempster et al., 1985] Reed, S. K., Dempster, A. and Ettinger, M. 1985. Usefulness of analogous solutions for solving algebra word problems. *Journal of Experimental Psychology: Learning, Memory and Cognition* 11, pages 106-125.

[Reed, 1987] Reed, S. K. A structure-mapping model for word problems. 1987. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 13, pages 124-139.

[Reed, Ackinclose et al., 1990] Reed, S. K., Ackinclose, C. C., & Voss, A. A. 1990. Selecting analogous solutions: Similarity versus inclusiveness. *Memory & Cognition*, 18, pages 83-98.

[Reed and Bolstad, 1991] Reed, S. K. and Bolstad, C. A. 1991. Use of examples and procedures in problem solving. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 17(4), pages 753-766.

[Reed, Willis et al., 1994] Reed, S. K., Willis, D. and Guarino, J. 1994. Selecting examples for solving word problems. *Journal of Educational Psychology*, 86(3), pages 380-388.

- [Renkl, 1997] Renkl, A. 1997. Learning from worked-examples: A study on individual differences. *Cognitive Science*, 21(1), pages 1-30.
- [Renkl, 1999] Renkl, A. 1999. Learning Mathematics from Worked-Out Examples: Analyzing and Fostering Self-Explanation. *European Journal of Psychology and Education*, 21, pages 1-29.
- [Ringenberg and VanLehn, 2006] Ringenberg, M. and VanLehn, K. 2006. Scaffolding Problem Solving with Annotated, Worked-Out Examples to Promote Deep Learning. In *Proceedings of the Eight International Conference on Intelligent Tutoring Systems*, (Jhongli, Taiwan), pages 625-634.
- [Roll, Aleven et al., 2006] Roll, I., Aleven, V., McLaren, B., Ryu, E., Baker, R. S. J. d. and Koedinger, K. 2006. The Help Tutor: Does Metacognitive Feedback Improve Students' Help-Seeking Actions, Skills and Learning? In *Proceedings of the Eight International Conference on Intelligent Tutoring Systems*, (Jhongli, Taiwan), pages 360-369.
- [Ross, 1987] Ross, B. 1987. This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory and Cognition* 13, pages 629-639.
- [Russell and Norvig, 1995] Russell, S. and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Los Altos, CA, Morgan-Kaufman.
- [Schoenfeld and Herrmann, 1982] Schoenfeld, A., Herrmann, D. 1982. Problem Perception and Knowledge Structure in Expert and Novice Mathematical Problem Solvers. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 8(5), pages 484-494.
- [Shute and Psotka, 1996] Shute, V. J. and Psotka, J. 1996. Intelligent Tutoring Systems: Past, Present and Future. *Handbook of Research on Educational*

Communications and Technology, D. Jonassen, Scholastic Publications, pages 570-600.

- [Silver, 1979] Silver, E. A. 1979. Student Perceptions of Relatedness among Mathematical Verbal Problems. *Journal for Research in Mathematics Education*, 10(3), pages 195-210.
- [Ting, Zadeh et al., 2006] Ting, Choo-Yee, Beik Zadeh, M. Reza and Chong, Yen-Kuan. 2006. A Decision-Theoretic Approach to Scientific Inquiry Exploratory Learning Environment. In *Proceedings of the Eight International Conference on Intelligent Tutoring Systems (ITS'06)*, (Jhongli Taiwan), pages 85-94.
- [VanLehn, 1988] VanLehn, K. 1988. Student modeling. *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ, Lawrence Erlbaum Associates, pages 55-78.
- [VanLehn, Ball et al., 1990] VanLehn, K., Ball, W. and Kowalski, B. 1990. Explanation-based learning of correctness: Towards a model of the self-explanation effect. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, (Massachusetts, USA), pages 717-724.
- [VanLehn, 1991] VanLehn, K. 1991. Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science*, 15(1), pages 1-47.
- [VanLehn, Jones et al., 1992] VanLehn, K., Jones, R. M. and Chi, M. 1992. A model of the self-explanation effect. *The Journal of the Learning Sciences*, 2(1), pages 1-59.
- [VanLehn and Jones, 1993a] VanLehn, K. and Jones, R. M. 1993. Better learners use analogical problem solving sparingly. In *Proceedings of the Tenth Annual Conference on Machine Learning*, (Massachusetts, USA), pages 338-345.

- [VanLehn and Jones, 1993b] VanLehn, K. and Jones, R. M. 1993. What mediates the self-explanation effect? Knowledge gaps, schemas or analogies? In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, (Colorado, USA), pages 1034-1039.
- [VanLehn, 1996] VanLehn, K. 1996. Cognitive skill acquisition. *Annual Review of Psychology*, 47, pages 513-539.
- [VanLehn, 1998] VanLehn, K. 1998. Analogy Events: How Examples are Used During Problem Solving. *Cognitive Science*, 22(3), pages 347-388.
- [VanLehn, 1999] VanLehn, K. 1999. Rule-Learning Events in the Acquisition of a Complex Skill: An Evaluation of Cascade. *The Journal of the Learning Sciences*, 8(1), pages 71-125.
- [Vicente and Torenvliet, 2000] Vicente, K. J. and Torenvliet, G. L. 2000. The earth is spherical ($p < 0.05$): Alternative methods of statistical inference. *Theoretical Issues in Ergonomics Science*, 1(3), pages 248-271.
- [Weber, 1996a] Weber, G. 1996. Episodic Learner Modeling. *Cognitive Science*, 20, pages 195-236.
- [Weber, 1996b] Weber, G. 1996. Individual Selection of Examples in an Intelligent Learning Environment. *Journal of Artificial Intelligence in Education*, 7(1), pages 3-33.
- [White, 1993] White, B. Y. 1993. ThinkerTools: Causal models, conceptual change and science education. *Cognition and Instruction*, 10(1), pages 1-100.

[White, Shimoda et al., 1999] White, B., Shimoda, T. and Frederiksen, J. 1999. Enabling students to construct theories of collaborative inquiry and reflective learning: computer support for meta-cognitive development. *International Journal of Artificial Intelligence in Education*, 10, pages 151-182.

Appendix 1

Heuristics for Implementing Case5b of the Algorithm in section 5.4.2.1

There are three heuristics defined to handle Case5b of the algorithm in Figure 5-3:

- Heuristic 1, used for solution graph nodes representing vector-component equations (e.g., sp_n , Figure A-1)
- Heuristic 2, used for solution graph nodes representing the sign of a vector-component equation (e.g., sp_{n+1} , Figure A-1)
- Heuristic 3, used as the fallback if heuristics (1) and (2) do not apply.

For the following description we rely on terminology introduced in Figure 5-3, which we summarize here:

- sp is a problem step (i.e., fact) in the Bayesian network derived by rule R ,
- S is the set of nodes $S = \{ se_1, \dots, se_n \}$ in the example's solution graph that were derived by R .

Heuristics 1 and 2

These heuristics apply to equations specifying the components of a vector (e.g., ' $T_x = -T \cos(40)$ ', Figure A-1, top left). The subsequent discussion of the corresponding heuristics for dealing with Case5b requires an understanding of how this type of step is represented in the solution graph. A vector-component equation has *two* nodes in the solution graph, because it is derived by two rules: one to generate the vector component equation, but without its sign (see node sp_n Figure A-1, left) and one to generate the sign (see node sp_{n+1} Figure A-1, left). Although we could have represented the principle to derive the equation by one rule, it would not allow the framework to model students' knowledge on a fine-grained level (for instance, to distinguish between instances when students knew how to generate the equation but not its sign).

Figure A-1 illustrates a situation requiring the two heuristics. Heuristic 1 is needed for node sp_n encoding a vector-component equation. This node is related to two nodes in the example's solution graph, se_n and se_m , since they are derived by the same rule as sp_n . Heuristic 2 is needed for node sp_{n+1} encoding the vector-component equation's sign, since it also is related to two nodes in the example's solution graph (i.e., se_{n+1} and se_{m+1}).

Heuristic 1. To identify which node in S to compare with sp :

- [Case 1-A] If a node se exists in S such that sp and se correspond to the same force *type* (e.g., tension, weight, etc., as is the case for sp_n and se_n in Figure A-1) and there is only one such node se then

- use node se in Step 6 of the algorithm in Figure 5-3 (i.e., compare sp with se).

This heuristic is based on the assumption that force *type* is a salient attribute that guides how students transfer from the example. For instance, a student who aims to generate a vector-component equation for the *tension* force by copying from the example will refer to the vector-component equation for the *tension* force in the example solution.

- [Case 1-B] Otherwise, there is no obvious way of identifying which step in S should be compared with se . In this case, the fallback heuristic is used, heuristic 3.

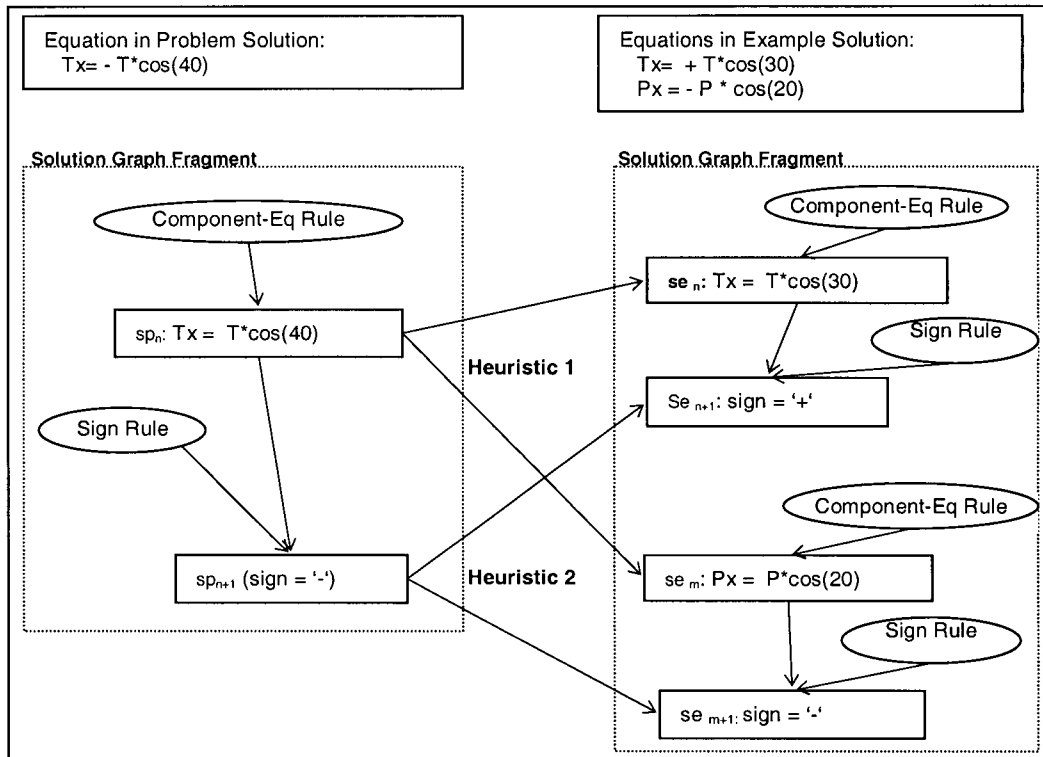


Figure A-1: Representation of vector-component equations in the solution graph

Heuristic 2. Since nodes in the solution graph encoding the *sign* of a vector component equation are related to the equation itself, this is utilized when determining the similarity. Specifically, for a node sp in the problem's solution graph corresponding to the sign of a component equation:

- find sp 's parent node $sp_{component_equation}$ for the corresponding the component equation (e.g., for node sp_{n+1} in Figure A-1, this is node sp_n);

- [Case 2-A] If the parent node $sp_{component_equation}$ has a corresponding node $se_{component_equation}$ in the example solution graph according to Heuristic 1, Case 1-A (e.g., for node sp_n in Figure A-1, this is node se_n) then
 - find $se_{component_equation}$'s child node se_{sign} corresponding to the sign of the component equation (e.g., for node se_n in Figure A-1, this is node se_{n+1});
 - use node se_{sign} in Step 6 of the algorithm in Figure 5-3 (i.e., compare sp with se_{sign}).
- [Case 2-B] Otherwise, there is no obvious way of identifying which step in S should be compared with se . In this case, the fallback heuristic is used, heuristic 3.

Heuristic 3

The fallback heuristic is to compare sp with each node se in S and set the similarity to:

- *Trivial* if a node se exists in S such that includes a trivial difference with sp ,
- *NonTrivial* otherwise.

Note that if Case 1-B is reached, then Step 6 of the algorithm in Figure 5-3 is skipped.

Appendix 2

Pre and Post Test used During Primary Pilot (Chapter 3)

PreTest

Name: _____

Date/Time: _____

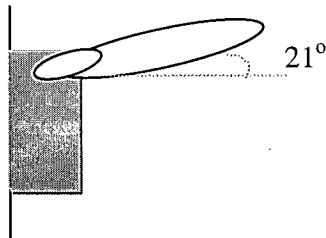
For all problems, assume a frictionless surface !!!

Forces / Directions:

For the following problems, please draw the free-body diagrams by showing the direction of all forces acting upon an object in the given situations (Don't worry about reflecting the magnitude of the force in the diagram by scaling the forces – just draw each force in the correct direction, and make sure to label each force clearly).

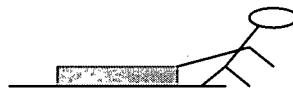
Problem 1

Jane is trying to hang a very very small picture. To see if it is straight, she pushes it into the wall and holds it there with her finger. Draw all the forces/ their direction that are acting on the picture.



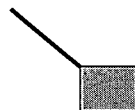
Problem 2

Bob pulls a large package along the ground with a force of $F=100\text{N}$, applied at 34 degrees to the horizontal. Draw all the forces/ their direction that are acting on the package.



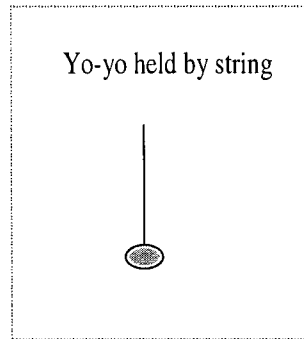
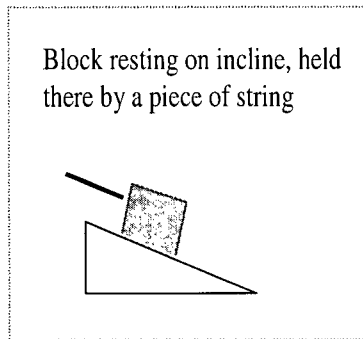
Problem 3

A crate has a string attached to it, which is pulled taut. Draw all the forces /their direction that are acting on the crate.



Axes

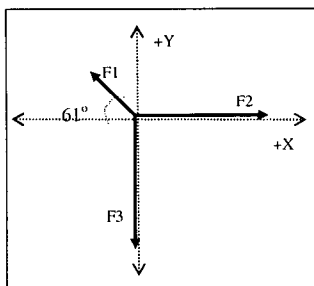
For the following situations, draw in the axes that you would use to define the component equations.



Components

Below is shown a free-body diagram, including the axis (shown in a dotted line) and the angles.

For each force (labeled F1, F2, F3 below), specify in the space provided the horizontal and vertical components.



Horizontal components (X):

F1_x: _____

F2_x: _____

F3_x: _____

Vertical components (Y):

F1_y: _____

F2_y: _____

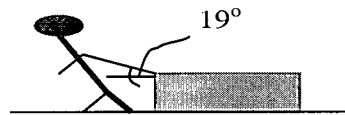
F3_y: _____

Word Problem

Please show ALL your work – the more steps you show, the more helpful it will be!

Don't worry about calculating things like $\cos(15)$ – just leave it in that form.

Bob pulls a loaded box of stuff, with a force of 140 N. The box has a mass 40 kg. This force is applied at 19 degrees from the horizontal.



- 1) What physics principle (or law) would you use to solve this problem?
- 2) Find the normal force N exerted by the ground on the box.

PostTest

Name: _____

Date/Time: _____

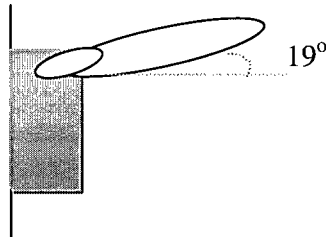
For all problems, assume a frictionless surface, unless otherwise stated.

Forces / Directions:

For the following problems, please draw the free-body diagrams by showing the direction of all forces acting upon an object in the given situations. Don't worry about reflecting the magnitude of the force in the diagram by scaling the forces – just draw each force in the correct direction, and make sure to label each force clearly.

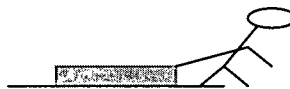
Problem 1.

A boy pushes an eraser into the chalkboard, holding it there with his finger. Draw all the forces/ their direction that are acting on the eraser.



Problem 2.

Ann pulls a sled with a force of $F=80\text{N}$, applied at 45 degrees to the horizontal. Draw all the forces/ their direction that are acting on the sled.



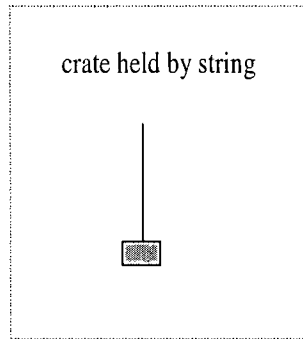
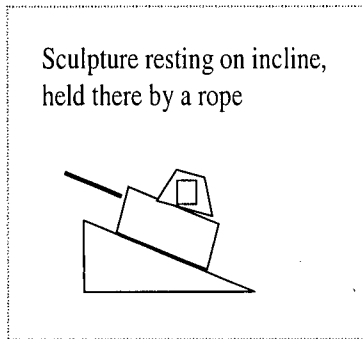
Problem 3.

A string is attached to a toy, as shown below, and pulled so that it is taut. Draw all the forces /their direction that are acting on the toy.



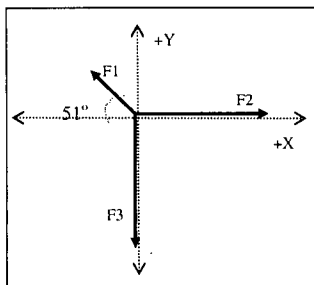
Axes

For the following situations, draw in the axes that you would use to define the component equations.



Components

Below is shown a free-body diagram, including the axis (shown in a dotted line). For each force (labeled F1, F2, F3 below), specify in the space provided the horizontal and vertical components.



Horizontal components (X):

F1_x: _____

F2_x: _____

F3_x: _____

Vertical components (Y):

F1_y: _____

F2_y: _____

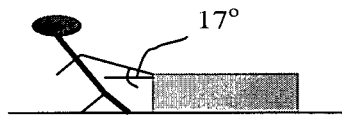
F3_y: _____

Word Problem

Please show ALL your work – the more steps you show, the more helpful it will be!

Don't worry about calculating things like $\cos(15)$ – just leave it in that form.

Ann pulls a box of books. The box has a mass 45 kg. This force is applied at 17 degrees from the horizontal.



1. What physics principle (or law) would you use to solve this problem?
2. Find the normal force N exerted on the box.

Appendix 3

Pre and Post Test used in Evaluation of the EA-Coach

PreTest

Name:

Date/Time:

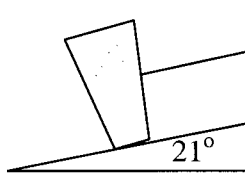
For all problems, assume a frictionless surface!

For the problems on this page, please draw the free-body diagrams:

- show the direction of all forces acting upon an object in the given situations
- don't worry about reflecting the magnitude of the force in the diagram by scaling the forces – just draw each force in the correct direction, and make sure to label each force clearly.

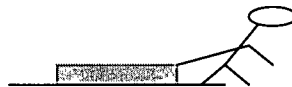
Problem 1

A vase with a mass of 2 kg sits on a frictionless shelf that is inclined 21 degrees from the horizontal. The vase is held in place by a taut string that runs parallel to the incline. Draw all the forces/ their direction that are acting on the vase.



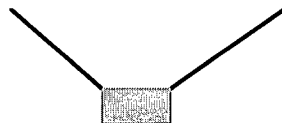
Problem 2

Bob pulls a large package along the ground with a force of $F=100\text{N}$, applied at 34 degrees to the horizontal. Draw the acceleration vector and all the forces that are acting on the package.



Problem 3

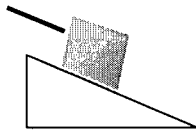
A picture hangs from two wires as shown below. Draw all the forces/ their directions that are acting on the picture.



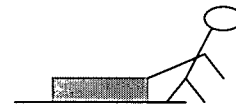
Axis

For the following problems, don't answer the question – just draw in the axis that you would use to define the component equations.

Block of mass 2kg resting on incline, held there by a piece of string which runs parallel to the incline. Find the normal force on the block.

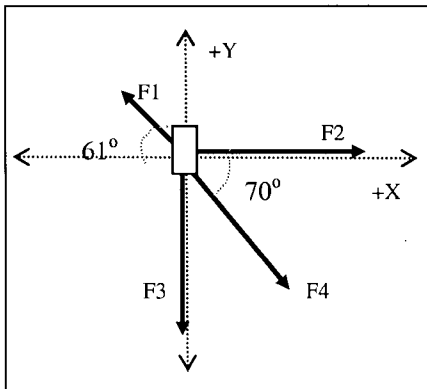


A man pulls a crate with a force P , applied at an angle of 31 degrees to the horizontal, with a magnitude of 100N. The weight of the crate is 40 kg, its acceleration is unknown. What is the normal force on the crate?



Components

A box is being pulled by 4 forces. Below is shown the free-body diagram, including the axis (shown in a dotted line) and the angles. For each force (labeled F_1 , F_2 , F_3 , F_4 below), specify in the space provided the equations for the horizontal and vertical components of each force acting on the box.



Horizontal components (X):

F_{1x} : _____

F_{2x} : _____

F_{3x} : _____

F_{4x} : _____

Vertical components (Y):

F_{1y} : _____

F_{2y} : _____

F_{3y} : _____

F_{4y} : _____

Word Problems

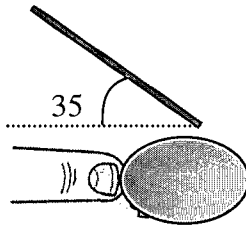
- Please show ALL your work – the more steps you show, the more helpful it will be.
- Assume no friction,
- Don't worry about calculating things like $\cos(15)$ – just leave it in that form.

Question 1

A 0.5 kg toy is hanging from a string. A child pushes on the toy so that the string makes an angle of 35 degrees with the horizontal. The push is applied parallel to the horizontal (i.e. the angle it makes with the ground is 0 degrees), but we don't know its magnitude. The toy is not moving. The tension in the string at this point is 8.5 N

What is the magnitude of the pushing force on the toy?

Please draw free-body diagram below as part of your answer.

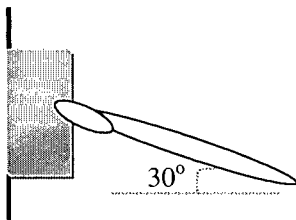


Question 2

Jane pushes a 4 kg book into the wall. She applies the push at an angle of 30 degrees with the horizontal, and a magnitude of 105 N. The book is sliding up the wall with some acceleration.

What is the acceleration of the block ? (assume no friction!)

Please draw free-body diagram as part of your answer, include the acceleration vector.



PostTest

Name: _____

Date/Time: _____

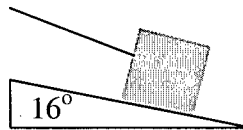
For all problems, assume a frictionless surface!

For the problems on this page, please draw the free-body diagrams:

- show the direction of all forces acting upon an object in the given situations
- don't worry about reflecting the magnitude of the force in the diagram by scaling the forces – just draw each force in the correct direction, and make sure to label each force clearly.

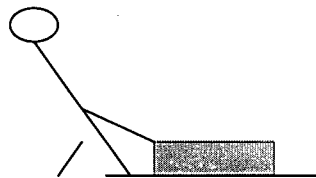
Problem 1

A box is held in place on an incline by a tight rope, which runs parallel to the incline. Draw all the forces/ their direction that are acting on the box.



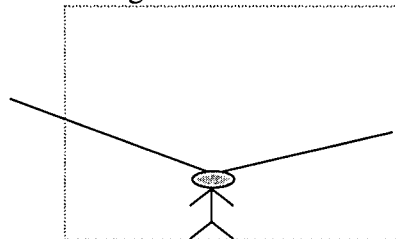
Problem 2

Ann pulls a sled with a force of $F=80\text{N}$, applied at 45 degrees to the horizontal. Draw the acceleration vector and all the forces that are acting on the sled.



Problem 3

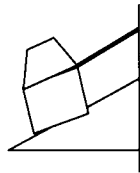
A circus stuntwoman has fallen and hangs suspended by two ropes. Draw all the forces/ their direction that are acting on the stuntwoman.



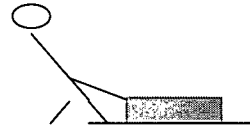
Axis

For the following problems, don't answer the question – just draw in the axis that you would use to define the component equations.

Block of mass 13kg resting on incline, held there by a piece of string which runs parallel to the incline. Find the normal force on the block.

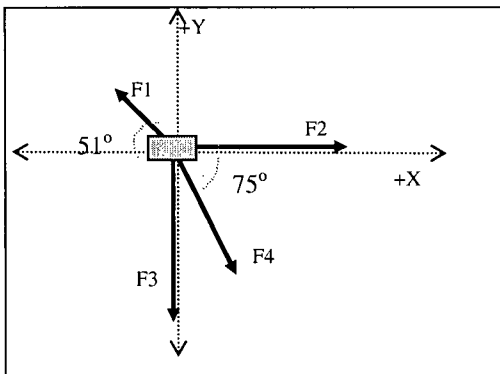


A man pulls a crate with a force P with a magnitude of 70N, applied at an angle of 30 degrees to the horizontal. The weight of the crate is 22 kg, its acceleration is unknown. Find the magnitude of the normal force on the crate.



Components

A box is being pulled by 4 forces. Below is shown the free-body diagram, including the axis (shown in a dotted line). For each force (labeled F_1 , F_2 , F_3 , F_4 below), specify in the space provided the equations for the horizontal and vertical components acting on the box.



Horizontal components (X):

F_{1x} : _____

F_{2x} : _____

F_{3x} : _____

F_{4x} : _____

Vertical components (Y):

F_{1y} : _____

F_{2y} : _____

F_{3y} : _____

F_{4y} : _____

Word Problems

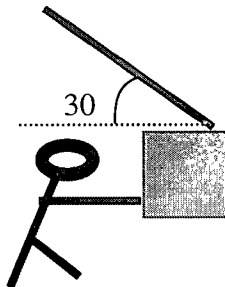
- Please show **ALL** your work – the more steps you show, the more helpful it will be.
- Assume no friction,
- Don't worry about calculating things like $\cos(15)$ – just leave it in that form.

Question 1

A 3 kg block is hanging from a rope. A man pushes on the block so that the string makes an angle of 30 degrees with the horizontal. The push is applied parallel to the horizontal (i.e. the angle it makes with the ground is 0 degrees), but we don't know its magnitude. The block is not moving. The tension in the rope at this point is 58 N

What is the magnitude of the pushing force on the block?

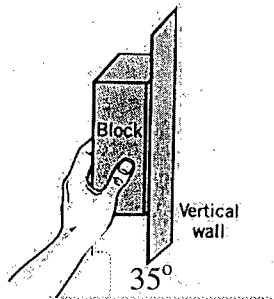
Please draw free-body diagram as part of your answer.



Question 2

Bob pushes a 3.5 kg block into the wall. He applies the push at an angle of 35 degrees with the horizontal, and a magnitude of 100 N. The block is sliding up the wall with some acceleration. What is this acceleration of the block? (Ignore friction in this problem!)

Please draw free-body diagram, including the acceleration vector as part of your answer.



Appendix 4

UBC Research Ethics Board Certificates