ABSTRACT

MOSTAFAVI, BEHROOZ ZAKARIA. Improving Individualized Instruction in a Logic Tutor using Data-driven Methods. (Under the direction of Dr. Tiffany Barnes.)

Intelligent tutoring systems (ITS) have the greatest potential to increase problem-solving performance when they adapt to individual students and domains. However, developing an ITS to adapt to individual students requires the involvement of experts to provide knowledge about both the academic domain and novice student behavior in that domain's curriculum. Because of the large possible range of problem-solving behavior for any individual topic, the amount of expert involvement required to create an effective, adaptable tutoring system can be high, especially in open-ended problem-solving domains.

Data-driven methods have shown much promise in increasing ITS effectiveness by analyzing previous data in order to quickly adapt to individual students. Intelligent tutoring systems that incorporate data-driven methods can present appropriate problems and challenges for each new student, provide individualized feedback by analyzing previous data for similar student behavior, and present problems and feedback accordingly while minimizing the amount of expert involvement required for development.

This work hypothesizes that applying data-driven methods to the feedback generation, problem selection, and pedagogical strategy in an intelligent tutor will increase problem-solving performance. To test this hypothesis, the Deep Thought logic proof tutor was used in three studies which test the effectiveness of applying these data-driven methods. The addition of data-driven methods to Deep Thought has been shown to reduce student dropout, allowing more students to complete the tutor, and improve student performance in constructing logic proofs.

The end goal of this work is a fully data-driven intelligent tutoring system framework that is domain agnostic. Utilizing previous student data, this framework provides interval-evaluated mastery learning, individualized problem selection, and formative feedback and pedagogy in the form of hints and worked examples. A tutor in any new domain can be developed from this framework based only on an initial expert provided description of the domain and knowledge components, a corpus of problems, and a problem solving interface.

Improving Individualized Instruction in a Logic Tutor using Data-driven Methods

by

Behrooz Zakaria Mostafavi

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2016

APPROVED BY:

Dr. Kristy Boyer	Dr. Min Chi
Dr. Deniz Eseryel	Dr. Roger Azevedo

Dr. Tiffany Barnes
Chair of Advisory Committee

DEDICATION

Oobie!

BIOGRAPHY

Behrooz Zakaria Mostafavi was born in Stillwater, Oklahoma in late 1981 to first generation immigrants. After a typical American childhood full of DIY engineering projects and genetic research, Behrooz attended the North Carolina School of Science and Mathematics, where his boundless love of learning and occasionally dangerous curiosity was nourished. After spending many years of his undergraduate career at UNC-Chapel Hill exploring music, bio-medical engineering and his future spouse, Behrooz settled at UNC-Charlotte, receiving Bachelor's and Master's degrees in Computer Science with a focus on educational systems. He currently resides in Raleigh, North Carolina with Dr. Amanda Cohen Mostafavi. A lover of all things art, science and engineering, Behrooz hopes to continue enriching his life with new experiences and sharing his knowledge and experience with new generations of life-long learners.

ACKNOWLEDGEMENTS

Dr. Tiffany Barnes, thank you so much for your patience and intimidation. You motivate me in the way I need, and allow me to work the way I want.

Dr. Amanda Twyla Cohen Mostafavi, you are the first and last reason I am ever able to focus my scattered thoughts and structure them into something meaningful.

A big thanks

- to the experts and faculty that have worked with me while completing this work: Dr. Marvin Croy, Richard Ilson, Dr. John Stamper at UNC-Charlotte; my committee Dr. Roger Azevedo, Dr. Kristy Boyer, Dr. Min Chi, Dr. Deniz Eseryel; Dr. Jennifer Albert and Dr. Collin Lynch.
- to the graduate students who have worked with me on this project: Dr. Michael Eagle, Zhongxiu Aurora Liu, Dr. Matthew Johnson, Thomas Price, Stephen Schroeder, Shitian Shen, Guojing Zhou.
- to the undergraduate students who put a lot of time into development of Deep Thought: Robert Benson, Cameren Dolecheck, Kathleen Wassell.
- to the friends and lab members who have been supportive and distracting: Dr. Acey Boyce, Rachel Brinkman, Dr. Thomas Butkiewicz, Veronica Catete, Dr. Remco Chang, Dustin Culler, Dr. Wenwen Dou, Katelyn Doran, Rachael Eagle, Samantha Finkelstein, Alex Godwin, Dr. Frederick Heckel, Dr. Andrew Hicks, Richard Pallo, Barry Peddycord III, Shaun Pickford, Dr. Evie Powell, Emily Purdie, Dr. Evan Suma, Dr. Xiaoyu Wang, Dr. Caroline Ziemkiewicz.

Daddy, Mommy, Ana, Meredith, Mom, Dad, I love you.

This material is based on work supported by the National Science Foundation under Grants 1432156 and 0845997.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 Research Question

Can data-driven methods be used to create an intelligent tutoring system that:

1. improves tutor performance by selecting appropriate problems and feedback,
2. reduces student dropout and time needed to complete the tutor.

Individualized student instruction is one of the most effective educational strategies, and students who receive individualized instruction often perform significantly better on performance evaluations than students who receive classroom instruction [Ma14; Van11a; SHC14]. However, the lack of human instructors and tutors in proportion to the learning populace make individual human interaction infeasible in a large scale setting. Computer learning environments that mimic aspects of human tutors have also been highly successful. Intelligent tutoring systems (ITS) have been shown to be highly effective in improving performance when used in combination with classroom instruction through scaffolding of domain concepts and contextualized, individual feedback [Van05]. Additionally, the development of ITSs has enabled schools and universities to reach out and educate students who otherwise would be unable to take advantage of one-on-one tutoring due to cost and time constraints [Koe97].

However, the development costs of individualized computer-human instruction are high, averaging 100–300 hours of domain-expert time to construct one hour of course material [Mur99]. Much of ITS development time is spent building models of student knowledge and behavior. While some model-building techniques and their derivations (such as the use of Bayesian Knowledge Tracing [CA95]), and the use of authoring tools to construct intelligent tutors (such as the Cognitive Tutor Authoring Tools [Koe04]) can reduce the construction time to around 20–30 hours of development for one hour of instruction, time cost is still high. ITSs still require significant expert involvement to determine pedagogical policy; what to teach, and how to effectively teach it to the student.

Additionally, it is important for intelligent tutoring systems to present problems to students that are appropriate to the student's current level of proficiency in the subject matter. Students are more likely to perform better in-tutor when given problems in their zone of proximal development through scaffolding of major concepts [MA02; Van06]. Proper scaffolding increases performance, and therefore increases the likelihood of further progression through the tutor and exposure to higher-level concepts. However, in open-ended problem solving domains such as deductive logic and programming, there is no defined breakdown of knowledge components, and traditional knowledge tracing models are therefore difficult to apply without developing an entire cognitive model and increasing the amount of development time and expert involvement required.

1.2 Objective

1. To show how data-driven methods can be used to develop an adaptive tutoring system in an open-ended problem solving domain with minimal expert involvement

2. To show the effectiveness of data-driven methods in an individualized, adaptive tutoring system to improve tutor performance in an open-ended problem-solving domain.

Data-driven methods have been used to great effect to increase student performance, particularly through contextualized formative feedback (hint generation) and appropriate task selection [BS08; Koe13]. These methods leverage student data to directly inform tutor instructional decisions. This circumvents the need to model student knowledge, and allows data-driven tutors to adapt to focusing on supporting students engaged in incorrect, but frequently observed, student behaviors. Our data-driven approaches can be particularly effective in open-ended problem solving domains, such as deductive logic and programming. In these cases, creating an expert system to diagnose student behavior and prescribe adaptive feedback would require a great deal of effort, and we have no way to estimate the potential return on that time investment.

What is proposed here is a set of experiments to test the effectiveness of data-driven methods on reducing student dropout in a tutor and selecting appropriate problems for increased problem-solving performance. The tutor used in these experiments is a deductive logic proof building tool called Deep Thought. Previous experiments with Deep Thought showed that using data-driven hint generation reduced problem solving time and increased student retention in the tutor (see Chapter 3). In our initial experiments with data-driven mastery learning, we have shown that we can achieve even higher student performance and retention rates while reducing the average time in the tutor. The ultimate goal of this research is to create a fully data-driven intelligent tutor.

1.3 Hypotheses

H1

The data-driven, adaptive and individualized instructional methods presented for mastery learning, assessment, feedback generation, worked example selection, and problem selection will improve student performance, as measured by performance on rule scores, in an open-ended problem-solving domain.

H2

The data-driven methods presented will reduce student dropout in the Deep Thought tutor.

H3

The data-driven methods presented will reduce the time spent to complete the Deep Thought tutor.

1.4 Contributions

This dissertation presents the following research contributions:

1. An implemented and tested data-driven mastery learning system (DDML), that incorporates Bayesian knowledge tracing into a problem-solving environment that does not have a skill inventory. This research has shown that the DDML is effective in improving tutor completion and reducing student dropout.

2. An implemented and tested data-driven proficiency profiling (DDPP) system, that results in lower dropout, low time in tutor, and higher likelihood of high post-test scores than DDML.

DDPP, unlike DDML, automatically sets thresholds for rule application based on clusters of prior successful students.

3. Results from three studies to determine the impact of different instructional methods (specifically worked examples and/or on-demand hints) on student performance.

4. A fully data-driven intelligent tutoring system for logic that incorporates problem selection, worked examples, on-demand hints, and the automatic setting of thresholds for mastery.

The impact of this work on student problem-solving performance is broad, since logic is taught to all students in computer science and computer engineering at NC State and in many other programs. Deductive logic is an important fundamental topic in computer science education. Meyers [MJ90] outlined this necessity very clearly. He noted that almost every aspect of computer science required logic as a foundational basis in order to understand it. In 2003 a study conducted by Page [Pag03] confirmed the necessity of discrete math, including deductive logic, to computer science education. While all the research in this dissertation is based on logic proofs, the methods we have derived are domain agnostic, and should be applicable in other open-ended problem-solving STEM domains. I believe that our data-driven methods can be used to augment computer-assisted instruction with intelligence to adapt to individual students, potentially expanding the availability of effective individualized instruction to a larger audience. Where data driven methods do not offer significant increases in tutor performance, they still allow us to create an adaptive tutoring system on par with systems that rely on expert involvement.

CHAPTER

2

RELATED WORK

2.1 Tutoring Systems

Most intelligent tutors are structured as two loops; an outer loop that operates at the problem-level, such as selecting the problem or task the student should work on next, and an inner loop that governs step-level decisions during problem solving, such as the type and amount of guidance (i.e. the kind of hints and feedback) to give the student [Van06]. An effective tutor should adapt its instruction to student needs based upon their current knowledge level [And95]; however, in order to make instructional decisions, most existing ITSs either use fixed pedagogical policies providing little adaptability, or expert-authored pedagogical rules based on existing instructional practices [And95; Van06]. Data-driven approaches to making these decisions, in particular selecting problems, when to apply worked examples, and the type of hint or feedback to provide would reduce the cost involved in creating and improving the effectiveness of ITSs.

2.1.1 Meta-analyses of Intelligent Tutoring Systems

Several meta-analyses of current intelligent tutoring systems have been conducted, comparing them to human-based tutoring and various forms of classroom based instruction. While the analyses all found ITSs to be more effective than classroom based instruction, they disagree on their effectiveness

in relation to human-based tutoring; conclusions range from slightly less effective than human-based tutoring [SHC14] to as effective as human-based tutoring [Van11a; Ma14].

VanLehn [Van11a] conducted a meta analysis of a set of intelligent tutoring systems (based on step and sub-step granularity) and non-intelligent computer-based tutoring systems, and compared them to the effects of human tutoring. He found that step-based intelligent tutoring was currently effective at approximating the effects of human tutoring. Meanwhile Ma et al. [Ma14] conducted a meta analysis of 107 studies that compared the learning outcomes from intelligent tutoring systems, human-based tutoring, and other classroom-based instructional methods. Similarly, these authors also found intelligent tutoring systems were overall more effective than other instructional modes, aside from human tutoring (although they also found intelligent tutoring systems approximated human-based tutoring).

In 2014, Steenbergen-Hu and Cooper [SHC14] conducted a meta-analysis on the effect of 22 intelligent tutoring systems on the learning gains of college students in 39 studies. Unlike previous work, they found that using ITSs in an academic environment resulted in smaller gains than human tutoring, while still resulting in higher learning gains than classroom based methods. The authors additionally note the following conclusions for future ITS development: development of ITSs must focus on ill-defined domains, and there is at least an observed but non-robust observation in the meta-analysis that teacher involvement in the development of ITSs may affect the learning gains.

Deep Thought best approaches the concerns listed in the conclusions in Steenbergen-Hu and Cooper's work [SHC14]. Deep Thought focuses on an ill-defined domain (logic-proof construction) and, as a data-driven intelligent tutor, seeks to mitigate the need for teacher involvement to increase its effectiveness by using data-driven methods. Steenbergen-Hu and Cooper [SHC14] also note that there is a marked increase in effectiveness when ITSs are used in a college setting in comparison to other instructional methods, which implies that the positive gains made in Deep Thought should be more pronounced than otherwise.

2.1.2 Logic Tutors

Several computer systems have been built for instruction in solving logic proofs. However, many of these tools simply teach logic without any sort of adaptability. LogEx [Lod], for example, is a learning environment for propositional logic that logs student data, but does not currently use that data to enhance the tutor. Sequent Calculus Trainer [Ehl], on the other hand, incorporates feedback only in the form of information presented on how to use each logic rule or an error if a rule is applied incorrectly, features that were present in the earliest version of Deep Thought.

Of greater interest is work that uses data-driven methods incorporated into an intelligent tutor to

improve its effectiveness. Logic is of particular interest because it is an open-ended problem solving domain that computer science students must learn [MJ90; Pag03]. To that end, there are several logic tutors that incorporate data-driven methods, mainly to improve feedback. Logic-ITA [MY05] provides students with performance feedback upon proof completion, but not much feedback beyond that. The authors performed association rule mining, clustering, and classification on the data gathered to determine indicators for students completing the tutor successfully, such as how many problems the student was able to successfully solve, number of mistakes made (as well as what kind of mistakes), number of rules used, and number of steps taken. They later implemented proactive hints based on previous student behavior that the instructor could apply or alter. CPT [Sie07] uses an automated proof generator to provide contextual hints. The proof generator specifically uses interaction calculus to automatically find a completed proof based on a set of premises. This search algorithm is used to generate hints for students based on the best possible path to completion from the current set of premises in that step of the problem. PLT [Luk02] uses a combination of dynamically generated hints and more general examples for rule applications and axiom generation based on previous student data. However, the authors determined that based on student responses, the tutor setup and design prevented students from gaining as much as they could have from a more effectively designed and user-friendly system. More recently, Gluz et al. implemented a dynamic and adaptive natural deduction proof tutor called Heraclito based on a dynamic student model that can automatically solve problems using strategies employed by instructors [Glu14]. The authors argue that the Heraclito system is fully adaptive, conforming to the way students reason and solve logic proofs.

The Deep Thought logic tutor detailed in Chapter 3 has been augmented for adaptive learning via hints and intelligent problem selection using data-driven methods. Unlike the previously listed tutors, which incorporate specific feedback generation methods, Deep Thought has been developed to flexibly provide students with a combination of hints and worked examples. Additionally, while the previous logic tutors use data-based or data-driven methods to improve feedback to students, they do not make any changes to the difficulty or type of problem presented to the students. Deep Thought uses a data-driven system to select the best next set of problems for students to solve.

2.1.3 Knowledge Tracing

On-going work in reducing tutor development time and improving problem-level decision making has focused on developing and using models based on previous and current student behavior to predict learning outcomes and leverage those predictions to select an appropriate subsequent problem or task.

Cognitive tutors use knowledge tracing (KT) to track students' mastery of particular Knowledge Components (KCs) based on their performance at each opportunity to apply that KC. A knowledge component is a structure or process that a learner uses, alone or in combination with other knowledge components, to accomplish steps in a task or problem [Van06]. Knowledge components in a tutoring system can range from domain concepts or operators, such as addition or multiplication in algebra, to elements of student behavior, such as the use of in-tutor references and time on-task.

The classic model used by Corbett and Anderson uses Bayesian Knowledge Tracing (BKT) to predict the probability of learned components based on learning, guess, and slip parameters [CA95]. Gong et al. [Gon10] compared several fitting methods for the BKT model, namely the expectation-maximization model (parameters are calculated based on maximizing the likelihood of student data) and the "brute force" method (ideal parameters are searched for within a given range). KT models with these fitting methods applied were then compared to performance factor analysis and found to be comparable in terms of accuracy and model complexity.

Once a KT model is created, it can be used to select problems which best suit the student and what he or she still needs to learn. BKT has been shown to be generally effective for predicting success on subsequent problems (meaning it can be used for problem selection), although it is more effective in domains with fewer and more well defined knowledge components, such as algebra [Xu12]. There have been attempts to improve and expand on the BKT model, such as incorporating item and problem difficulty by Pardos and Heffernan [PH11]. Pardos et al. used multiple models in ensemble-based classification algorithms to improve student knowledge prediction. Their results showed significant improvement in learning model prediction accuracy [Par11].

However, BKT is not perfect. It gets time and computationally expensive when there are more features (KCs) to analyze, and its reliance on calculating conditional probabilities is a weakness [Hua14]. In addition, in tutoring systems with an intelligent problem selection component, problem selection thrashing can occur. Thrashing is a cycle that can occur in problem selection, where students will be given a string of problems that focus on concepts they have already mastered until they reach a problem with multiple concepts, including concepts they have not yet mastered, inevitably get the incorrect answer, and then are given a similar string of easy but unhelpful problems. Koedinger et al. argued that thrashing was caused by incorrect blame assignment in knowledge tracing systems; when a student attempts a problem with multiple knowledge components and gets an incorrect answer, knowledge tracing works on the assumption that each knowledge component must have been applied or executed equally incorrectly [Koe11]. This can result in greater problems in domains where knowledge components may overlap or conflict with each other, such as deductive logic proof problem-solving. Aleven et al. provided an overview of the uses and effectiveness of knowledge component based modeling in intelligent tutoring systems, as well as well as current challenges. In

particular, the authors focus on the challenge of determining the knowledge components acquired in any given domain [AK13].

The data-driven mastery learning (DDML) system presented in Chapter 4 was designed to avoid problem selection trashing, in that we sought to intelligently select appropriate problem sets by determining which knowledge components each student needed to focus on and use that determination to select appropriate problems. The DDML system uses historical student data to provide thresholds for target-rule KT score comparison to current students, and weights the importance of those rules strictly based on expert-decided priority for the level. In other words, a behavioral analysis is performed on the performance of successful students, broken down by their application of available rules in the system, and the intended rules the problems were designed to use. The performance of these successful students provides a benchmark for new students in the tutor. Snow et al. showed that students who were more controlled in their behavior exhibited much higher strategy performance than their less controlled counterparts [Sno14]. The DDML system accomplishes this through strict ordering of selected problems. In the DDPP system presented in Chapter 6, the problem selection bypasses the need for expert-decided KC priorities by clustering the actions taken by students who had previously completed the tutor to determine problem solving strategies and determining the most important rules for each strategy.

2.1.4 Hints and Hint Generation

Providing hints to students in the course of an intelligent tutor as a possible form of step-based feedback has the potential to increase learning gains. Razzaq, Leena, and Heffernan [RH10] compare the effects of proactively providing hints to students and allowing them to seek hints themselves. They found that learning gains increased for students given on-demand hints in comparison to students who were provided hints proactively. Additionally Mitrovic et al. [Mit13] implemented a system to provide positive feedback in a SQL language tutor and compared it to the previous system they had, SQL-Tutor, which only provided feedback when a student committed an error. The authors in this case similarly found an improvement when incorporating positive feedback: students were able to learn new constraints much faster when compared to the control group, which only received negative feedback.

This research contains studies that compare performance results from problems presented both with and without on-demand hints. In Deep Thought, the hint system used is called Hint Factory. Hint Factory is an automatic data-driven hint generator that converts an interaction network of student trace behavior into a Markov decision process (MDP) to automatically select on-demand hints for students upon request, based on their individual performance on specific problems. The

MDP is data-driven, using actions logs from previous Deep Thought use in the classroom to assign weight to proof-state actions based on whether or not that action ultimately led to successful completion of the proof. In this way Hint Factory provides contextualized hints. These hints help students solve problems by suggesting what step should be taken next on a multi-step problem. Hint Factory has been implemented in the Deep Thought logic tutor to automatically deliver context-specific hints to students during problem-solving [BS08]. While help-seeking behaviors can be unproductive [Ale04], hint use is not limited in Deep Thought, particularly since bottoming out hints can be seen as searching for a worked example, an effective learning strategy [Shi08].

In a previous study Hint Factory was shown to provide context-specific hints over 80% of the time [Bar08]. In a pilot study, Barnes & Stamper found that Hint Factory can provide sufficient, correct, and appropriate hints for the Deep Thought Logic tutor and help students to solve more logic proof problems in the same span of time [BS08]. The Hint Factory can evolve over time, providing some automatically generated hints initially and improving as additional expert and student problem attempts are added to the model.

2.1.5 Worked Examples

Numerous studies have shown the benefits of combining worked examples with problem solving as compared to problem solving alone [Atk00; SC85]. However since there is little empirical evidence to show how worked examples and problem solving should be combined, most existing tutoring systems exclusively choose problem solving [And95; Van05]. Najar and Mitrovic [NM12] presented a more modern review of previous research in using worked examples in intelligent tutoring systems. Their conclusion was that the research was still inconclusive, and raised several questions on when to give examples, when to scaffold, and how to design examples. However Hilbert and Renkl [HR09] found that improved learning outcomes occurred when providing worked examples with a prompt, and proposed that this was due to allowing the students to have a greater cognitive load at once. Sweller and Cooper [SC85] came to similar conclusions when they found that students who were given a combination of unguided problem solving and worked examples learned faster and demonstrated better performance. However, students who were given worked examples received more information than students solving problems without aid, which could explain the improvement.

There has also been research in comparing worked examples with existing intelligent tutoring systems. McLaren, Kim, and Koedinger [McL08] found that replacing selected problems with worked examples does not increase learning gains over tutoring alone. However students had higher learning efficiency and achieved learning gains much more quickly. Similar results were found in subsequent studies. McLaren and Isotani [MI11] compared three tutoring using all worked exam-

ples, all traditional unguided problem solving, and a mix of worked examples and problem solving. Each group achieved similar learning gains, but the students who were given all worked examples required less time to achieve those gains. McLaren et al. later conducted a similar study comparing the results from giving students worked examples, erroneous examples, tutored problem solving via an ITS, and traditional unguided problem solving [McL14]. Again, students who were given worked examples had higher learning efficiency.

Based on the current literature, worked examples have the potential to improve learning gains and increase how quickly students achieve these gains when combined with an intelligent tutor. The studies planned for Fall 2014 and Spring 2015 (see Chapters 5 and 6) are being used to further explore the effects of worked examples, and use the results to inform pedagogical policy regarding intervention with worked examples. In using Deep Thought with Hint Factory, we have a unique opportunity to explore the use of bottom-out hints as worked examples, since the method that Hint Factory presents requested hints will lead to a worked example for that problem step.

2.2 Data-Driven Methods

The use of data-driven methods to develop intelligent tutoring systems is just starting to be explored in the field. Koedinger et al. [Koe13] gave a very detailed overview on developing data-driven intelligent tutoring systems, in particular techniques for incorporating data in a useful way. They first summarized the general process of intelligent tutoring systems; the system selects an activity for the student, evaluates each student action, suggest a course of action (either via hints, worked examples, or another form of feedback), and finally updates its own internal evaluation of the student's skills (depending on how this information is represented and updated, e.g. Knowledge Tracing). This is similar to the outer and inner loop of most intelligent tutoring systems described by VanLehn [Van06], with the addition of an evaluation and reevaluation of student skills. With this process and the functions required in mind, the authors then present several machine learning approaches for using data to develop intelligent tutoring systems (particularly SimStudent and Hint Factory). The authors concluded by discussing various data-driven ways to improve the effectiveness of intelligent tutoring systems. They discussed optimizing the cognitive model using learning factors analysis; fitting statistical models to individual students; developing a model of student mood and engagement by modeling off-task behaviors, careless errors, and mood; and improving how the tutor selects actions for the student via Markov Decision Processes (MDP) or Partially Observable Markov Decision Processes (POMDP). In a later work Koedinger et al. [Koe14] compared and contrasted current data-driven methods for intelligent tutoring and discussed the potential for these methods to improve MOOCs. They went over the success cases for using data to improve tutors and

coursework, in particular cognitive task analysis.

An early example of a data-driven intelligent tutor is the Cognitive Algebra Tutor, first introduced by Ritter and Steven [Rit07]. Here the authors introduce an algebra tutor which is fully driven by historical data, interpreted based on a cognitive theory. Student behavior is modeled based on the cognitive theory ACT-R and student data was gathered from several previous studies. This was several years and studies into development at this time, and the result is an example of a fully data-driven tutor that improved student performance. The authors noted that although it over-predicted student performance, it would be improved the more data was collected.

There have also been several recent studies that demonstrate the potential for data driven methods to result in more effective tutors that accurately predict student behavior. Lee and Brunskill [LB12] examined the benefits and drawbacks to basing model parameters on existing data from individual students in comparison to data from an entire population, specifically as it pertained to the number of practice opportunities a student would require (estimated) to master a skill. The authors estimated that using individualized parameters would reduce the number of practice opportunities a student would need to master a skill. González-Brenes and Mostow [GBM13] demonstrated a data-driven model which automatically generates a cognitive and learning model based on previous student data in order to discover what skills students learn at any given time and when they use skills they have learned. They used a topical Hidden Markov Model to create a cognitive model and a student model for a set of students using an algebra tutor. The resulting model predicted student behavior without the aid of previous domain knowledge and performed comparably to a published model.

Data-driven intelligent tutors not only have the potential to more accurately predict student behavior, but interpret why it occurs. For instance, Elmadani, Mathews, and Mitrovic [Elm12] proposed using data-driven techniques to detect student errors that occur due to genuine misunderstanding of the concepts (misconception detection). They processed their data using FP-Growth in order to build a set of frequent itemsets which represented the possible misconceptions students could make. The authors were able to detect several misconceptions based on the resulting itemsets of student actions. Fancsali [Fan14] used data-driven methods to detect behaviors that usually detract from a student's experience with an ITS (off-task behavior, gaming the system, etc).

Overall the literature shows that incorporating data-driven methods has a great deal of potential to accurately adapt to and predict student behavior, and increase learning gains. However the focus has been on using data to develop or determine a cognitive model, and predict student behavior based on how it fits with the model. Deep Thought does not attempt to develop a cognitive model, but a student model that is fully based on previous student behavior in the tutor.

CHAPTER

3

THE DEEP THOUGHT LOGIC TUTOR

The Deep Thought logic tutor is central to the work presented in this thesis, forming both part of the motivation, and a framework for implementation and testing. This chapter presents a description of problem space and the time-line and significance of Deep Thought's development towards a fully data-driven intelligent tutor.

3.1 Problem Space

The problem space for the construction of logic proofs in Deep Thought is open-ended, without any limitations on the length or quality of a proof's solution. There are 59 possible proof problems within the 7 different versions of the tutor, with each problem giving between 2 and 4 logical expressions as premises, in order to reach a single logical expression as a conclusion. While no individual problem in Deep Thought *requires* more than 10 steps (applications of logical axioms on one or two expressions in the problem space) in order to reach the problem's conclusion, the actual length of problem solutions by students in practice ranges from the minimum length of a problem's solution to over two dozen steps in practice, not counting expressions deleted by the user mid-proof. When considering that students can delete steps and continue forward within a given proof, the number of steps taken by students in solving proofs has exceeded 100 steps in extreme cases.

This is possible because of the mechanisms provided to the students to solve proofs. Students are given 19 different rules (axioms) that can be applied to logic expressions in Deep Thought, with 8 of these rules being logical implication (inference) rules, 5 of these rules being logical equivalence (replacement) rules, and 6 of these rules being logical binary operation rules. Logical implication rules (*Modus Ponens*, Disjunctive Syllogism, Simplification, Hypothetical Syllogism, *Modus Tollens*, Addition, Conjunction, Constructive Dilemma) are one-way operations, where a new logical expression can be extracted or constructed from a given premise, without an equivalent method of returning to the original expression. Logical equivalence rules (Double Negation, Implication, Equivalence, DeMorgan's, Contrapositive) and logical binary operation rules (Commutative, Associative, Distributive, Absorption, Exportation, Tautology) are two-way operations, where a new logical expression resulting from a transformation of a given premise can be transformed back to its original premise using an equivalent operation (The descriptions of these rules are presented in Appendix A). In addition, students have the ability to delete leaf expressions in the course of solving proofs, including expressions resulting from logical implication operations.

This results in an *open-ended* problem space for any given proof, giving students a theoretically limitless number of possible steps to reach the proof's conclusion, despite the bounds given by the proof problem's premises (the number of premises, the premises' logical variables, and the complexity of the premises' expressions). In practice, however, and excluding extreme outliers, the variation in student solutions is a limited state space [EB14a; Eag14], but these state spaces for Deep Thought's logic proof problem are procedurally unpredictable for any given set of students. It is for these reasons that data-driven methods are inherently better for generating hint and feedback systems for domains such as deductive logic, where a mass of prior student solutions for a proof are used to determine the best next-steps for a new student entering the system. This forms the basis of the incremental changes made to Deep Thought.

3.2 Deep Thought Versions

3.2.1 Deep Thought 0 (DT0)

Inception

Dr. Marvin Croy, a Philosophy professor teaching undergraduate deductive logic at the University of North Carolina at Charlotte, developed Deep Thought version 0 (also referred to as DT0) into a Java Web-Applet in 2000 as a visual-based practice tool for logic proof construction and data collection. The proofs presented in Deep Thought were divided into three sets of problems. Croy designed each set of problems to require different logical rules and problem solving skills for problem comple-

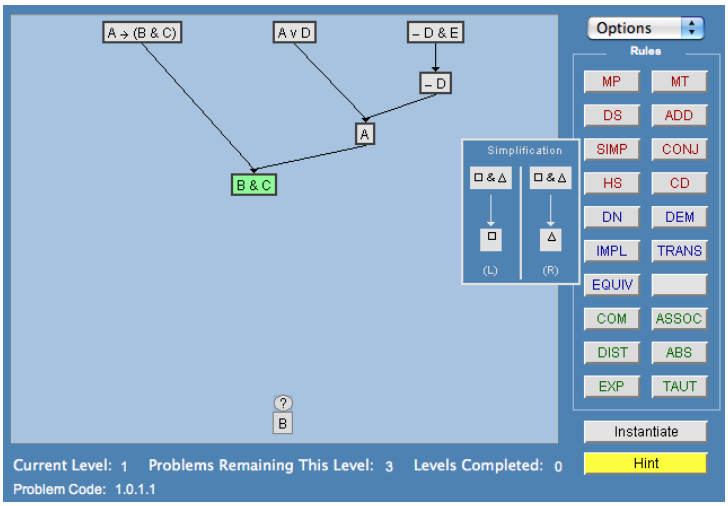

Figure 3.1 A screen capture of the Deep Thought tutor, showing given premises at the top, conclusion at the bottom, and rules for application on the right. The inset window is an example of a rule-axiom reference that appears when a student hovers a mouse pointer over a rule button.

tion, and designed the entire set of problems in Deep Thought to allow students to demonstrate satisfactory knowledge of proof problem solving at an introductory undergraduate level.

User Interaction

Figure 3.1 shows the interface for Deep Thought, where students construct logic proofs by connecting displayed logical premises (seen at the top of the figure window) using buttons for logical rules (seen at the right of the figure) to derive a conclusion (seen at the bottom of the figure window). For example, students solving the proof in Figure 3.1 use premises $A \rightarrow (B \wedge C)$; $A \vee D$; and $\neg D \wedge E$ to derive conclusion B. An example solution to this proof is shown in Table 3.1. Each of the logic rule buttons has a rule-reference in graphical representation that could be seen by hovering the mouse pointer over the selected rule button for two seconds (see inset window in Figure 3.1). As a student worked through a problem, each step was logged in a data file that recorded the current problem, the rule being applied, any errors made (such as attempting to use a rule that was logically impossible), completion of the problem, time taken per step, and elapsed time taken to solve the problem.

Problem selection and window options in DT0 were accomplished through a drop-down window. Students could select the representation of the logical operators in proof expressions (and; or; not; implication; equivalence): either in English mode (as "and"; "or"; "not"; "if...then"; "iff"); or in symbolic mode (as "&"; "v"; ["~"/"-"]; ["⊃"/"→"]; "="). Students had the option of working forwards

(top down), by selecting one or two expression nodes and applying a rule to derive a new expression, or by working backwards (bottom up) by selecting the "?" button above a conclusion block to create rule-axiom-structured parent blocks, that then must have their unknown variables instantiated [Cro00].

Table 3.1 A solution to the proof shown in Figure 3.1.

#	***Premise***	***Derivation***
1	$A \rightarrow (B \wedge C)$	Given
2	$A \vee D$	Given
3	$\neg D \wedge E$	Given
4	$\neg D$	3/Simplification
5	A	2,4/Disjunctive Syllogism
6	$B \wedge C$	1,5/*Modus Ponens*
7	B	6/Simplification

Problem Set

The proofs presented in Deep Thought were divided into three sets of problems. The first set contained problems requiring basic logical implication rules (e.g. *Modus Ponens*, Simplification), the second set contained problems expanding the rule requirements to include more advanced logical implication rules (e.g. Hypothetical Syllogism, Constructive Dilemma), and the third set contained problems that required logical equivalence rules in addition to implication rules (e.g. DeMorgan's, Contrapositive). In DT0 students were allowed to solve assigned problems at will, in any order, though assignments were ordered. There were a total of thirteen available problems in the tutor for the duration of the studies referenced in this thesis.

Problem-solving Feedback

Feedback in DT0 was limited to alerts upon mistakes made during rule application, either by attempting an action incorrectly (such as applying a rule before first selecting expression blocks), or by not applying the correct action given the current state (such as attempting to apply a logically incorrect rule).

Improvements to Deep Thought

Croy envisioned Deep Thought as a system that could identify at-risk students early in a course curriculum, and give individualized feedback to aid those students [Cro08]. This motivated the following updates and changes to Deep Thought.

3.2.2 Deep Thought 1 (DT1-NSH)

Inception

Barnes & Stamper extended Deep Thought with a data-driven hint-generation system called Hint Factory to create Deep Thought 1 (also referred to as DT1-NSH, for Next-Step Hints) between 2006 and 2008 [BS08]. Hint Factory used a Markov decision process (MDP) to automatically select on-demand hints for students upon request, based on their individual performance on specific problems. The MDP was data-driven, using actions logs from previous Deep Thought use in the classroom to assign weight to proof-state actions based on whether or not that action ultimately led to successful completion of the proof.

User Interaction

Aside from the addition of the "Hint" button to the Deep Thought interface, DT1-NSH user interaction was identical to user interaction in DT0. The Hint button was only usable if a hint file existed for the current problem. There were a total of eight (out of thirteen total) problems that had available hints.

Problem Set

The problem set for DT1-NSH was identical to the problem set from DT0.

Problem-solving Feedback

In addition to the same rule-application errors as DT0, when hints were available for a given proof problem, clicking the "Hint" button with the mouse pointer would result in one of four possible feedback responses for any individual proof-state, displayed on successive "Hint" button presses for that same step. The student is given a broader hint when they first request one, and given more direct hints the more they ask. The responses for the proof-state in Figure 3.1 are shown below in Table 3.2:

Table 3.2 A generated on-demand hint sequence for the proof-state in Figure 3.1

Hint #	Hint Text	Hint Type
1	Try to derive "B" working forward	Indicate goal expression
2	Highlight "B & C" to derive it	Indicate premises to select
3	Click on the rule Simplification (SIMP)	Indicate the rule to use
4	Highlight "B & C" and click on Simplification (SIMP) to derive "B"	Bottom-out hint

Findings

Barnes & Stamper showed that with past performance data, Hint Factory was able to provide a hint when requested by a student 72% of the time on average using one semester of performance data, 79% when using two semesters, and 82% of the time when using three semesters of performance data [Bar08]. Eagle, Barnes, and Stamper showed these hints were effective in reducing the time taken to complete the tutor by 55%, and increasing retention in the tutor by 300% [EB14c; EB14b].

3.2.3 Deep Thought 2 (DT2-MP)

Inception

In their research, Stamper et al. observed student dropout in DT0 and DT1-NSH were high [Sta11]. Despite improvements to the student dropout rate from DT0 to DT1-NSH with the addition of hints, there were still significant numbers of students that did not complete the tutor. To address this issue, I have developed a novel data-driven mastery learning system (DDML) to improve learning in open-ended problem solving domains such as deductive logic. The DDML system was integrated into DT0 in 2013 as Deep Thought 2 (also referred to as DT2-MP, for Manual Proficiency). It was hypothesized that the DDML system developed for DT2-MP would result in higher percentage completion and lower student dropout rates than DT0 and DT1-NSH.

User Interaction

The user interaction of DT2-MP was identical to DT0, except that instead of allowing students to select a proof problem from a list in the "Options" menu, DT2-MP selected problems based on that student's measured proof-solving proficiency. Students were allowed to skip a tutor-selected problem using the drop-down menu; the problems populated in the drop-down menu were dependent on that student's current progression through the tutor.

Problem Set

DT2-MP divided the problem sets of DT0/DT1-NSH into more levels. Each of the three sets of problems in DT0/DT1-NSH was split into two sets in DT2-MP to allow assessment of student rule application at regular intervals. DT2-MP Level 1 & 2 problems required basic logical implication rules, Level 3 & 4 problems expanded the rule requirements to include more advanced logical implication rules, and Level 5 & 6 problems required logical equivalence rules in addition to implication rules.

Problem-solving Feedback

Since the DDML system was a hidden process, the feedback for DT2-MP was the same as the feedback from DT0. No additional hints or problem-solving feedback were provided so that the DDML could accurately assess student proficiency without the proficiency being affected by additional feedback.

Findings

Results show that the DDML system increased mastery of target tutor-actions, improved tutor scores, and lowered the rate of tutor dropout over DT0 or DT1-NSH. The design, implementation, and results of using DT2-MP are detailed in Chapter 4.

3.2.4 Deep Thought 3 (DT3)

Inception

Deep Thought was created when user-interface development capabilities were limited, and in 2011 Web-Applets were removed from Java web support. For both these reasons, I have re-designed Deep Thought and have led a ground-up re-build between 2013 and 2014. In addition to expanding the Deep Thought window and removing menus to improve accessibility of options and relevant information, the re-design includes the ability to develop and manage system features based on course and research requirements, and provides comprehensive step-level logs. Deep Thought 3 (also referred to as DT3) is designed to be a modular instructional and research tool, where features or modules can be included or excluded for particular classroom or research settings.

User Interaction

DT3's new features included a traditional text-based list of proof steps that reflected the proof state in the the graphical window, the ability to work a problem indirectly by negating the conclusion and searching for a logical contradiction, and more detailed feedback. DT3 also included support for

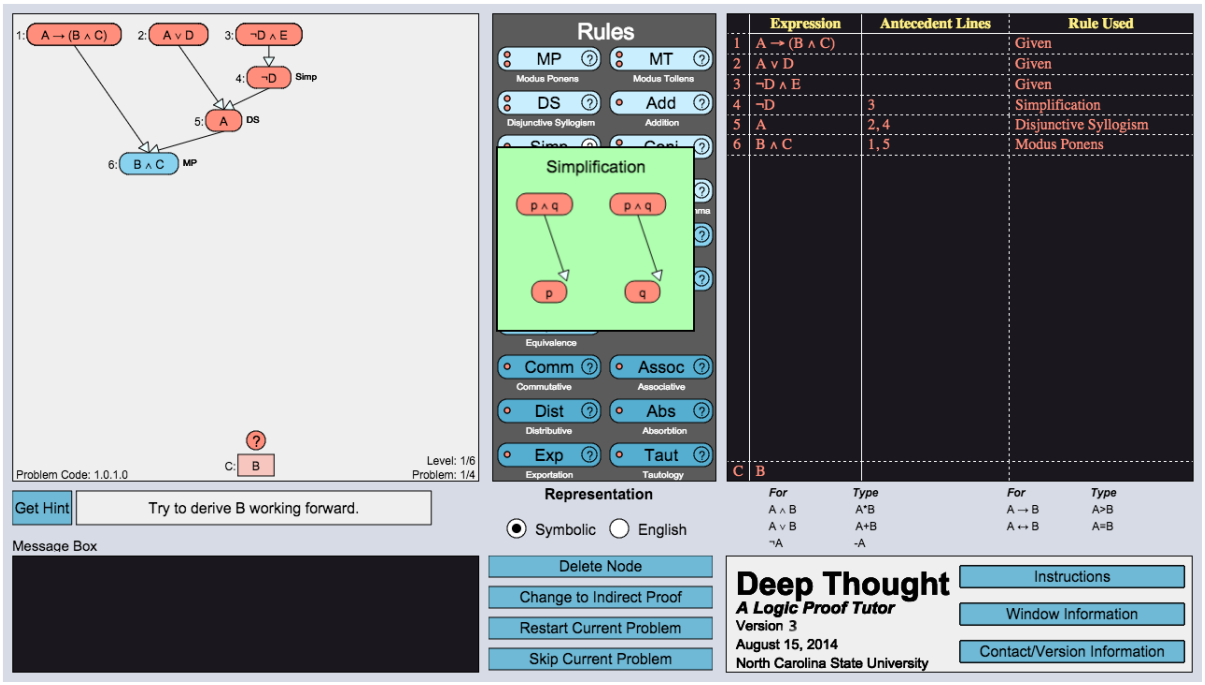

Figure 3.2 A screen capture of the Deep Thought tutor redesign, mirroring the proof-state in Figure 3.1.

hints while students work backwards, which was facilitated by a complete redesign of the data logs and hint functionality. DT3 was also fully usable with mobile-web devices.

Problem Set

DT3 used the same DDML problem set structure as DT2-MP. DT3 randomly presented some problems as worked examples based on course options. In these cases, the existing problem set was extended to ensure a balance between problem solving and worked examples.

Problem-solving Feedback

DT3 had an updated DDML system that allowed hints, and a hint policy that selected hint types based on instructor parameters. The hypothesis after testing and evaluating DT2-MP was that the addition of hint systems would significantly improve student retention and learning gains. DT3's hint policy could present problems with no hints, next-step hints, high-level hints, or worked examples.

Findings

In Fall 2014, DT3 was tested in two configurations: the first is the same no-hint system as DT2-MP (DT3-NoHint) with additional rule-highlighting hints; the second is a policy that chooses between presenting a problem with no hints, presenting a problem with Hint Factory style next-step hints, and presenting a worked example of the problem (DT3-Random). Results show that the addition of hint systems and random worked examples to the DDML increased mastery of target tutor-actions, improved tutor scores, and lowered the rate of tutor dropout over DT2-MP. The design, implementation, and results of using DT3 are detailed in Chapter 5.

3.2.5 Deep Thought 4 (DT4)

Inception

DT4 served mainly as an extension of DT3 for additional data collection, as well as gathering data on student behavior for new features to be studied in depth at a later time.

In an earlier study, Mostafavi & Barnes used DT1-NSH as a testbed for the automatic generation of proof problems to determine if parameterized backwards construction of proof problems could match conceptual and difficulty levels of expert-authored problems [MB11]. LQGen, a logic question generator that automatically generates logic problems based on a set of criteria (number of premises, number of steps, re-used premises, logic rules, difficulty, and conclusion), was used to create a set of logic proof problems to use in DT1-NSH. These generated problems were then tested in a philosophy deductive logic course in 2010-2011 at the University of North Carolina at Charlotte. 80 students in the Fall 2010 class solved the original expert authored problems in DT1-NSH, while 80 other students in the Spring 2011 class solved the automatically generated problems. The students who were given the automatically generated problems to solve performed as well as students who solved the expert-authored problems; the number of attempts and percentage of failed attempts were comparable between the students solving expert-authored problems and the students solving the automatically generated problems, while students given the automatically generated problems took less time to solve the problems overall. While the automatic generation of proof problems was not part of DT4, the exploration of data-driven pedagogical strategies in DT4 could inform problem parameter classification, therefore leading to data-driven automatic problem generation at a later date.

User Interaction

DT4 user interaction was the same as user interaction in DT3, with the addition of step-decision self-explanation added to certain steps when presented with worked examples.

Problem Set

The problem set for DT4 was identical to the problem set from DT3.

Problem-solving Feedback

In addition to the hint types from DT3, high-level (sub-goal) hints were available for use by the hint policy.

Findings

In Spring 2015, DT4 was tested in four configurations: the first is the same system as DT3-NoHint (DT4-NoHint); the second is the same policy that chooses between presenting a problem with no hints, presenting a problem with next-step hints, and presenting a worked example of the problem (DT4-Random); the third is the same policy as DT4-Random with the addition of step-decision self-explanation when presented with worked examples (DT4-SE); the fourth is the same policy as DT4-Random with the use of high-level hints instead of next-step hints (DT4-HLH). The purpose of this study was two-fold; extending the study of worked examples from DT3, and gathering new data about the behavior of students using high-level hints and self-explanation steps for a future study. Our results show that students using Deep Thought with worked examples completed more of the assigned problems and were less likely to drop out. Additionally, the students given worked examples were significantly more likely to finish the tutor. The results from the additional data collected on worked examples are detailed in Chapter 5.

3.2.6 Deep Thought 5 (DT5)

Inception

For the objective of realizing a fully data-driven system, Deep Thought 5 (also referred to as DT5) was created by extending DT3 and DT4 to replace the expert authored proficiency calculation with a data-driven proficiency profiler (DDPP). This system classifies a student's proficiency based on comparisons to all actions taken by exemplars from DT2-MP and DT3-Random, and automatically sets rule priorities and thresholds in the DDML between levels.

User Interaction

DT5 user interaction is the same as user interaction in all versions of Deep Thought since DT3.

Problem Set

The problem set for DT5 is identical to the problem set from DT3 and DT4.

Problem-solving Feedback

The hint types in DT5 are the same as in all versions of Deep Thought since DT3 (next-step hints, high-level hints, and worked examples). A new feature is available that automatically gives a next-step or high-level hint after a student works on a problem for longer than the median problem time demonstrated by previous students solving the same problem in Deep Thought.

Findings

In Fall 2015, DT5 was tested in seven configurations:

- Condition 1 (DT5-NSH) is the same system as DT3-NoHint with the addition of next-step hints, and serves as a control.
- Condition 2 (DT5-Random) is the same system as DT3-Random, and serves as a secondary control.
- Condition 3 (DT5-RL-Base) is the same system as DT5-Random, but using a baseline reinforcement learning pedagogical policy to decide whether to provide a worked example or problem-solving opportunity based on data from DT3 and DT4. This condition will be used for separate unrelated studies.
- Condition 4 (DT5-RL-LPSI) is the same system as DT5-RL-Base, but using a custom reinforcement learning pedagogical policy instead of the baseline policy. This condition will be used for separate unrelated studies.
- Condition 5 (DT5-Random-NSH-tr) is the same system as DT5-Random, but automatically gives a next-step hint (triggered hint) after a student works on a problem for longer than the median problem time demonstrated by previous students solving the same problem. This condition will be used for separate unrelated studies.

- Condition 6 (DT5-Random-HLH-tr) is the same system as DT5-Random-NSH-tr, but with high-level hints presented instead of next-step hints. This condition will be used for separate unrelated studies.
- Condition 7 (DT5-DDPP) is the same system as DT5-Random, but with the data-driven proficiency profiler (DDPP), which replaces the expert-decided rule priorities and thresholds in the DDML between levels.

The DDPP was first tested on historical data collected from DT2-MP and DT3-NoHint. The results were first compared to the original student proficiency calculation in DT3, as well as student proficiency calculated based on average and minimum rule scores, showing that the DDPP could potentially predict student proficiency at the same level as the expert-authored DDML with additional data. It was then implemented and tested in a classroom setting. In terms of the DDPP's performance, the results were not statistically significant between the DDPP group and the DDML groups. However, the trends in student dropout and rule application accuracy indicated that the DDPP performed at least as well as the DDML in guiding students through Deep Thought. The design, implementation, and results of testing and using the DDPP in DT5-DDPP are detailed in Chapter 6.

3.2.7 Deep Thought 6 (DT6)

Inception

DT6 served mainly as an extension of DT5 for additional data collection, as well as gathering data on student behavior for new features to be studied in depth at a later time.

User Interaction

DT6 user interaction is the same as user interaction in all versions of Deep Thought since DT3.

Problem Set

The problem set for DT6 is identical to the problem set from DT5.

Problem-solving Feedback

The hint types in DT6 are the same as in all versions of Deep Thought since DT3 (next-step hints, high-level hints, and worked examples). The same triggered hint feature from DT5 is available for certain test conditions.

Findings

In Spring 2016, DT6 was tested in six configurations:

- Condition 1 (DT6-Random) is the same system as DT3-Random, and serves as a control.
- Condition 2 (DT6-DDPP) is the same system as DT5-DDPP.
- Condition 3 (DT6-RL-Immediate) is the same system as DT6-Random, but using an immediate reward reinforcement learning pedagogical policy to decide whether to provide a worked example or problem-solving opportunity based on data from DT5. This condition will be used for separate unrelated studies.
- Condition 4 (DT6-RL-Delay) is the same system as DT6-Random, but using a delayed reward reinforcement learning pedagogical policy to decide whether to provide a worked example or problem-solving opportunity based on data from DT5. This condition will be used for separate unrelated studies.
- Condition 5 (DT6-Random-NSH-tr) is the same system as DT5-Random-NSH-tr. This condition will be used for separate unrelated studies.
- Condition 6 (DT6-Random-HLH-tr) is the same system as DT5-Random-HLH-tr. This condition will be used for separate unrelated studies.

The results from the additional data collected on worked examples and the DDPP are detailed in Chapter 7.

CHAPTER

4

STUDY ONE - DATA DRIVEN MASTERY LEARNING

Towards Data-driven Mastery Learning

4.1 Research Question

Can the tutor completion and dropout rate from DT0 and DT1-NSH be improved using data-driven methods of assessment and problem selection?

4.2 Objective

To create and implement a procedure for the systematic assessment of student performance and assignment of problems that match the student's preparation and provide practice on still-needed skills.

This chapter describes a novel data-driven mastery learning system (DDML) that uses the knowledge tracing (KT) of tutor actions in past student-tutor performance data to regularly evaluate new stu-

dent performance and select successive structured problem sets. An overview of the DDML system is shown in Figures 4.1 & 4.2. Problem sets are separated into *levels* based on the rules or concepts required for those problems, with the order of levels determined by the course curriculum. Each level presents an opportunity for *mastery learning*, requiring students to complete of a minimum number of problems containing the required rules or concepts for that level in order to progress to the next. Between levels, student performance is assessed by the knowledge tracing of all *actions* taken in the tutor (in the case of this experiment, *target-rule application*), weighted by the *priority* of those actions within the level. The resulting score from the assessment is compared to a *threshold* value: the average data-driven knowledge tracing (DDKT) scores of corresponding problems solved by past students who have successfully completed the entire tutor (*exemplars*). A score value above or below the threshold determines the user's relative *proficiency* in the subject matter, and the difficulty of the problem set given in the next level. This process continues through the end of the tutor. **This is a new, novel approach to achieve mastery learning.**

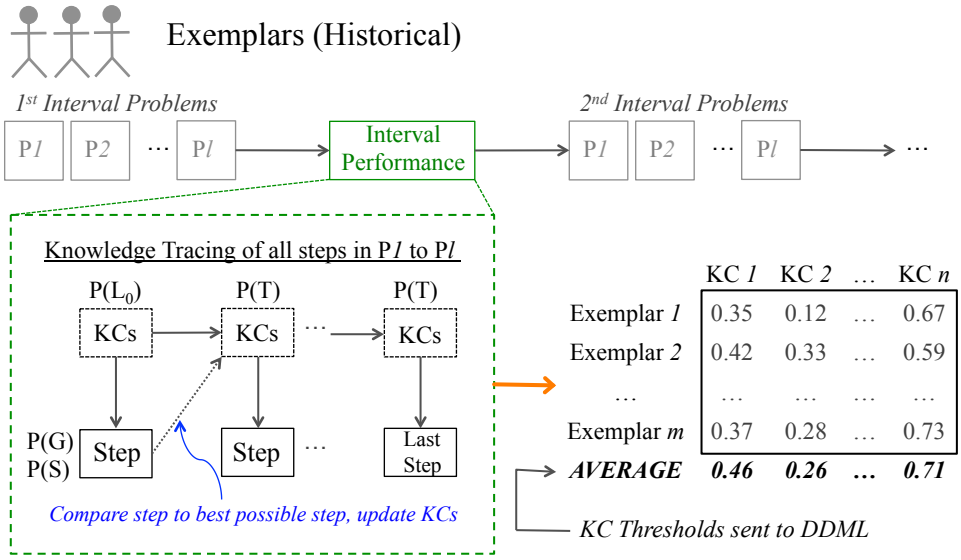

Figure 4.1 The data-driven threshold builder. Knowledge components (KCs) for each exemplar are updated using action steps from an interval set of tutor problems. The KC score averages at each interval are used as thresholds in the DDML system.

The DDML system is similar to the "level up" procedure by VanLehn [Van11b], which was designed to measure tutor learning gains without the need for pre- and post-testing. The DDML system is a mastery learning system with embedded assessment, split into levels of varying difficulty,

and requiring users to demonstrate competence in their current level before proceeding to the next. However, the DDML system is data-driven, using a comparison of current student performance to past exemplars for assessment, rather than using a static mastery test that must be passed. The DDML system is dependent only on the existence of prior tutor data. The knowledge-traced actions taken in tutor are dependent only on the domain itself. Since the DDML system considers all actions taken in tutor and weighs them by current level priority, it is ideal for open-ended problem solving domains, such as deductive logic.

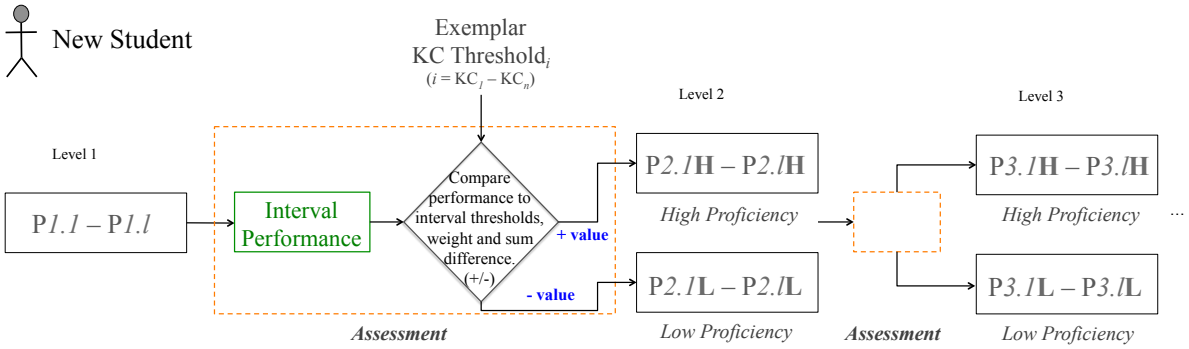

Figure 4.2 The Data-Driven Mastery Learning system. At each level interval, new student KC scores are calculated and then compared to exemplar thresholds for corresponding problem sets (see Figure 4.1). The +/− threshold sign difference is weighted by KC priority and summed (see Section 4.4.3). The sign of the result determines whether students are assigned problem sets of high or low proficiency in the next level.

4.3 Hypothesis

The data-driven mastery learning system in DT2-MP will result in higher tutor completion and lower tutor dropout than DT0 and DT1-NSH.

For this experiment, three versions of the Deep Thought logic tutor were compared: the original undirected proof tool with un-ordered problem sets (DT0); DT0 with integrated on-demand hints (DT1-NSH); and DT0 integrated with data-driven mastery learning (DT2-MP) (see Table 5.2).

Analysis of the results indicate that students using DT2-MP had significant improvement in full tutor completion by almost 3 times over students using DT0. Students using DT2-MP also showed improvement over students using DT1-NSH, although this was not statistically significant. The

Deep Thought Version	Features
DT0	Basic Deep Thought Logic Tutor
DT1-NSH	Deep Thought with on-demand step level hints
DT2-MP	Deep Thought with the DDML implemented for problem selection

Table 4.1 Overview of Deep Thought versions 0-2

results show that DT2-MP's data-driven mastery learning system is effective in improving tutor percent completion for most students over DT0 and DT1-NSH, confirming the hypothesis.

4.4 Approach

DT2-MP's data-driven mastery learning system consists of two major components: a mastery learning leveling component; and an assessment component. Their processes are described in this section.

4.4.1 Mastery Learning: DT2-MP Leveling Up

To provide sufficient student-performance benchmarks for regular assessment, DT2-MP breaks the problem sets of DT0/DT1-NSH into more levels.

The new DT2-MP problem sets maintain the same rule applications and the same difficulty level of problems in DT0/DT1-NSH. Each of the three sets of problems in DT0/DT1-NSH is split into two sets in DT2-MP to allow regular assessment of student rule application at the end of each level. DT2-MP Level 1 & 2 problems require basic logical implication rules, Level 3 & 4 problems expand the rule requirements to include more advanced logical implication rules, and Level 5 & 6 problems require logical equivalence rules in addition to implication rules. The new problem sets were made and tested by two domain experts to ensure consistency between DT0/DT1-NSH and DT2-MP problem rule requirements and problem difficulty.

The DT2-MP DDML system strictly orders levels and their problem sets to ensure consistent, directed practice using increasingly difficult logic rules. Problems sets are presented at two degrees of difficulty (higher and lower). As opposed to traditional mastery learning, where students are required to complete a number of training tasks until they pass a mastery test, the DDML system requires a minimum number of completed problems in a level. Their cumulative performance on target-rule actions at the end of a level determines whether they attempt the next level at higher or lower proficiency, as shown in Figure 4.2.

Table 4.2 An example of lower and higher proficiency set problems from DT2-MP requiring the same target-rules: Level 4 Problem 3 from the lower proficiency set (top); Level 4 Problem 2 from the higher proficiency set (bottom). The prioritized rules required for these problems are Conjunction and Constructive Dilemma.

#	***Premise***	***Derivation***
1	$(A \rightarrow B) \wedge (\neg D \rightarrow F)$	Given
2	$A \vee \neg D$	Given
3	$\neg A \rightarrow (D \vee G)$	Given
4	$\neg A$	Given
5	$B \vee F$	1,2/Constructive Dilemma
6	$\neg D$	2,4/Disjunctive Syllogism
7	$D \vee G$	3,4/*Modus Ponens*
8	G	6,7/Disjunctive Syllogism
9	$(B \vee F) \wedge G$	5,8/Conjunction

#	***Premise***	***Derivation***
1	$Z \rightarrow (\neg Y \rightarrow X)$	Given
2	$Z \wedge \neg W$	Given
3	$W \vee (T \rightarrow S)$	Given
4	$\neg Y \vee T$	Given
5	Z	2/Simplification
6	$\neg W$	2/Simplification
7	$\neg Y \rightarrow X$	1,5/*Modus Ponens*
8	$T \rightarrow S$	3,6/Disjunctive Syllogism
9	$(\neg Y \rightarrow X) \wedge (T \rightarrow S)$	7,8/Conjunction
10	$X \vee S$	4,9/Constructive Dilemma

In DT2-MP, evaluation of student performance is performed at the beginning of each level of problems. Level 1 of DT2-MP contains three problems common to all students who use the tutor, and provides initial performance assessment data to the DDML model.

Levels 2–6 of DT2-MP are each split into two distinct sets of problems, labeled higher and lower proficiency. The problems in the different proficiency sets prioritize the same rules judged by domain experts to be important for solving the problems in that level. However, the degree of problem solving difficulty between proficiency sets is different, with problems in the low proficiency set requiring fewer numbers of steps for completion, lower difficulty of logical expressions, and lower number of rule applications than problems in the high proficiency set. An example of a high proficiency set problem and a low proficiency set problem, and the steps to complete them, are

listed in table 4.2. These two problems both start with four premises and prioritize the correct usage of the same two rules – Conjunction and Constructive Dilemma – but there are a couple of differences. The lower proficiency problem (top) requires five steps to reach the conclusion, while the higher proficiency problem (bottom) requires six. Additionally, it is much more readily obvious where to apply one of the prioritized rules - Constructive Dilemma - in the low proficiency problem as opposed to the high proficiency problem. In the low proficiency problem the student is able to apply Constructive Dilemma directly from the premises, which would resemble the types of examples the student would likely have seen in a classroom setting. In the high proficiency problem on the other hand, the student must derive a few additional logic statements before they are able to use either of the prioritized rules. Figure 4.3 shows the possible path progressions of students using DT2-MP. Students cannot progress from one level to the next in DT2-MP without showing an expected level of proof solving mastery in the current level by completing the problems in their proficiency path.

Students on the higher proficiency path for a given level are required to solve two proof problems of higher difficulty to continue to the next level, while students on the lower proficiency path are required to solve three problems of lower individual difficulty with identical target-rules to continue to the next level. This difference in the number of problems is compensated for by the length of the problems. Harder problems are longer and easier problems are shorter, allowing for similar numbers of opportunities to apply each target-rule. An example of the difference between proficiency sets in a given level is shown in Figure 4.4, where students solve 3 lower proficiency problems in about 14 cumulative steps or 2 higher proficiency problems in about 15 cumulative steps. Individual problems in the lower proficiency set each require fewer rule applications than individual problems in the higher proficiency set; however, by giving an additional problem in the lower proficiency set, students receive more opportunity to practice in applying specific target-rules and problem solving skills.

In DT0 and DT1-NSH, students were allowed to solve problems in any order, giving flexibility in how students chose to approach the assignment, and preventing a student from becoming stuck on any individual proof. Because the problem sets in DT2-MP are completely ordered instead of random access, DT2-MP allows students to temporarily skip problems within a level, to allow that flexibility.

Depending on the state of progression in the current level, skipping a problem has several outcomes. Students who skip once in the higher proficiency set (with two problems) are given the next unsolved problem in the set. Skipping twice in the higher proficiency set will drop students to the lower proficiency problem set in the same level, regardless of how many problems they solved in the higher proficiency set (see Figure 4.3). Students in the lower proficiency set (with three

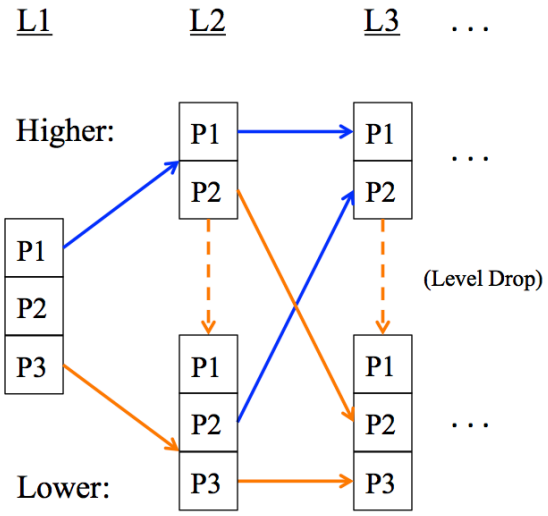

Figure 4.3 DT2-MP path progression. At each level, students are evaluated and provided either the higher or lower proficiency problem sets. Students can also be switched from the higher to lower proficiency set within a level.

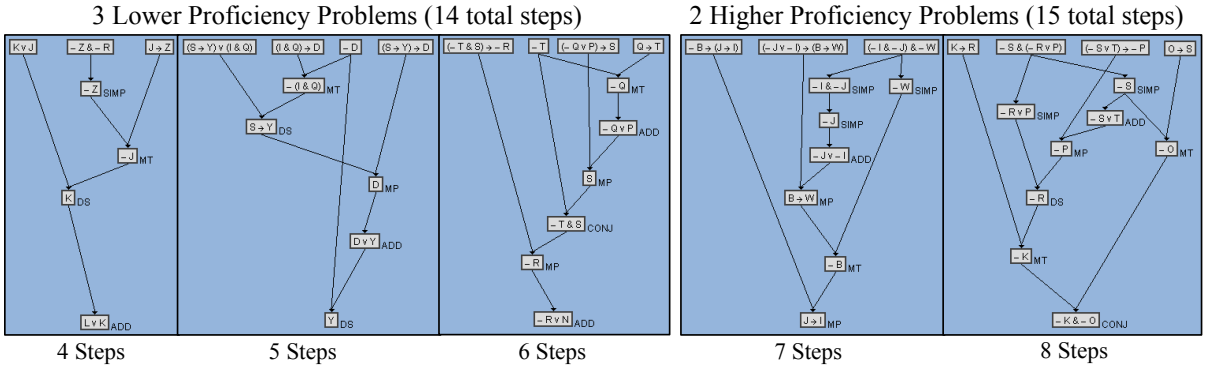

Figure 4.4 An example of lower and higher proficiency set problems from DT2-MP, Level 3. Lower proficiency proofs contain fewer rule application steps so students do one more problem to have more target-rule opportunities. The target-rules for this level are Modus Tollens, Addition, Conjunction (high priority) and Modus Ponens, Disjunctive Syllogism, Simplification (low priority).

problems) who skip a given problem are first offered an alternate version of the same problem with different ordering of required rule applications in the proof, before moving them to the next unsolved problem in the set (see Table 4.3). This gives students who get stuck on a problem the opportunity to approach the problem in a different manner. Students who repeatedly skip problems in the lower proficiency set will cycle through the unsolved problems in the set until three assigned problems have been solved, before moving on to the next level in the tutor.

Table 4.3 An example of regular and alternate problems from a lower proficiency set in DT2-MP: Level 2 Problem 2 (top); Level 2 Problem 2-Alt (bottom).

#	***Premise***	***Derivation***
1	$(\neg K \vee L) \rightarrow (M \wedge N)$	Given
2	$K \rightarrow O$	Given
3	$\neg O$	Given
4	$\neg K$	2,3/*Modus Tollens*
5	$\neg K \vee L$	4/Addition
6	$M \wedge N$	1,5/*Modus Ponens*
7	N	6/Simplification

#	***Premise***	***Derivation***
1	$(\neg O \vee L) \rightarrow (M \wedge \neg N)$	Given
2	$\neg O$	Given
3	$K \rightarrow N$	Given
4	$\neg O \vee L$	2/Addition
5	$M \wedge \neg N$	1,4/*Modus Ponens*
6	$\neg N$	5/Simplification
7	$\neg K$	3,6/*Modus Tollens*

Since there are more problems per level in the lower proficiency track than the higher proficiency track, we had to ensure that being dropped from the higher to lower path would not require the student to solve more problems than would be otherwise required. Therefore, the system maintains a maximum of three required solved problems to continue to the next level, taking into account any problems that may have been solved in the higher proficiency set, and ensures that all rules required for demonstration are presented to the student regardless of skipped status. The purpose of the skipped problem sub-system is to compensate for students who may have shown higher proficiency in a previous level, but have a harder time with the current set of problems. Students

given the higher proficiency set are expected to have satisfactory mastery needed for the next set of problems, without the need for alternate problems that reorder rule applications. If students have difficulties with the harder set, they are given the opportunity to work through a greater number of easier problems in order to practice the required rules before moving on to the next level.

At the beginning of the next level, accumulated data from the student's performance in the tutor is used with the data-driven knowledge tracer (DDKT) presented in the next section to determine the proficiency set given to the student. Between the two proficiency sets, and the alternate problems offered in the lower proficiency set, there are 43 problems in the DT2-MP problem set between Levels 1–6. Students are required to solve at least 13 problems for full tutor completion if always given the higher proficiency set, and are required to solve at most 18 problems if always given lower proficiency set.

DT2-MP saves student progress in the tutor, placing the student at the beginning of their most recent non-completed problem each time the student logs into the system, as opposed to DT0/DT1-NSH, where students select a problem to work on from a list. This prevents the ability to repeatedly complete the same problem, and ensures that the student stays on track to complete the tutor.

The mastery learning system in DT2-MP is designed to be generally applicable, and is not specific to propositional logic; proficiency path progression is independent of the type of problem presented in the tutor. The same process that was applied in the DDML system was envisioned to be applicable to any open-ended problem solving domain, and where quality of student solutions are difficult to evaluate. By limiting the number of problems students are required to solve and by not requiring specific mastery tests to progress to the next level, the DDML system ensures that students do not spend an unreasonable amount of time solving problems in the current level – allowing them to move forward in the tutor and be exposed to new concepts – yet still receive practice with problems that have difficulty relative to their performance. The difficulty of the problem sets that students are given is dependent on how their performance compares to past exemplars, which is described in the next section.

4.4.2 Assessment: DDKT Rule Score Updating

The instructor who developed Deep Thought chose proof problems that were deemed necessary for students to solve to demonstrate sufficient skill in propositional logic by the end of the course curriculum. It was hypothesized that students who have completed Deep Thought in the past have acquired a coherant skill set for proof problem solving and rule application. By splitting the DT0/DT1-NSH problem set into levels, DT2-MP allows regular evaluation of new students compared to the standard of what exemplars were able to accomplish at corresponding points in the tutor.

DT2-MP uses data-driven knowledge tracing (DDKT) of past and current students to evaluate student performance, and specifically the way students apply logic rules. The knowledge tracing calculations require the following parameters to be defined; the learning value $p(L_0)$, which is the probability that a rule was learned prior to its application; the acquisition value $p(T)$, the probability that the rule will have been learned after the student has been given the opportunity to apply it; the guess value $p(G)$, the probability that the student will guess the correct application of the rule before it has been learned; and slip value $p(S)$, the probability that a student will make a mistake in applying the rule if it has been learned. For each student, a DDKT score $ruleScore_i$ for each logical rule i in the tutor is created when a student first logs into DT2-MP, and these scores are maintained and updated at each rule application made by the student. Table 4.4 shows the list of rules required for completion of the tutor.

Table 4.4 List of rules required for completion of DT2-MP.

Rule	
MP	*Modus Ponens*
DS	Disjunctive Syllogism
SIMP	Simplification
MT	*Modus Tollens*
ADD	Addition
CONJ	Conjunction
HS	Hypothetical Syllogism
CD	Constructive Dilemma
DN	Double Negation
DEM	DeMorgan's
IMPL	Implication
TRANS	Transportation (Contrapositive)
EQUIV	Equivalence

The $ruleScore_i$ for a given rule i is initialized with a learning value $p(L_0) = 0.01$, acquisition value $p(T) = 0.01$, guess value $p(G) = 0.3$, and slip value $p(S) = 0.1$. After each observed attempted application of rule i, $ruleScore_i$ is updated using Bayesian knowledge tracing equations for inference and prediction of individual skill knowledge [EB12], shown below. Both of the equations are the sums of two probabilities; the posterior probability that rule i was already learned based on correct or incorrect application of the rule, and the probability that the rule will be learned if it had not already been learned. Equation 4.1 is used for rule i when that rule is correctly applied in

the current state, reflecting the probability of the student knowing and correctly applying that rule. Equation 4.2 is used for rule i when that rule is not correctly applied in the current state. Applying a rule correctly in the context of the current problem shows evidence that the student recognizes the applicability of that rule, while an incorrect rule application implies lack of recognition of a better option.

$$p(L_n) = \frac{p(L_{n-1})(1-p(S))}{p(L_{n-1})(1-p(S)) + (1-p(L_{n-1}))p(G)} + (p(1-L_{n-1}))p(T) \tag{4.1}$$

$$p(L_n) = \frac{p(L_{n-1})p(S)}{p(L_{n-1})p(S) + (1-p(L_{n-1}))(1-p(G))} + (1-p(L_{n-1}))p(T) \tag{4.2}$$

In the next section, it is explained how $ruleScore_i$ is used to calculate student proficiency.

4.4.3 Assessment: Proficiency Determination

By the time the student has completed a level of the tutor, that student will have accumulated a set of rule scores for each rule i calculated based on their performace. These scores can then be used to determine the student's current level of mastery, as well as what set of problems to complete next. At the end of each level, the DDKT scores for each rule $ruleScore_i$ are compared to a threshold value for that same rule. This threshold value, $ruleThreshold_i$, was calculated using data-driven knowledge tracing of DT0 tutor logs from six 2009 Deductive Logic course sections via equations 4.1 and 4.2. Only students who completed the entire DT0 assignment ($n = 302$) were used for threshold calculation, since these exemplars demonstrated proficiency for proof problem solving using all required rules. DDKT student scores were computed for problems in DT0, and mapped to the corresponding levels in DT2-MP. The scores from DT0 for each rule score were averaged at each of these break points and set as $ruleThreshold_i$. Students using DT2-MP are therefore judged to have proficiency on a rule based on how their performance compares to average student usage from previous use of Deep Thought by past exemplars.

For $i = 1 : n$

$$scoreSign_i = \textbf{sign}(ruleScore_i - ruleThreshold_i) \tag{4.3}$$

For each student in DT2-MP, every $rule_i$ is assigned a positive $scoreSign_i$ value if the rule score is above $ruleThreshold_i$, the prior student performance for students who completed the whole tutor. Each $rule_i$ receives a negative $scoreSign_i$ value if the score is below the $ruleThreshold_i$, meaning that they have not yet shown the same level of proficiency in deciding which rules to use as past students who successfully completed all the problems in DT0. This positive or negative value is weighted based on the priority of rule i in the current level. Rule priority is determined by the set of

rules deemed necessary by domain experts to best solve the current proof problem.

Rules with high priority (rules required for completion of the proof problems in that level) are weighted by 1. Rules with low priority (rules not required for completion of the proof problems in that level) are weighted by 0.5. Rules with 0 priority (rules that are not required for completion of the proof problems in that level) are weighted by 0. Low priority rule scores are included in calculation of student proficiency because it is assumed that every legal or illegal rule application, regardless of level rule priority, is an indication of a student's knowledge (or lack thereof), and should be considered when determining a student's overall proof-solving proficiency at any given point.

Proficiency =

$$\textbf{sign}\left[\sum_{i=rule_0}^{rule_n} scoreSign_i \times \begin{cases} 1 & \text{if priority}_i = \text{high} \\ 0.5 & \text{if priority}_i = \text{low} \\ 0 & \text{if priority}_i = 0 \end{cases}\right] \tag{4.4}$$

The weighted $scoreSign_i$ values are summed, and the sign of the sum determines whether a student is sent to the higher proficiency path ($Proficiency = +$ value) or the lower proficiency path ($Proficiency = -$ value) in the next level, as shown above.

The DDML system measures proof-solving proficiency by aggregating the scores of individual tutor actions. This focus on individual student actions rather than domain-based knowledge concepts is what allows the DDML system to be domain independent. The DDKT method used in the DDML system can be applied to any subject matter where actions taken by students can be prioritized in the context of the current problem set. The list of rules or operations required for the tutor are dependent by the domain; all other operations are only dependent on prior student data.

4.5 Method

The DDML system was implemented into Deep Thought (DT2-MP), and tested in Spring 2013 to determine its effect on tutor completion, student dropout, and overall learning effect.

4.5.1 Dataset & Sampling Method

DT2-MP was used as a mandatory homework assignment by students in a philosophy deductive logic course in the Spring 2013 semester (DT2-DDML group, $n = 47$). Students were allowed to work through the problem sets at their own pace for the entire 15-week semester. Problem Levels 1–6 were assigned for full completion of the tutor, totalling 13–18 problems depending on proficiency path progression. Class credit was given at completion of each level.

Data from the DT2-MP was compared to data collected in two prior semesters (Spring and Fall 2009) of the same philosophy course using DT1-NSH and DT0, respectively. Both courses were taught by the same instructor as the DT2-DDML group, and were assigned the same 13 problems in DT0 and DT1-NSH for full completion of the tutor. The problems in DT0 and DT1-NSH cover the three logical rule sets described in Section 4.4.1 (basic implication rules; advanced implication rules; implication and equivalence rules), and correspond to problems in DT2-MP.

The Spring 2009 students used the version of Deep Thought (DT1-NSH) with a hint-generation system developed by Barnes & Stamper to aid students in solving proof problems (DT1-Hint group, $n = 48$). Students using DT1-NSH show a significant increase in tutor completion over the DT0 system without hints [BS08]. The Fall 2009 students used the original unaltered DT0 (DT0-Control group, $n = 43$) with no hints. Class credit was given at the completion of each assigned problem.

The DT2-DDML group were only provided hints in problems that were identical to hint-problems from DT1-NSH. In total there were 8 problems with available hints out of the 43 problems in the DT2-MP problem set, spread between proficiency paths and alternate problems. The odds of a student enountering more than three of the hint-problems in the tutor was less than 1 in 32. Due to the very low likelihood that any individual student would encounter a high percentage of these hint problems during the course of the tutor, the DT2-DDML group is considered a non-hint group.

4.5.2 Statistics & Metrics

4.5.2.1 Tutor Completion

Percentage completion of the tutor is used as a measure of overall success for each group of students; it is a measure of *how far* students progress through the tutor. However, because of differences in assignment structure, calculation of percent completion for DT0/DT1-NSH is not the same as DT2-MP. For the DT1-Hint and DT0-Control group, class credit was given per-completed problem. Because problem selection was un-ordered, and the number of problems assigned per problem set was not uniform, the percentage completion of the total tutor is calculated as:

$$pctComplete = \frac{\text{assigned problems solved}}{\text{total assigned problems}}. \tag{4.5}$$

For the DT2-DDML group, class credit was given per-level completed. In DT2-MP, the number of problems necessary for problem completion is dependent on individual student path progression. Students who remain strictly in the high proficiency path work a total of 13 problems, and students who remain strictly in the low proficiency path work a total of 18 problems; students who move between proficiency paths can work any number of problems between the two limits. Therefore,

since path-progression for an individual student is not known unless the entire assignment is completed, percentage completion is calculated as:

$$pctComplete = \frac{\text{assigned levels completed}}{\text{total assigned levels}}. \tag{4.6}$$

4.5.2.2 Student Dropout

Student dropout is defined as the termination of tutor activity prior to completion of assigned problems (DT0/DT1-NSH) or levels (DT2-MP). It is a measure of *how many* students are using the system in each level.

4.5.2.3 Rule-Application Errors

Rule-application errors are committed when students incorrectly apply a rule, either because of invalid logical operation, or incorrect number of selected premises. These errors are more indicative of students dealing with conceptual challenges, and committing conceptual errors by the invalid use of rules applied to justified premises, rather than operational errors that indicate the student is struggling with the tutoring system itself. Our hypothesis is that since the system in DT2-MP shows improved proficiency of target-rules in tutor, the rate of rule-application error should be lower than in the DT0-Control and DT1-Hint groups.

4.5.3 Rule Score Threshold Comparison

The DDML system was designed to use historical exemplar data as a benchmark for new students, rather than an expert model. This assumes that student skill development changes over the course of the tutor, and does not presume that even prioritized actions are optimal for success. Therefore, evaluation of the overall learning effect of the DDML system model is determined by comparing final $ruleScore_i$ scores for all i rules from the DT2-DDML group to end-of-tutor $ruleThreshold_i$ scores from the DT0 logs. The $ruleThreshold_i$ scores are calculated using DDKT Equations 4.1 & 4.2. End-of-tutor $ruleThreshold_i$ scores and the DT2-DDML group scores were taken only from students who completed the entire tutor, to measure successful student rule-application performance.

4.6 Results

4.6.1 Tutor Completion

Table 5.4(a) shows the number of total assigned problems solved in tutor for the three groups. Students in the DT2-DDML group solved 13 problems on average – the minimum required for completion of the tutor – while students in the DT1-Hint group completed 8 assigned problems out of 13, and students in the DT0-Control group completed 6 out of 13.

Table 4.5 (a) Number of total assigned problems solved in tutor. (b) Percentage of tutor completion. * indicates significance over the control.

(a)	DT2-DDML	DT1-Hint	DT0-Control
Mean	13.09 *of 13–18*	7.98 *of 13*	6.07 *of 13*
StDev	4.94	4.78	5.20

(b)	DT2-DDML	DT1-Hint	DT0-Control
Mean	79.79*	61.38	46.69
Median	100.0	61.54	46.15
StDev	29.88	36.78	40.02

Table 5.4(b) shows that students from the DT2-DDML group completed 33% more of the tutor on average than the DT0-Control group, and 18% more of the tutor than the DT1-Hint group, with lower variance in the final scores. The median scores show that over half (55%) of the DT2-DDML group completed the entire tutor (see Table 4.6 for tutor completion by group).

A one-way ANOVA test on percent completion difference was performed on tutorial completion across all three test groups, showing significance ($F(2,120) = 7.559$, $p = 0.001$). A Tukey post-hoc comparison shows a significant improvement of the DT2-DDML group ($M = 0.80$, 95% $CI[0.70, 0.90]$) over the DT0-Control group ($M = 0.51$, 95% $CI[0.40, 0.62]$, $p < 0.001$, *Cohen's d* $= 0.84$). Although the DT2-DDML group had a higher mean percentage completion than the DT1-Hint group, the results were not significant ($p = 0.138$, *Cohen's d* $= 0.35$). However, this still indicates that DT2-MP's data-driven knowledge tracing combined with individualized problem set selection can improve student percent completion as much as on-demand hints (DT1-NSH).

4.6.2 Student Dropout

Figure 5.2 shows the dropout trend of the three data groups over the course of the tutor. The markers on the DT2-DDML line indicate the end of each level in DT2-MP, and the markers on the DT1-Hint and DT0-Control lines indicate the end of each problem in DT1-NSH and DT0. The horizontal axis represents the percentage completion of the tutor, calculated as in Equations 4.5 & 5.1. The vertical axis represents the percentage of all students in each group active in the tutor at the completion of each level (DT2-MP) or problem (DT1-NSH, DT0).

Both the DT1-Hint and DT0-Control groups have higher dropout than the DT2-DDML group across all problems. The DT0-Control group also has greater dropout than the DT1-Hint group, with a noted difference in higher level problems (corresponding to Levels 4–6 in DT2-MP). This is consistent with the results found by Stamper et al [Sta11], that on-demand hints help student retention in-tutor.

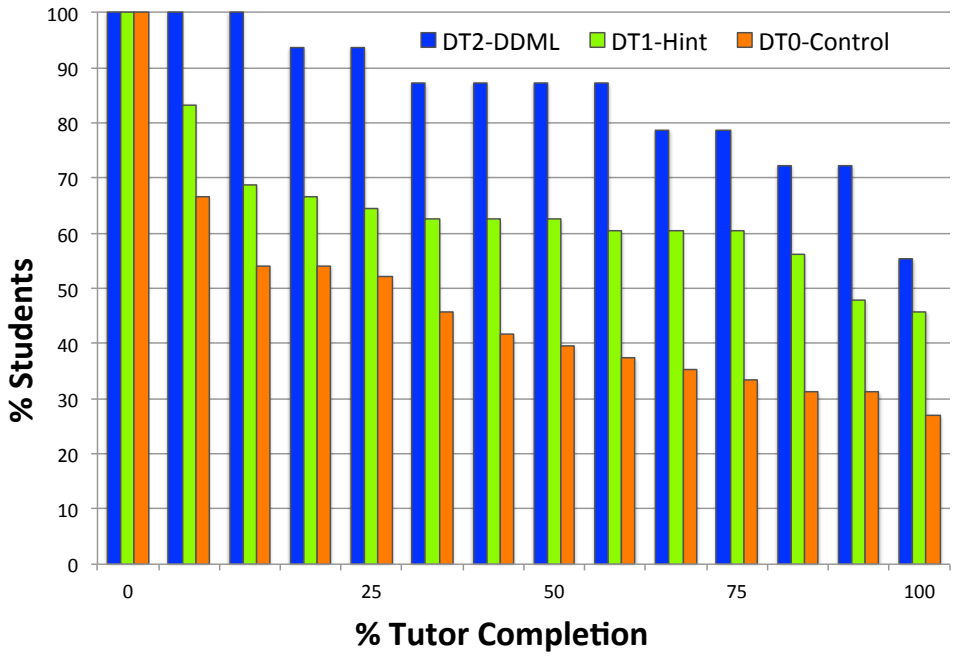

Figure 4.5 Rate of student dropout for the DT2-DDML, DT1-Hint, and DT0-Control groups over the course of the tutor.

Table 4.6 summarizes the number of students who completed and dropped out of the tutor across all three groups. Students in the DT2-DDML group were 2.97 times as likely to complete

the tutor than students in the DT0-Control group, and 1.77 times as likely as students in the DT1-Hint group. An overall chi-square test was performed to examine the relationship between student dropout and group, showing significance ($\chi^2(2, 123) = 11.75$, $p = 0.003$). The DT2-DDML group was significantly less likely (by 82%) to drop out of tutor when compared to the DT0-Control group ($\chi^2(1, 84) = 11.50$, $p = 0.001$, $OR = 5.31$). Although the DT2-DDML group had lower student dropout than the DT1-Hint group [put dropout rate, reference Table 7], the difference was not significant ($\chi^2(1, 86) = 3.23$, $p = 0.072$, $OR = 2.21$). The DT1-Hint group had a lower dropout rate than the DT0-Control group, but not significantly ($\chi^2(1, 76) = 2.74$, $p = 0.098$, $OR = 2.40$).

Table 4.6 Student completion of the tutor by group. Dropped indicates that the student did not complete Deep Thought.

Group	Completed	Dropped	Total
DT0-Control	8 *(18.6%)*	35 *(81.4%)*	43
DT1-Hint	15 *(31.3%)*	33 *(68.7%)*	48
DT2-DDML	26 *(55.3%)*	21 *(44.7%)*	47
Total	49 *(35.5%)*	89 *(64.5%)*	138

The last result is different than the result from Stamper et al [Sta11], where the addition of on-demand hints significantly reduced dropout rate; however, the data set in the experiment run by Stamper et al had a higher sample size ($n = 105$ from the Hint group, $n = 98$ from the Control group), taken from six classes taught by three different instructors. Our experiment used a sub-set of the same data, only using the two classes taught by the same instructor as DT2-MP for consistency in instructional methods. Therefore, because of the small sample size, and with instructor effects removed, the same significant result for dropout rate as Stamper et al was not achievable.

As a whole, the DDML system in DT2-MP was as effective in improving student in-tutor performance as much as on-demand hints in DT1-NSH, and significantly better than DT0. The purpose of on-demand hints is to aid and improve student learning, and tutor performance is a reflection of a student's knowledge and abilities in the subject matter. This result is important for intelligent tutor design, indicating that the DDML system presented here can achieve or outperform the learning gain from on-demand hints, reducing tutor development time. The next two sections explore performance in the application of rules in DT2-MP.

4.6.3 Rule-Application Errors

Rule-application error rates for the three groups are shown in Table 4.7. The average rule-application error rate for the DT2-DDML group is much lower than the DT1-Hint group. However, the rate of error for the DT0-Control group is lower than either the DT2-DDML group or the DT1-Hint group, counter to expectations. The low rule-application error rate of the DT0-Control group is explained by the high dropout rate early in the first problem set (over 50%). Students who dropped out committed almost 40% of the rule application errors, but made up less than 25% of total tutor interactions by the DT0-Control group.

Table 4.7 Percentage of rule-application errors to total interactions in tutor. * indicates significance over the other groups.

	DT2-DDML	DT1-Hint	DT0-Control
Mean	4.61	6.97*	2.92
Median	3.52	4.01	1.90
StDev	4.90	5.92	3.90

A one-way ANOVA test on error difference was performed across the three test groups, showing significance ($F(2,120) = 6.586$, $p = 0.002$). The DT1-Hint group had a significantly higher error rate than the DT2-DDML group ($M = 0.046$, 95% $CI[0.032, 0.06]$, $p = 0.034$) and the DT0-Control Group ($M = 0.028$, 95% $CI[0.012, 0.044]$, $p < 0.001$, $d = 0.83$).

Since the DDML system was built using level thresholds by DT0 exemplars with no in-tutor direction, the next section explores how those threshold scores compare to student scores from DT2-MP, with the expectation that rule scores from DT2-MP should match or exceed threshold values.

4.6.4 Rule Score Threshold Comparison

A comparison for rule score thresholds are shown in Table 4.8. The rules listed in Table 4.8 are rules that were required for completion of the tutor. The first eight rules listed are logical implication rules, prioritized in Levels 1–4 in DT2-MP. The last five rules listed are logical equivalence rules not required in Levels 1–4 but prioritized in Levels 5–6 in DT2-MP. The Difference column shows whether the DT2-DDML group rule scores are higher (+) or lower (–) than the $ruleThreshold$ scores. With the exception of *modus tollens* (MT) and constructive dilemma (CD), average rule scores for the DT2-DDML group are higher than the respective $ruleThreshold_i$ scores. This

indicates that students using DT2-MP show a higher overall awareness of proper individual rule usage by end of tutor over students using DT0.

Table 4.8 Average rule DDKT scores for DT2-DDML group students who completed the entire tutor ($n = 26$), compared to end-of-tutor $ruleThreshold_i$ averages from DT0 ($n = 302$).

Rule	$ruleThreshold_i$	**DT2-DDML**	***Difference***
MP	0.649	0.743	+
DS	0.491	0.647	+
SIMP	0.734	0.947	+
MT	0.352	0.238	−
ADD	0.426	0.747	+
CONJ	0.348	0.578	+
HS	0.455	0.669	+
CD	0.163	0.120	−
DN	0.437	0.697	+
DEM	0.182	0.423	+
IMPL	0.463	0.703	+
TRANS	0.298	0.433	+
EQUIV	0.106	0.151	+

The low scores for both MT and CD in both $ruleThreshold_i$ and the DDML group are acceptable in context of the material presented to students. Both rules had low representation in the tutor, and in the case of CD, only a single occurrance in each proficiency set. CD is also the most difficult rule presented in Deep Thought, and not emphasized in course material.

4.7 Discussion

The results from this study show that the DDML system used in DT2-MP combining DDKT and individualized problem selection is effective in improving overall student proof-solving proficiency, for the majority of students using DT2-MP.

The DT2-DDML group was almost 3 times as likely as the DT0-Control group to complete the tutor, and over 1.7 times as likely as the DT1-Hint group (Table 4.6). Students from the DT2-DDML group have higher completion of the tutor than either DT1-Hint and DT0-Control groups, with the DT1-Hint group performing better than the DT0-Control group (Table 5.4) This indicates that DT2-MP improves student-tutor completion even further than DT0 or DT1-NSH with hints. The

high median percentage completion especially indicates this improvement, as it shows that over half (55%) of all students in the DT2-DDML group completed the entire tutor.

Average DT2-DDML tutor completion (79.8%) was significantly greater than the DT0-Control group (46.7%), improving tutor completion by 33%. DT2-DDML group completion was also higher than the DT1-Hint group (61.4%), improving tutor completion by 18%. Students using DT2-MP were 82% less likely to drop out of the tutor than students using DT0, and 53% less likely than students using DT1-NSH. This indicates that problems were selected such that students were more likely to continue through the tutor, rather than giving up in frustration. These students experience more higher-level rules and have more practice solving problems than students in the DT1-Hint and DT0-Control group, increasing the likelihood of learning required rules and exceeding the threshold requirements of DT0 exemplars.

As illustrated in Section 4.6.3, rule-application error gives an indication of the overall conceptual challenge that students have in applying rules. The DT2-DDML and DT0-Control groups had significantly lower rule-application error rates (4.61% and 2.92% respectively) than the DT1-Hint group (6.97%). The results indicate that the DT2-DDML and DT0-Control students had a better mastery of the rules and their proper usage. It was hypothesized that without the directed feedback presented to students in the DT1-Hint group, and without the explicit requirement to complete all problems in a given level in the DT2-DDML group, students in the DT0-Control group were prone to frustration, and were more likely to drop out early in the tutor, explaining the low error rate. It was also hypothesized from these results that the addition of on-demand hints in DT2-MP would increase the rule-application error rate for students, but still be less than the error rate for students in the DT1-Hint group.

Students using DT2-MP also showed overall proficiency in rule-application, as shown in Table 4.8. For 11 of the 13 required rules, completion of the entire Deep Thought tutor by students using DT2-MP resulted in higher DDKT rule scores than students who had used DT0. This indicates more consistent use of rules in the tutor when they are most needed. The two rules that showed lower proficiency had less representation in the tutor, and were therefore weighted less when determining proficiency level.

4.8 Conclusions

This chapter presented a holistic data-driven mastery learning system that uses knowledge tracing of domain concepts in existing student-tutor performance data to regularly evaluate current student proficiency and select problem sets. Analysis of the results show that this system can improve tutor completion and proficiency of applied target-rules compared to past exemplars.

Students using DT2-MP complete 33% more of the tutor on average than students using DT0, and 18% more of the tutor than students using DT1-NSH with hints. Students using DT2-MP have a lower rate of tutor dropout than students using DT0 and DT1-NSH, exposing more students to important domain concepts. DT2-DDML students are also significantly more likely (by almost 3 times) to complete their entire assignment when compared to the DT0-Control group, and almost 1.7 times as likely as the DT1-Hint group. This is particularly important, since success in Deep Thought is correlated with course completion for introductory deductive logic [Sta11]. By solving more problems, DDML students have more practice solving logic problems than students using DT0 or DT1-NSH. These students also demonstrate a low rate of error in rule application, and show higher proficiency of tutor concepts. The conclusion is that the data-driven mastery learning system presented in this paper significantly aids mastery of core concepts, improves tutor scores, and lowers rate of tutor dropout for a majority of students over an undirected tutor, and at least as well as the same tutor with on-demand hints. It is expected that the integration of hints with the DDML system will provide an even greater effect than what was seen in this study, as detailed in the next chapter. In conclusion, the DDML is a successful strategy for selecting problems for students, and has therefore been incorporated into all subsequent versions of Deep Thought.

CHAPTER

5

STUDY TWO - DATA-DRIVEN MASTERY LEARNING WITH HINTS AND WORKED EXAMPLES

Data-driven Mastery Learning with Hints and Worked Examples

5.1 Research Question

Will the addition of problem-solving feedback, specifically next-step hints and worked examples, to the data-driven mastery learning system in DT2-MP further improve student performance?

5.2 Objective

To test if the addition of on-demand hints and worked examples to problems in DT2-MP increases tutor completion rate and reduces dropout, as hypothesized at the end of the Study One.

This chapter describes a study on the effect of augmenting the data-driven mastery learning system

from Chapter 4 with hints and worked examples. Stamper & Barnes [Sta11] demonstrated that the addition of on-demand hints increased tutor completion and reduced student dropout in DT1-NSH. The DDML system in DT2-MP further improved on these metrics without hints. In this chapter we evaluate DT3-Random (DT2-MP with hints and worked examples).

5.3 Hypothesis

- The addition of on-demand hints and data-driven worked examples to the data-driven mastery learning system will significantly increase tutor completion rate and reduce student dropout in Deep Thought.
- The ordering of worked examples and problem solving will affect student performance by reducing the amount of time spent on each step of a problem.

For this experiment, the original Deep Thought design and software base (used in DT0, DT1-NSH, and DT2-MP) was discarded and re-built using modern development tools as Deep Thought 3 (DT3) (see Table 5.1). The DDML system from DT2-MP was re-implemented and expanded to incorporate a pedagogical policy that chooses when to give a student either a problem to solve with on-demand next step hints, or a worked example of that same problem. In this chapter we do not evaluate the pedagogical policy, rather we investigate the impact of hints and worked examples.

5.4 Approach

5.4.1 Tutor Redesign

DT2-MP was redesigned to streamline the interface and improve usability to create DT3 (see Figure 3.1). In addition, DT3 was designed as a modular development and instructional platform. New features can be added and removed at the behest of the course instructor, or for research purposes. The DT3 design also provides comprehensive, detailed step-level logs, making analysis of any student behavior in the system possible.

DT3 was specifically designed to add on-demand hints and worked examples to the DDML problem set. As with DT2-MP, the DT3 problem set consists of 6 strictly ordered levels, each split into a higher proficiency track with a lower number of more difficult problems, and a lower proficiency track with a greater number of less difficult problems. Evaluation of student performance occurs at the beginning of each level of problems, with the first level consisting of common problems to collect initial data for assessment.

Deep Thought Version	Features
DT0	Basic Deep Thought Logic Tutor
DT1-NSH	Deep Thought with on-demand step level hints
DT2-MP	Deep Thought with the DDML implemented for problem selection
DT3-Random	Deep Thought with the DDML and on-demand step-level hints and/or worked examples (depending on features enabled by the instructor)

Table 5.1 Overview of Deep Thought versions 0-3

DT3 has the option of selecting any combination of no-hint, on-demand next-step hint, and worked example problems as determined by the course instructor. For the experiment in this study, DT3 adds on-demand hints to the DT2-MP problem set, as well as worked examples of the same problems, conditionally chosen as described below (DT3-Random). To ensure the worked examples did not greatly reduce the amount of practice students received solving problems, this version of DT3-Random adds an additional problem in each level (one in each proficiency track), required to solve, and with no on-demand hints. This also provides a built-in post-test on each level of the tutor.

5.4.2 Next-Step Hints

The on-demand next-step hints added to DT3-Random are created using Hint Factory. Hint Factory has has been shown to increase student learning and reduce student dropout [Sta11], and its inclusion in DT3-Random is expected to improve on the results from DT2-MP, which itself outperformed an undirected tutor with hints (DT1-NSH).

As with DT1-NSH, if hints are available for a given proof problem, clicking the "Get Hint" button with the mouse pointer will result in one of four possible hints for any individual proof-state, displayed on successive "Get Hint" button presses for that same step. The hints for the proof-state in Figure 3.2 are shown in Table 5.2.

The hints for DT3-Random were created using student log data from DT2-MP, and are available for every problem in the DT3-Random data set, with the exception of the last problem in each level, which did not allow hints. Instructors may select which problems will have available hints for students; this is helpful for ensuring that we collect student problem solving without hints for every problem.

The addition of next-step hints had previously been shown to increase the completion and dropout rate of students using Deep Thought between DT0 and DT1-NSH. It was hypothesized at the end of Chapter 4 that adding next-step hints to DT2-MP would show an additional improvement.

Table 5.2 A generated on-demand hint sequence for the proof-state in Figure 3.2

Hint #	Hint Text	Hint Type
1	Try to derive B working forward.	Indicate goal expression
2	Highlight B & C to derive it.	Indicate premises to select
3	Click on the rule simplification (SIMP).	Indicate the rule to use
4	Highlight B & C and click on Simplification to derive B	Bottom-out hint

5.4.3 Worked Examples

Worked examples incorporated into intelligent tutoring systems have the potential to increase learning gains and learning efficiency [HR09; McL08]. Figure 5.1 shows a worked example in DT3-Random. Students are given an entire solution to a proof problem step-by-step, with each step annotated with the next action. Students can move backward and forward between steps at leisure. Once a student attempts to continue past the final example step, the next problem in the level is given. Each problem in the DT3-Random data set has a corresponding worked example, which was created from past student proof solutions. For each problem, the solution chosen for the worked example was the one that had the lowest number of steps, and included all the primary expected rules for the given problem. In the event that a past student solution did not meet these criteria, an expert solution was used for the worked example. This was only needed for problems where there was little student data, which were only the alternate problems. The annotations for the worked examples were procedurally generated from the worked example steps.

5.4.4 Pedagogical Policy

To gather information of how student behavior in-tutor is affected by worked examples, the hint-based version of DT3-Random randomly assigns the next problem selected by the DDML as either a problem with on-demand hints to solve, or as a worked example of that same problem. To ensure students receive enough practice solving problems, and are assigned enough worked examples to collect sufficient data, a pedagogical policy was created to control the balance of instruction through example and the opportunity for proof-solving practice.

An additional problem was added to each level in the problem set (one in each proficiency track). These problems were designed to mirror the rules and problem solving strategies as the other problems in the respective levels and proficiency tracks. Unlike the other problems in the DT3-Random data set, the additional problems are presented without hints, and with no corresponding worked example. The purpose of these problems is three-fold. The first purpose is to ensure that

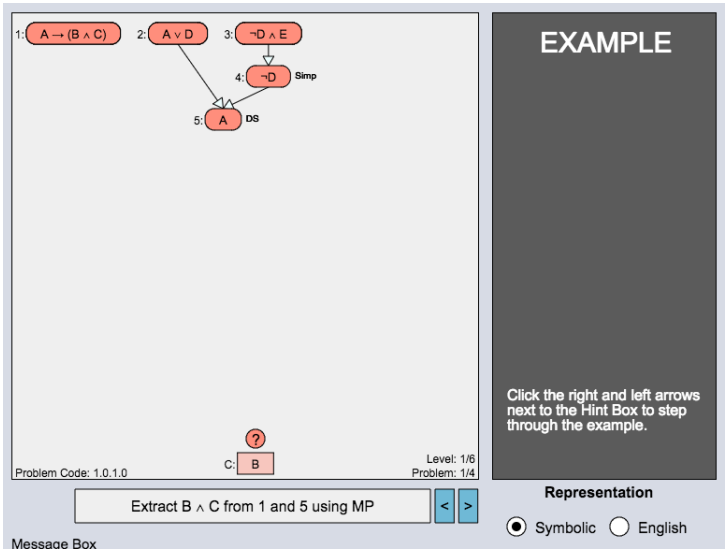

Figure 5.1 A worked example from DT3-Random, for the same problem in Figure 3.2. A problem step-state is displayed on the screen, with an annotation in the box below describing the next step. Students can move between worked example steps using the arrow buttons next to the annotation.

students receive sufficient practice solving proof problems. With some of the problems presented as worked examples, students will solve fewer problems over the course of the tutor than students using DT2-MP. The addition of a non-example problem adds an additional problem to solve.

The second purpose of the additional problem at the end of each level is to act as a demonstration of proof solving ability. Worked examples have shown to be effective in improving learning [SC85]. However, Deep Thought requires demonstration of proof-solving ability to assess learning between levels. As bottom-out hints are themselves a type of worked example [Shi08], which DT3-Random allows with next-step hints, the inclusion of a required no-hint proof problem forces students to demonstrate what they have learned before allowing them to continue forward.

Third, for the purpose of understanding the difference in learning gains between problem solving and worked examples for a later study, a comparison of performance of the same problem set with and without worked examples is needed. Within a level, the number of worked examples is randomly decided. By having a problem required for all students, the effect of worked examples can be determined by studying how the number of worked examples presented in a level affect student performance metrics in the additional "post-test" problem (such as time taken, rules applied, and any errors made).

For the hint-based version of DT3-Random, there are a total of three problems in the high proficiency track in each level, and four problems in the first level and low proficiency track in later

levels. The pedagogical policy ensures that for each level, students will receive at least one, and no more than two, worked examples and problems to solve in the high proficiency track, and at least two, and no more than three, worked examples and problems to solve in the common and low proficiency track.

Because of the reduction in the number of problems required to complete each level (1-2 for the high proficiency track and 2-3 for the low proficiency track), the rule threshold scores used for assessment by the DDML system will not be effective for students who solve fewer problems than in DT2-MP. Since there is less opportunity to demonstrate rule application, individual rule scores for a student solving fewer problems will be lower than the threshold for that level, even with ideal proof solving strategy. To counter-act this effect, the rule scores for each student are updated when students receive a worked example with the rules and number of rule application in the worked example, as if the student had solved the problem themselves. [**Shi08**] found that students who go through a worked example learn as much as if they had worked through the problem themselves. This is expected to put most students on the high proficiency path in each new level, assuming their cumulative performance up to that point is satisfactory. However, as with DT2-MP, students have the option of skipping problems in the high proficiency track if they are having difficulty, resulting in a drop to the low proficiency path in the same level. Also as with DT2-MP, the number of problems and worked examples required for a single level are maintained, regardless of a level drop.

5.5 Methods

5.5.1 Problem Set

DT3-Random with on-demand hints and data-driven worked examples was implemented and tested in Fall 2014 to determine its effect on tutor completion and student dropout. Problem Levels 1–6 were assigned for full completion of the tutor, resulting in students solving 6–18 problems and working through 7–10 worked examples depending on proficiency path progression and number of worked examples given. Class credit was given at completion of each level.

5.5.2 Dataset & Sampling Method

DT3-Random was used as a mandatory homework assignment by students in two computer science discrete mathematics courses taught by the same instructor in Fall 2014 (WE group, $n = 261$). Data from the DT3-Random was compared to data collected from DT2-MP for tutor completion and dropout comparison, since DT2-MP is the only previous version of Deep Thought that utilizes the data-driven mastery learning system.

DT2-MP was used as a mandatory homework assignment by students in a philosophy deductive logic course in Spring 2013 (NoWE group, $n = 47$). The DT2-MP problem set was identical to DT3-Random with the exception of the additional problems described in Section 5.4.4. Problem Levels 1–6 were assigned for full completion of the tutor, totalling 13–18 problems depending on proficiency path progression. Class credit was given at completion of each level.

DT3-Random also differs from DT2-MP in that it provides on-demand next step hints for each problem in the DT2-MP data set. However, since in the DT3-Random data set there are only 98 instances of students requesting hints out of ∼500,000 total actions, we consider the effect of hints on total student performance negligible, and these data points were removed for the worked example analysis.

For the results, in addition to a direct comparison of the DT2-MP and DT3-Random groups, a repeated random sampling of the DT3-Random group was compared to the DT2-MP group in order to account for the drastic difference in class sizes between DT2-MP and DT3-Random. A random sample of $n = 47$ students were chosen with replacement from the $n = 261$ students in the DT3-Random group, repeated 10,000 times to compare to the DT2-MP group. The average and median drop rate, mean completion rate, and p-values from one-way ANOVA tests are presented in the results.

5.5.3 Statistics & Metrics

We use percent completion as a measure of overall success for each group of students. Percent completion is a measure of *how far* students progress through the tutor, on average. Recall from Chapter 4 that the percentage of tutor completion for these groups is calculated as follows:

$$pctComplete = \frac{\text{assigned levels completed}}{\text{total assigned levels}}. \tag{5.1}$$

Student dropout is defined as the termination of tutor activity at any point in the tutor prior to completing all of the assigned problems. It is a measure of *how many* students are using the system in each level.

5.6 Results

5.6.1 Direct Group Comparison

Descriptive statistics for the entire WE group ($n = 261$) compared to the NoWE group ($n = 47$) is presented before comparing repeated random samples from the WE group to the NoWE group.

The first hypothesis is that data-driven worked examples would increase students' completion rate and reduce student dropout. In the analysis below I will refer to the data from DT3-Random as the WE group, and data from DT2-MP as the NoWE group. I present overall performance measures, completion, and dropout to answer our first hypothesis. Table 5.3 shows the number of problems solved, worked examples received, and total time spent in tutor for the WE and NoWE groups.

Table 5.3 The number of problems solved, number of worked examples received, and total tutor time for the WE and NoWE groups.

Group	**# Solved Problems**			**# Worked Examples**			**Total Tutor Time (mins)**		
	Mean	*Median*	*StDev*	*Mean*	*Median*	*StDev*	*Mean*	*Median*	*StDev*
WE	14.12	13	5.09	7.53	8	1.35	224.4	97.7	346.5
NoWE	13.08	13	4.94	–	–	–	307.2	244.4	263.4

Students in the WE group completed an average of 14.12 problems compared to an average of 13.08 problems in the NoWE group. A one-way ANOVA test was performed on the number of problems solved between the WE and NoWE groups with no significant difference found between them ($p = 0.327$). Students in the WE group spent an average of 224.4 minutes in the tutor, and students in the NoWE group spent an average of 307.2 minutes in the tutor, a difference of 27%. A one-way ANOVA test on time spent in tutor showed only marginal significance ($p = 0.063$).

5.6.1.1 Completion & Dropout

We use percent completion as a measure of overall success for each group of students. Table 5.4(a) shows the average percentage of tutor completion by group. Student dropout is defined as the termination of tutor activity at any point in the tutor prior to completing all of the assigned problems. Table 5.4(b) summarizes the number of students who completed and dropped out of the tutor across both groups.

On average, students in the WE group completed 94.02% of the entire tutor, compared to 79.79% of students in the NoWE group. A one-way ANOVA test shows a significant difference ($p = 0.014$). The average percentage of tutor completion was an improvement of 14.2%. Out of 261 students in the WE group, 236 student completed the entire tutor (90.4%) while 25 students dropped out of the tutor before completing it (9.6%), compared to the NoWE group where 26 students completed the entire tutor (55.3%) while 21 students dropped out of the tutor before completing it (44.7%). The difference in drop rate was very significant ($p < 0.001$).

Table 5.4 (a) Percentage of the tutor completed by group. (b) The number and percent of students who completed and dropped the tutor by group. Dropped indicates that the student did not complete Deep Thought. A * indicates significance.

(a)	Mean	StDev
WE	94.02*	21.07
NoWE	79.79	29.88

(b)	Completed	Dropped	Total
WE	236 *(90.4%)*	25* *(9.6%)*	261
NoWE	26 *(55.3%)*	21 *(44.7%)*	47
Total	262	46	308

To visualize the rate of dropout over the course of the tutor, Figure 5.2 shows the retention trends for the WE and NoWE groups at the end of each level. From this figure, it is evident that the worked examples are dramatically improving retention and tutor completion over the course of the tutor.

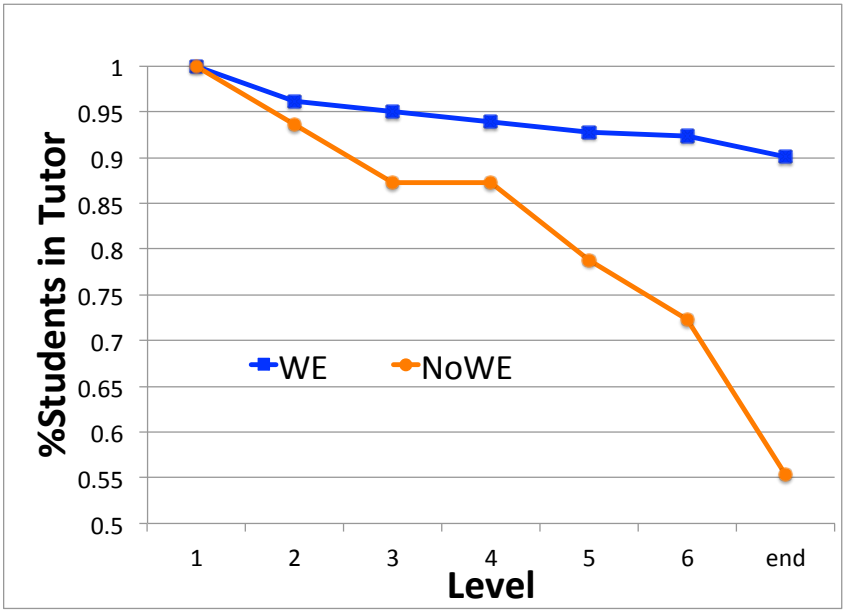

Figure 5.2 Rate of student dropout for the WE and NoWE groups over the assignment.

5.6.2 Sample Group Comparison

Because of the large difference in population between the NoWE and WE groups, a repeated random sampling of the WE group was made to compare to the NoWE group, in order to reduce the effect of different sample sizes. A random sample of $n = 47$ students were chosen from the $n = 261$ students

Table 5.5 Percent tutor completion, percent students dropped, and time in tutor for the NoWE group, as well as mean, median, and standard deviation values for the 10,000 random samples of size $n = 47$ from the WE group. The means, medians, and standard deviations for p-value results from the 10,000 individual ANOVA tests for each sample are also presented.

		WE					
		Sample Values			***ANOVA p-Values***		
	NoWE	*Mean*	*Median*	*StDev*	*Mean*	*Median*	*StDev*
%Completed	79.79	93.34	93.61	0.030	0.042	0.013	0.072
%Dropped	44.7	10.00	10.64	0.040	<0.001	<0.001	0.002
Total Tutor Time (mins)	307.2	224.5	221.7	45.57	0.272	0.172	0.273

in the WE group, 10,000 times with replacement. This matches the size of $n = 47$ students in the NoWE group. Completion, dropout, and time in tutor were calculated from each random sample, and a one-way ANOVA test was performed between the WE and NoWE groups in each sample to test for significance. These average of the overall performance measures are presented in Table 5.5.

On average, students in the WE group completed 93.34% of the entire tutor, compared to 79.79% of students in the NoWE group. This is consistent with the results from the direct comparison between the WE and NoWE groups. These results were significant for the vast majority of samples taken from the WE group (mean: $p = 0.042$, standard deviation = 0.072). Only 10.00% of students on average in the WE group dropped out of the tutor before full completion, compared to 44.7% of students in the NoWE group. This result is also consistent with the results from the direct comparison between the WE and NoWE groups. These results were extremely significant for all the samples taken from the WE group ($p < 0.001$). However, when compared to the time-in-tutor results from the direct comparison, on average there was no significant difference in the total time in tutor between the NoWE and WE groups.

The results show that students using Deep Thought with hints and worked examples (DT3-Random) completed more of the assigned problems and were less likely to drop out than students using the DDML alone (DT2-MP). Moreover the WE students who dropped out did so at a later point than the dropouts in the NoWE group, as shown in Figure 5.2. This indicates that the addition of worked examples increases the chances that students will be able to finish the tutor, and increases the percentage of problems they will complete.

5.6.3 Worked Example Ordering

Our second hypothesis is that the ordering of worked examples and problem solving would affect student performance. We used the additional end-of-level problems required in DT3-Random as

post-tests to study on the effect of the ordering of worked examples and problem solving practice in each level. We classified problem instances into two groups, *WE–PS* and *PS–WE*. *WE–PS* cases are where students viewed worked examples,then solved problems, and then solved the post-test problem. *PS–WE* cases are where students solved problems, then viewed a worked example, and then solved the post-test problem. We omitted 21 outliers (out of a total of 1363 instances) where the elapsed problem time exceeded 1 hour and the mean time between steps was more than 5 minutes.

Two indicators of proficiency in logic proof solving are the number of rule-application steps used in solving a proof, and the elapsed time taken. Our initial results show no significant differences between the *WE–PS* and *PS–WE* groups for step count and elapsed time. However, in levels 3 and 4, we did observe a few interesting significant differences in these numbers, best illustrated when we used them to calculate the time taken per step on the post-test problem.

Table 5.6 Levels 3 and 4 post-test problem mean seconds per step, by tracks and ordering of practice problems within each level.

Ordering of Practice Problem Type	**Seconds per Step**	
High Proficiency Track	*Lvl 3*	*Lvl 4*
Worked Example then Problem Solving (WE–PS)	30.00	27.47
Problem Solving then Worked Example (PS–WE)	28.37	26.69
Low Proficiency Track	*Lvl 3*	*Lvl 4*
Worked Example then Problem Solving (WE–PS)	28.12	24.73
Problem Solving then Worked Example (PS–WE)	9.06	10.00

Table 5.6 shows the mean number of seconds per step taken on the last “post-test" problem in Levels 3 & 4 in DT3-Random, disaggregated by low or high proficiency track and the ordering of worked examples versus problem-solving encountered in the level before the post-test problem. Students in the high-proficiency track did not experience any ordering effects from worked examples, but that students in the low-proficiency track took significantly longer on each step in the post-test problem in both levels 3 and 4, if they saw worked examples before attempting to solve problems in these levels. These results indicate that there may be a disadvantage for low proficiency students to see efficiently-worked examples chosen from our corpus. This supports the idea that we have discussed in prior work: that data used for interventions should be chosen to match as closely as possible to the current student’s work. However, further analysis is required; another interpretation is that low-proficiency students who see worked examples first are taking more time to plan steps, which could be a positive effect. Yet another explanation could be that a small set of pondering

students experienced *WE–PS* in one level and *PS–WE* in the other, and are skewing all the results. Therefore, our hypothesis that the ordering of worked examples versus problem solving would impact student performance was not confirmed.

5.7 Conclusion

This chapter presented two data-driven methods of problem-solving feedback; hints and worked examples. Analysis of the results show that the addition of on-demand hints and data-driven worked examples improves student performance in the tutor by significantly increasing the amount of the tutor students are able to complete, significantly increasing the likelihood that students would finish the tutor. However, analysis of the results also show that, while there were a few significant differences in the amount of time students spend on a step in relation to the order they receive problems and worked examples, overall the order had no significant effect.

Students using DT3-Random complete 17% more of the tutor on average than students using DT2-MP, and 35% more students using DT3-Random completed the entire tutor. When analyzing the order of problems and worked examples, *WE–PS* versus *PS–WE*, in terms of step time and total elapsed time, there were no significant differences overall. There were some interesting effects that suggest that we may be able to generate better worked examples for low proficiency students. In conclusion, while the ordering of problems and worked examples did not make a significant difference, the addition of data-driven worked examples and on-demand hints improved student tutor completion, and reduced student dropout.

CHAPTER

6

STUDY THREE - DATA-DRIVEN PROFICIENCY PROFILING

Data-driven Proficiency Profiling

6.1 Research Question

Can we replace the expert-authored rule-priority proficiency calculation in DT2-MPs DDML system with a data-driven system that performs comparably to the original by automatically setting rule priorities and thresholds?

Since the current DDML system uses expert-decided priorities for each of the rule application actions when calculating a student's proficiency, any new problems or levels added to the system will require expert involvement to determine which rules were prioritized in each new or altered level.

This chapter describes a data-driven proficiency profiler (DDPP), which augments the data-driven mastery learning system from Chapters 4 and 5 by replacing expert-based proficiency calculation with a data-driven method.

6.2 Objective

To build a tutor that uses data-driven methods for between-level performance assessment, without the need for further expert involvement.

For this experiment, we created a test condition in DT5 by extending DT3-Random to replace the expert authored mastery settings with a data-driven system (DT5-DDPP) (see Table 6.1 to compare to previous versions). We created a data-driven problem profiler (DDPP), which uses the clustering of exemplar scores at each level interval for each rule, weighted by primary component importance, to classify exemplars into *types* of student progress through the tutor. New student performance will be compared to exemplars of the same classification, accounting for different problem solving strategies. This was expected to improve proficiency classification of students who used Deep Thought over the current expert-authored system, improving overall tutor scores. In addition, this could allow Deep Thought to be used in other classrooms where the pedagogical method and problem-solving ability of the class may be disparate from the current exemplar data from Deep Thought. This would eliminate the potential "cold start" in using the system in a different classroom setting before data from that class could be gathered.

The DDPP system was first compared alongside proficiency calculations using the minimum rule scores and average rule scores of exemplars, also weighted by primary component importance, to see how these methods compare to each other and to the expert authored system to determine the effectiveness of each method using data gathered from DT2 and DT3. During Fall 2015, a version of Deep Thought with the ensemble of best data-driven approaches was implemented (DT5-DDPP), and the results are compared to the students in DT3-Random and an identical version (DT5-Random) that was also run in Fall 2015.

6.3 Hypothesis

The use of data-driven proficiency classification using student performance data in DT5-DDPP will result in DT5-DDPP accurately calculating student proficiency earlier and more consistently than in DT3-Random or DT5-Random (as measured by the number of times the student changes proficiency paths in the middle of a level from where the system originally placed them, which would indicate that the system initially assigned a student to the incorrect proficiency path), while also resulting in higher completion and lower dropout.

The data-driven proficiency profiler (DDPP) presented will result in higher tutor completion scores

Deep Thought Version	Features
DT0	Basic Deep Thought Logic Tutor
DT1-NSH	Deep Thought with on-demand step level hints
DT2-MP	Deep Thought with the DDML implemented for problem selection
DT3-NoHint	Identical to DT2-MP except for the new user interface
DT3-Random	Deep Thought with the DDML and on-demand step-level hints and/or worked examples (depending on features enabled by the instructor)
DT5-Random	Identical to DT3-Random
DT5-DDPP	Identical to DT3-Random with the DDPP for problem selection

Table 6.1 Overview of Deep Thought versions 0-5

and lower dropout scores than the original DDML system regardless of the individual course curriculum and instructional method of the classroom in which the system is used (i.e. Computer Science versus Philosophy). The DDPP will result in as accurate predictions of student proficiency as an average/minimum of scores, and should perform at least as well as the original DDML system.

6.4 Approach

In the DDML system presented in Chapter 4, between-level assessment is performed by comparing rule-application scores for an individual student to threshold values for those rules, and then weighing the positive or negative differences by rule-priorities for the given level. These rule-priorities are determined by domain experts for DT2-MP and DT3 NoHint and Random. The priorities are weighted at three discrete levels: full priority (1), half priority (0.5), and no priority (0). At the conclusion of Study Two, enough performance data from DT3-NoHint and DT-Random exemplars (students who successfully completed the entire tutor) is available to determine the importance of required rules in a level for improved student performance. We use unsupervised learning methods to learn different strategies for solving particular problems. Clustering only the exemplars allows us to determine successful problem-solving strategies, as in the problem-solving strategies that lead to a student successfully completing the tutor, while avoiding the least successful strategies. This allows us to direct new students along a path that will most likely lead to those students successfully completing the tutor, as well as preventing the system from suggesting a worse path simply because more students overall used a particular problem-solving method.

6.4.1 Cluster-based Classification

Cluster-based classification has several advantages when applied to data-driven tutoring. First, new educational technologies may reveal unexpected learning behaviors, which may not yet been incorporated in expert-decided classification process. For example, Kizilec et al. [Kiz13] clustered MOOC learners into different engagement trajectories, and revealed several trajectories that are not acknowledged by MOOC designers. Second, experts classify with their perceptions on the average students' performance[Per12] [WR06]. This perception may be different from the actual participants group. Cluster-based classification, however, are able to classify and update classifications based on actual student behaviors.

Moreover, previous studies have shown that personalized tutoring based on cluster-based classification not only helps learning, but improves users' experience. Klasnja-Milicevic et al. [KM11] gave students different recommendations on learning content based on their classified learning styles. As a result students who used hybrid recommendation features completed more learning sessions successfully, and perceived the tutor as more convenient. Despotovic-Zrakic et al. [DZ12] adapted different course-levels, learning materials, and content in Moodle, an e-learning platform, for students in different clusters. Results showed that students with adapted course design had better learning gain, and more positive attitude towards the course.

However, the majority of previous work clustered students solely on their overall performance statistics. In contrast, our method clusters students based on their application of specific knowledge components throughout the tutor.

6.4.2 Data-driven Proficiency Profiler

The Data-driven Proficiency Profiler (DDPP) is a system which calculates student proficiency at the end of each level in Deep Thought based on how a given student performs in comparison to exemplars who employed similar problem solving strategies (see Fig. 6.1), with rule scores weighted as determined through principal component analysis. Based on how similar exemplars were assigned in subsequent levels, the DDPP can determine the best proficiency level for a new student. In contrast to the DDML system previously employed, this proficiency calculation and rule weighting is entirely data-driven, with no expert involvement. We hypothesize that this data-driven proficiency calculation will perform more accurately when compared to average and minimum methods.

Expert weighting was replaced by principal component analysis (PCA) of the frequency of the rules used for each exemplar for each level, accounting for 95% variance of the results. PCA is typically used to reduce the dimensionality of a data set by determining the most influential factors in the data set. The influence of a given factor is based on how much that factor contributes to the

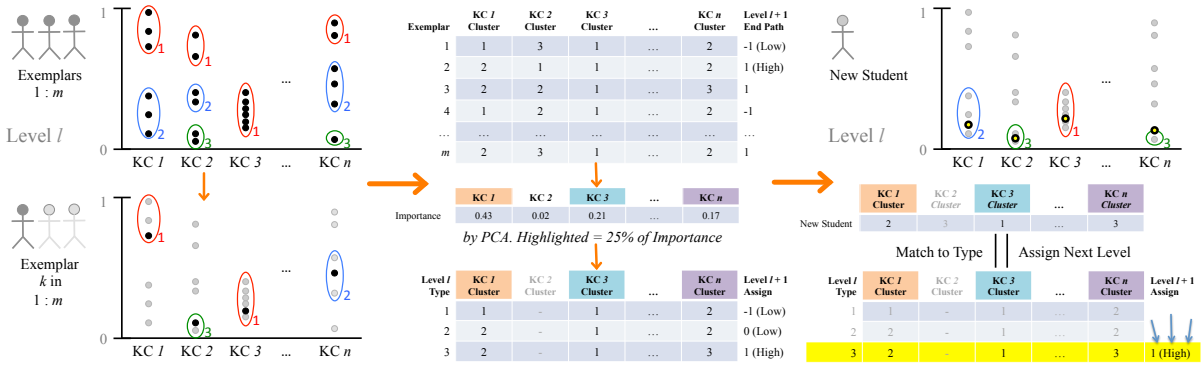

Figure 6.1 The Data-Driven Proficiency Profiler. (Left) At each level interval, exemplar KC scores are clustered, and exemplars are assigned a cluster for each KC Score. (Center) KCs that make up 25% of importance in the current level are used to assign exemplars to types. (Right) New student scores are assigned to clusters, and compared to existing types to determine next level path.

variability in the data. In our analysis, we use PCA analysis on the Deep Thought data set to determine which rules were most important to success in the tutor at each level. Rules which account for 25% of importance and higher are considered most important for completing a level. This percentage was determined through incremental testing at 5% intervals. From the results generated, 25% is the percentage that maximized accuracy. For each rule, its PCA importance value is the new weight for that rule score. Unlike expert authored weights, these rule score weights are based on each rule's importance as determined by the data.

New students in the system will have their KC scores labeled by the clusters their scores fall into (distance to closest centroid). New students will then be compared to exemplars to find which Type the new student has the most clusters in common. The new students will then be assigned the proficiency track in the next level in Deep Thought of that Type. In the case of multiple matching Types, the new students will be assigned the Type that is weighted higher by frequency. To classify a student an existing type, they must match at least half the clusters of any one Type. Otherwise, the student is considered to have no match, and will be given the low proficiency track in the next level.

In the original system, the student proficiency was determined based on one set of rule thresholds and a set of expert authored weights. However this means the system did not take varying student strategies into account. The data is based on students who completed the tutor, who have therefore shown the level of mastery required to successfully complete Deep Thought, but the scores are averaged over all the students at the end of each level. By taking the average of these student scores at this point, we're still assuming only one successful problem solving strategy for completing each level in the tutor. However while most strategies might be the same for earlier levels, there may be a

variety of strategies in later levels that can still result in successful completion.

The DDPP method accounts for that possible variety in problem solving methods. In using an unsupervised clustering method, we're able to account for different clusters while not knowing how many clusters there are. By clustering the scores, we're essentially looking for different strategies that utilize particular rules and determining these strategies based on the student data. Once we determine which strategy a new student is utilizing, we can look to the data again to see how exemplars who employed a similar strategy were placed in the tutor and how they performed, thus determining the best way for the tutor to react to that particular student. Finally, using PCA based weights allows us to weight rule scores based on rule importance as determined by previous students who completed the tutor, rather than expert determination.

6.5 Method

This study consists of two stages: a comparison of data-driven methods using historical data, and implementation of those methods in Deep Thought for students to use. The data used is from the same population of students as the studies in chapters 4 and 5. For the first part of the study, we select students who used the original DDML system (DT2-MP) and the data-driven methods for proficiency calculation (DT2-MP from Spring 2013 $n = 47$, DT3-NoHint from Ilson's Fall 2014 course, $n = 47$). For the second part of the study, we will use performance data from DT3-Random, which was collected from the two computer science discrete mathematics course in Fall 2014, ($n = 261$) and DT3-NoHint($n = 47$) to compare the effectiveness of implementing data-driven proficiency calculation into Deep Thought (DT5).

6.5.1 Study Stage 1: Comparison of Methods

The DDPP method is compared to results from thresholds set based on the average rule score and minimum rule score. The rule thresholds will be weighted based on PCA scores, replacing the expert-decided rule priorities. The average rule scores are the set of average scores for each rule in each level, while minimum scores are the smallest scores for each rule. Comparing the current DDML system with expert-decided rule priorities and average score or minimum score based proficiency calculation weighted by PCA score offers insight into the effect of introducing PCA based rule weighting to students' performance base-line on the prediction accuracy. From here, the methods will be distinguished as follows: the DDML method in DT2-MP will be referred to as DT2-MP (MP in this case would stand for *Manual Proficiency* determination, to distinguish it from the data-driven proficiency calculation in the DDPP); the proficiency calculation based on average

scores will be referred to as AEP (for *Average Exemplar Proficiency*); the minimum score method will be referred to as MEP (for *Minimum Exemplar Proficiency*); finally, the DDPP calculation as CEP (for *Clustered Exemplar Proficiency*).

The first stage of the experiment is to compare the results of calculating student proficiency using the CEP, DT2-MP, AEP, and MEP methods with existing data from DT2-MP and DT3-Random. Exemplars for clustering are chosen from the students who started and completed a given level of Deep Thought on the proficiency track that they were assigned.

Data is used from students using the unaltered DDML system (DT2-MP Spring 2013 $n = 47$, DT3-NoHint Fall 2014 $n = 47$). We perform a 10-fold cross-validation to compare the estimation of proficiency level for students using three methods: CEP, AEP, and MEP.

6.5.2 Study Stage Two: Implementation into Deep Thought

The DDPP was augmented with additional exemplars from the pool of students who used Deep Thought in Spring 2015. Exemplars from the Spring 2015 class were added to the existing dataset of exemplars. This dataset was then clustered based on the same hierarchical clustering methods for the DDPP described above. The DDPP was then implemented into Deep Thought as DT5-DDPP.

This version of Deep Thought was used as a mandatory homework assignment in an undergraduate computer science discrete mathematics course in Fall 2015. Students were required to solve 1-3 problems per level depending on proficiency, another 1-2 problems were randomly presented as worked examples. The median number of problems students were required to solve was 12. As students used Deep Thought, their actions and progress were recorded. The study results encompass three test groups; the DT5-Random control group ($n = 65$); the DT3-Random control comparison group($n = 47$); and the DT5-DDPP group (DDPP, $n = 27$). Students in all three groups were taught by the same instructor, using the same curriculum. We examine data from the DT3-Random group in order to analyze the consistency in the DDML's performance between years.

6.5.3 Statistics & Metrics

6.5.3.1 Path Prediction Accuracy

Path prediction accuracy is based on the possibility that a student may not finish a level in the same proficiency path that the system originally assigned them to due to skipping a certain number of times within a level. If the student did not finish the level on the same proficiency path, it is an indication that the proficiency calculation system may have initially assigned the student to the wrong proficiency path. Therefore we can calculate the accuracy of the original system by determining

how often students who completed the entire tutor changed proficiency paths throughout. Given $S_{sameTrack}$ as the number of students who finished a level on the same proficiency track, and S_{total} as the total number of students who completed the level, the path prediction accuracy for each level ($LevelAccuracy$) is calculated as follows:

$$LevelAccuracy = \frac{S_{sameTrack}}{S_{total}} \tag{6.1}$$

6.6 Results

6.6.1 Stage 1: Comparison of Methods

The prediction accuracy of the MEP, AEP, and CEP methods were calculated for the 76 exemplars that completed the tutor. Ten-fold cross validation was used to train and test the methods across the combined data. We focus on the results of the path prediction accuracy described in the previous section as a basis of comparison between DT2-MP, CEP, AEP, and MEP. These results are in tables 6.2, 6.3, and 6.4.

6.6.1.1 Path Prediction Accuracy

Table 6.2 shows the path prediction accuracy of the DT2-MP, CEP, AEP, and MEP assesments across all the students in the Philosophy and CS courses. The DT2-MP accuracy was very high, ranging from 75% at the end of level 3 to 88.2% at the end of level 1. CEP was somewhat accurate, ranging from 61.8% path prediction accuracy at the level 4 interval to 67.1% path prediction accuracy at level 2. While these accuracies are not nearly as high as in the original system, they are very good considering that, unlike the original system, path prediction in the DDPP is entirely data-driven. It should also be noted that the DDPP was more consistent in its accuracy, only varying by at most 5% between levels (in comparison to the original DDML system, which ranged in accuracy by 9.3%).

Overall the original DDML system with expert-set thresholds for skills (DT2-MP) predicted paths more accurately than the CEP, AEP, or MEP methods across all levels. The MEP method was least accurate across all levels. This makes sense, considering that the MEP method is based on the minimum thresholds, so several students were assigned to the higher proficiency track when it is likely they should have been assigned to the lower proficiency track. The CEP method was more accurate than the average AEP method on level 3, and level 5. The CEP was equally as accurate as the average AEP method at the end of level 1, and less accurate at the end of levels 2 and 4. However, some of the lower accuracy was likely due to the distribution of exemplars across the two courses. Recall that the CS students made up a higher proportion of the analyzed exemplars than

Table 6.2 Path prediction accuracy of the original DDML system, (DT2-MP) the DDPP system (CEP), average score assessment (AEP), and minimum score assessment (MEP), for both Philosophy and CS students at the end of each level

	DT2-MP	CEP	AEP	MEP
Lvl 1	**88.2%**	65.8%	65.8%	35.5%
Lvl 2	**85.5%**	67.1%	73.7%	18.4%
Lvl 3	**75.0%**	63.2%	60.5%	69.7%
Lvl 4	**78.9%**	61.8%	64.5%	40.8%
Lvl 5	**78.9%**	64.5%	59.2%	59.2%

the Philosophy students. Analyzing the path prediction accuracy by the individual course reveals more detail on the path prediction accuracy.

6.6.1.2 Philosophy & CS Accuracy

In the case of the Philosophy students, where proportionally fewer of the students were selected as exemplars, the CEP method was more accurate than the original system on every set of levels except for level 5 (see Table 6.3). In comparison to the average calculation method, the CEP method was only more accurate on level 3. In level 1 and 5, the CEP was as accurate as the AEP method, and in levels 2 and 4 the CEP was less accurate.

Table 6.3 Path prediction accuracy of the original DDML system (DT2-MP), the DDPP system (CEP), average score assessment (AEP), and minimum score assessment (MEP), for Philosophy students

	DT2-MP	CEP	AEP	MEP
Lvl 1	76.9%	**80.8%**	**80.8%**	23.1%
Lvl 2	65.4%	69.2%	**76.9%**	19.2%
Lvl 3	50.0%	**84.6%**	80.8%	38.5%
Lvl 4	65.4%	69.2%	**76.9%**	30.8%
Lvl 5	**53.8%**	46.2%	46.2%	26.9%

In the CS course, where proportionally more of the students were selected as exemplars, not only was the original system far more accurate than it was for the entire set of students overall, but the DDPP path accuracy was much worse in some places. However, in comparison to the average method, the DDPP method was only less accurate in level 2. In all other levels the DDPP was either more accurate than the average method (levels 3 and 5) or equally as accurate (levels 1 and 4).

Table 6.4 Path prediction accuracy of the original DDML system (DT2-MP), the DDPP system (CEP), average score assessment (AEP), and minimum score assessment (MEP), for Computer Science students

	DT2-MP	CEP	AEP	MEP
Lvl 1	**94.0%**	58.0%	58.0%	42.0%
Lvl 2	**96.0%**	66.0%	72.0%	18.0%
Lvl 3	**88.0%**	52.0%	50.0%	86.0%
Lvl 4	**86.0%**	58.0%	58.0%	46.0%
Lvl 5	**92.0%**	74.0%	66.0%	76.0%

6.6.2 Stage 2: Implementation into Deep Thought

There were no significant results between the DT5-Random group and the DT5-DDPP group. However there were some interesting trends in tutor dropout, rule application scores, and path prediction accuracy that indicate the DDPP's ability to improve and perform well with additional data.

6.6.2.1 Path Prediction Accuracy

Table 6.5 shows the path prediction accuracy of the DT5-DDPP and the DT5-Random group in comparison to the path prediction accuracy of the DDPP and the DDML at the time of stage 1 of the study (DDPP(E) and DDML(E) respectively). As previously mentioned, path prediction accuracy is based on the possibility that a student may not finish a level in the same proficiency path that the system originally assigned them to due to skipping a certain number of times within a level.

There was improvement in the path prediction accuracy in the DDPP between the first stage of this study and the second stage. In almost every level, the path prediction accuracy improved by 23-30%. Additionally, while the DDPP in the first stage ranged from 52-74% accuracy, the DDPP in the second stage was never less than 80% accurate. The DDML path prediction accuracy did not change significantly between stage 1 and 2, which was expected. In Levels 3 and 5, the DDML was slightly less accurate than the DDPP. This confirms our hypothesis that additional data would improve the DDPP's performance and enable it to calculate student proficiency much more accurately.

Table 6.6 shows the percentage of students assigned to the high proficiency path and the low proficiency path at each level. The DDPP assigns students to the lower proficiency path in the event that a student cannot be assigned to a type, so it is expected that a higher percentage of the students using the DDPP would be assigned to the lower path than the students using the DDML. This expectation is supported by the data, 7-48% of the students using the DDPP were assigned to the lower proficiency path. In comparison 1-13% of the students using the DDML were assigned to the lower proficiency path.

Table 6.5 Path prediction accuracy of the DDPP and DDML in the new study, and the accuracy of the DDPP and DDML in the exploratory study (E) at the end of each level.

	DDPP	**DDML**	**DDPP(E)**	**DDML(E)**
Lvl 1	88.9%	93.8%	*58.0%*	*94.0%*
Lvl 2	92.6%	93.8%	*66.0%*	*96.0%*
Lvl 3	81.5%	75.4%	*52.0%*	*88.0%*
Lvl 4	81.5%	89.2%	*58.0%*	*86.0%*
Lvl 5	100%	89.0%	*74.0%*	*92.0%*

Table 6.6 Path placement percentage of the DT5-DDPP and DT5-Random students at the end of each level.

	DT5-DDPP		***DT5-Random***	
Path	**High**	**Low**	**High**	**Low**
Lvl 1	63.0%	37.0%	97.0%	3.00%
Lvl 2	51.9%	48.1%	86.2%	13.8%
Lvl 3	92.6%	7.40%	98.5%	1.5%
Lvl 4	85.2%	14.8%	90.8%	9.2%
Lvl 5	69.2%	30.8%	95.3%	4.7%

Table 6.7 shows the number of types identified in each level, the number of those types that were matched, and the percentage of students that were successfully assigned to a type. These metrics are an indication of the DDPP's ability to match new students to a type. Overall the percentage of students assigned to a type was very high, between 82% and 100%. The number of types found increased steadily over the course of the tutor, from 14 types in Level 1 to 44 types in Level 5, indicating an increasing number of possible problem solving types as the tutor problems are more difficult. The number of those types that were matched ranged from 9-15.

6.6.2.2 Student Dropout

Students are exposed to more advanced concepts and more difficult rule applications the further they get into Deep Thought. It is therefore beneficial for as many students to complete as much of the tutor as possible before dropping out. Students are more likely to perform better in-tutor when given problems in their zone of proximal development through scaffolding of major concepts [MA02]. Proper scaffolding increases learning, and therefore increases the likelihood of further progression through the tutor and exposure to higher-level concepts. Figure 6.2 shows the rate and level of student dropout, shown via the percentage of students assigned the tutor that were still using it at any given level. The DDPP resulted in the lowest dropout rate in comparison to

Table 6.7 The number of types identified per level from historical data (DDML), the number of types matched with students using the DDPP system, and the percentage of students who were matched an existing type in the DDPP system.

Level	**1**	**2**	**3**	**4**	**5**
Types Found (DDML)	14	17	19	22	44
Types Matched (DDPP)	10	9	11	9	15
% Students Matched (DDPP)	100	96.3	82.4	94.1	89.2

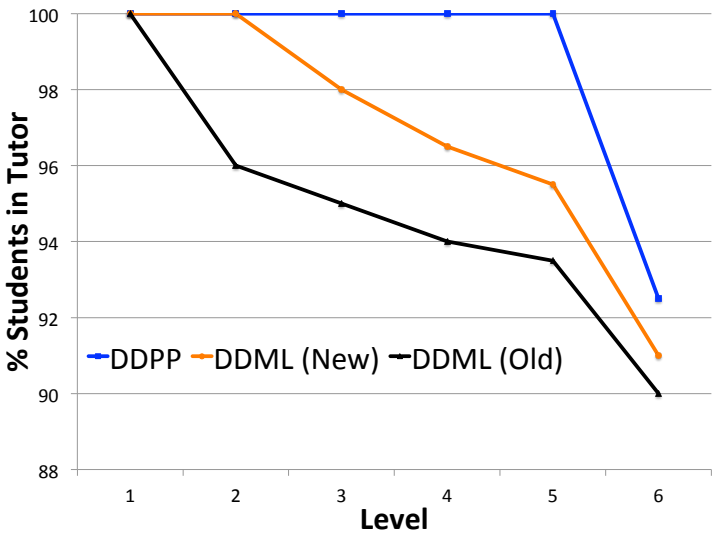

Figure 6.2 The percentage of students at the end of each level of the tutor.

the DT3-Random and DT5-Random iteration of the DDML Additionally, all of the students in the DT5-DDPP group stayed in the tutor until Level 6, where roughly 7% of the students stopped. In the DT5-Random group, the students started dropping out around Level 3, while in the DT3-Random group, the students started dropping out as early as Level 2. As a result, a far higher percentage of the students using the DDPP were exposed to the advanced concepts introduced in the problems in later levels of Deep Thought than any set of students using Deep Thought with the DDML alone. A higher percentage of the students using the DDPP also completed the entire tutor in comparison to the students using the DDML in earlier semesters or the current semester. 93% of the students using the DDPP completed the entire tutor, while only 91% of the students using the DDML in this current study and 90% of the students using the DDML in previous semesters completed the entire tutor.

6.6.2.3 Rule Application

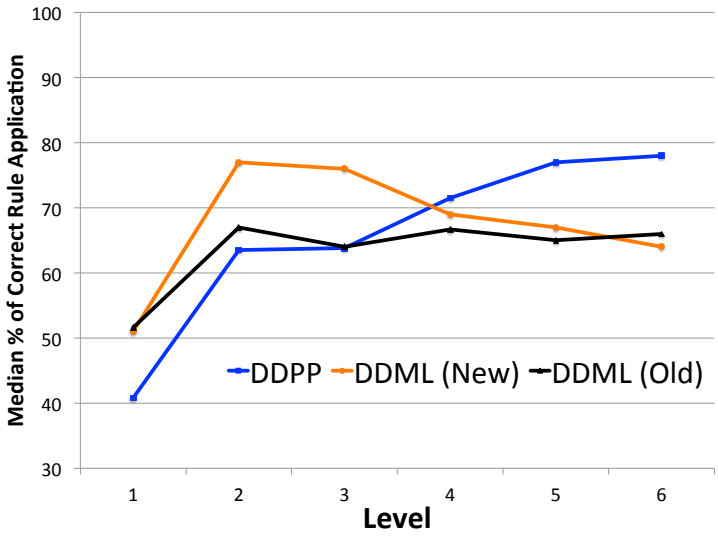

Figure 6.3 The median percentage of correct rule application in each level of the tutor. Each level's calculation is independent of the other levels.

Figure 6.3 shows the median percentage of correct rule applications at each level of the tutor. A correct rule application is when a student uses a logical rule in the correct way in the proof. We track the change in percentage in correct rule applications in order to track whether students are applying rules more accurately as they progress through the tutor. The DDPP, new DDML, and old DDML all showed an increase in the correct percentage of rule applications between Levels 1 and 2. After Level 2 the percentage decreases, and then levels out, for students in this or the previous study using the DDML. For students using the DDPP on the other hand, there was a steady increase in the percentage of correct rule applications as they progress through the tutor. This steady increase in understanding is supported by the post-test results. The median Post-Test Scores for DT5-Random is 75%, for DDPP is 87.5%.

6.7 Discussion

The first stage of the study showed some potential for the DDPP based on path prediction accuracy, but it was far less accurate than the DDML. The DDPP path prediction accuracy at the time of the

first stage of the study ranged from 52% to 74%, compared to the DDML at that timewhich ranged from 86% to 96%. Since the DDPP was augmented with additional student data however, the path prediction accuracy increased to a range of 81.5% to 100%. This indicates that the DDPP was able to place students on a proper proficiency path much more often once there was additional data, as expected.

Looking at the number of types found, matched, and the percentage of students that were matched shows additional indications of the DDPP's improved performance. In the first stage, there were only at most 26 types found, and the number of those types that were matched to students in the test set ranged from 0 to 10. In the second stage, there are a couple of notable changes. The number of types found at each level increased overall, indicating that the DDPP was able to find a greater variety of problem solving strategies. This is especially noticeable at Level 5, where 44 types were found by the DDPP as opposed to 26 in the exploratory study. This high number is expected at this level, since students are more likely to take a variety of possible approaches to solving more difficult problems. The other notable change is that more of the extracted types are matched with students. The range of types matched is 9 to 15, indicating that the DDPP was able to find types that correspond to the way multiple students solve problems and was able to detect those types in new students. Proportionally, however, there are still very few types matched to a student, particularly in the later levels. Level 4 only matched 9 out of 22 possible types, and Level 5 only matched 15 out of 44 types. However the percentage of students that were matched was very high throughout all levels of the tutor. This would indicate that a majority of the students are being assigned to a proportionally small number of types. This result reflects analysis by Eagle et al [Eag15], that showed that problem-solving interaction networks in Deep Thought and other environments are scale-free - meaning that only a handful of strategies are used frequently.

Considering the rate of tutor dropout and the rule application accuracy throughout the levels, the DDPP was beneficial overall. None of the students using the DDPP in DT5 dropped out of the tutor before Level 6, so more of those students were able to be exposed to the advanced concepts presented in the later level problems. This dropout occurs much later than in the DDML groups. The students in the new DDML group started to drop out around Level 3, while students in the old DDML group started to drop out around Level 2. This indicates that the DDPP was able to properly assign students to the proficiency path that contained problems close to their proximal development in terms of difficulty, presumably reducing the frustration that would cause students to stop their progress in the tutor prematurely.

The rule application accuracy shown in Figure 6.3 indicates that students using the DDPP increased their understanding of logic rules. All the groups show an increase in rule application accuracy from Level 1 to Level 2. This is expected considering that students have a better under-

standing of the tutor after the first level, and are less likely to make general usage errors at that point. However, whereas the students in the DT3-Random and DT5-Random groups maintained a steady level of rule application accuracy throughout the rest of the tutor, the students in the DT5-DDPP group increased the percentage of accurate rule applications throughout the tutor, to the point where those students had a higher median percentage of correct rule applications than either of the DDML groups. This confirms that the DDPP led the students through the tutor in a way that enabled them to learn concepts and improve on how they apply them.

We analyzed the data from students in the DT3-Random group in order to determine the consistency in the trends and performance of the DDML over the course of a year. In this case there were also no significant differences between the DT5-Random and DT3-Random groups in relation to any of the metrics. However, the trends across levels in terms of dropout and rule application accuracy are consistent between the two versions. This speaks to the DDML's consistency in performance, and its ability to maintain that consistency across identical instructors and curricula with a different population of students.

6.8 Conclusions

We have conducted a two-stage study in order to determine the performance of our data-driven proficiency profiler (DDPP) in comparison to a data-driven mastery learning system based on average score thresholds and expert-determined rule priorities. We determined based on the first stage of the study that the DDPP's performance would improve with additional data. This was the first semester where we could test the effectiveness of the DDPP with additional data and fully incorporated into Deep Thought. In terms of the DDPP's performance, the results were not statistically significant between the DDPP group and either of the DDML groups. However, the trends in student dropout and rule application accuracy indicate that the DDPP performs at least as well as the DDML in guiding students through Deep Thought so they reach later levels and more advanced concepts, while gaining greater understanding of the knowledge components in deductive logic. The improvements in path prediction accuracy, types matched, and percentage of students matched to a type between the DDPP at the time of the exploratory study and the DDPP incorporated into Deep Thought this semesters confirms that the DDPP does improve with additional data. Overall, the results demonstrate that the DDPP could feasibly replace the current proficiency calculation based on the average rule scores and expert-determined rule priorities, making Deep Thought a completely data-driven intelligent tutor.

CHAPTER

7

EXPLORING THE IMPACT OF DATA-DRIVEN METHODS ON STUDENTS' DEMONSTRATIVE KNOWLEDGE

Exploring the Impact of Data-driven Methods on Students' Demonstrative Knowledge

7.1 Research Question

How do specific data-driven methods impact student performance in post-tests?

Can we differentiate high and low post-test students based on their behavior in the tutor to detect the need for intervention, rather than letting students spend time but still perform poorly on post-tests?

Deep Thought Version	Features
DT0	Basic Deep Thought Logic Tutor
DT1-NSH	Deep Thought with on-demand step level hints
DT2-MP	Deep Thought with the DDML implemented for problem selection
DT3-NoHint	Identical to DT2-MP except for the new user interface
DT3-Random	Deep Thought with the DDML and on-demand step-level hints and/or worked examples (depending on features enabled by the instructor)
DT5-NSH	Identical to DT3-NoHint with the addition of next-step hints
DT5-Random	Identical to DT3-Random
DT5-DDPP	Identical to DT3-Random with the DDPP for problem selection
DT6-Random	Identical to DT5-Random
DT6-DDPP	Identical to DT5-DDPP

Table 7.1 Overview of Deep Thought versions 0-6

7.2 Objective

The studies presented in Chapters 4, 5, and 6 have shown that the addition of data-driven hints, worked examples, and problem assignment can improve student performance and retention in the Deep Thought tutor. While improvements in student retention and tutor scores have been observed with each of these additions, the difference in post-tutor examinations when these methods are combined in different test conditions has not been studied. This chapter explores how the specific methods of problem assignment and combination of hints and worked examples may have impacted student performance on related questions on the course midterm exam. The results from this chapter also help differentiate the behavior of higher and lower performing students in tutor, allowing for quicker interventions where a student requires additional instructional feedback and support, giving students a higher chance of success in learning the material required for their course knowledge.

7.3 Methods

Deep Thought was used as a mandatory homework assignment by students in an undergraduate “discrete mathematics for computer scientists" course in Fall 2015 and Spring 2016. For this exploration, the data from three Deep Thought test conditions (DT5-NSH, DT5-Random/DT6-Random, and DT5-DDPP/DT6-DDPP) were used across the two semesters to explore the impact of our data-driven methods on student performance.

The first group evaluated for this study were assigned only problem-solving opportunities (DT5-NSH, hereafter referred to as the PS group, $n = 26$). The problem assignment system used was the DDML system described in Chapter 4, where students were assessed between levels and placed on either a high or low proficiency track in the next level, based on how they compared with expert-set rules and thresholds for the current level.

The second group of students were randomly assigned either problem-solving opportunities or worked examples of the same problems within each level (DT5-Random/DT6-Random, hereafter referred to as the PS/WE group, $n = 179$), with the number of problem-solving opportunities controlled to match the number of problems solved by the PS group, as described in Chapter 5. Like the PS group, the PS/WE group were assigned proficiency tracks using the DDML. However, because individual rule application scores were updated during the use of worked examples in addition to problem-solving opportunities, most students were consistently assigned to the high track in most levels, and were only assigned the low track when their individual performance was below satisfactory. This group of students were taken from both the Fall 2015 and Spring 2016 semesters.

The third group of students were randomly assigned problem-solving opportunities or worked examples in the same manner as the PS/WE group, but with the DDPP method assigning proficiency tracks instead of the DDML, where students were assigned the same proficiency track as prior students who most closely matched their rule application behavior (DT5-DDPP/DT6-DDPP, hereafter referred to as the DDPP group, $n = 61$). This group of students were also taken from both the Fall 2015 and Spring 2016 semesters. Students in all three groups had access to on-demand next-step hints.

All students were evaluated using two proof problem questions as part of a mid-term examination, which was used as a post-test for this study. The post-tests for both Fall 2015 and Spring 2016 were graded by the same teaching assistant, ensuring consistent evaluation across semesters. Students were separated for evaluation by performance on the post-test and by the predominant track level in Deep Thought. The post-test was a set of two proofs students had to solve on paper for a midterm exam. These questions were hand-graded with partial credit given based on the percentage of the proofs completed and points taken off for misapplication of rules and skipping non-trivial rules. Two performance levels were considered: post-test scores greater than or equal to 80% (AB), or less than 80% (CDF). The post-test scores mark the final evaluation of students' ability to solve proof problems, and occurs immediately following the Deep Thought tutor homework assignment.

The second dimension studied was the proportion of high to low proficiency track levels the students completed. Students who were assigned to the high proficiency track in a level had the ability to finish on either the high or low proficiency track depending on the number of problems skipped within that level. Students who completed more levels on the high track than the low track

were marked as high track students, and students who completed more levels on the low track than the high track were marked as low track students. The track assignments indicate the number and difficulty of problems students received, with the low track having more problems of lower difficulty, and the high track having fewer problems of higher difficulty. The tracks were designed so that students would have a similar number of rule applications across the tracks, even though the number of problems differs. Typically, the low track has three problems with expert solutions using 5 rule applications, and the high track has 2 problems with expert solutions using 7 – 8 rule applications - meaning that both tracks minimally required about 15 total rule applications (though students typically used more).

In addition to post-test and predominant track level, total time in tutor, average time spent per problem, percentage of correct rule applications out of all rule applications, and the total number of rule applications were examined. Ancillary behaviors (hint usage, skipped problems, and reference requests) that could differentiate high and low performing students were also explored. These metrics were compared to better understand the impact of worked examples, hints, and data-driven track selection on student performance. The results of this descriptive analysis are presented in the next section.

7.4 Results

Table 7.2 displays the percentage of AB students in each group and track. Table 7.2 also displays the percentage of all students (including those scoring AB and CDF on the post-test) in each group and track who dropped out of the tutor before full completion. In DT3-NSH, it was found that students completed 94% of the tutor on average, with a retention rate of 90%. The average percent tutor completion for the groups in this exploration are similar (PS: 95%, PS/WE: 93%, DDPP: 94%).

The percentage of students who scored AB on the post-test was higher for the PS (65%) and DDPP (64%) groups than for the PS/WE group (50%), overall and for both tracks. In the PS and PS/WE groups, a higher percentaage of high-track students scored A or B on the post-test than low track students (71% vs 63%, for high and low-track students in the PS group, and 52% vs 37% for high and low-track students in the in the PS/WE group).

On the other hand, the DDPP group more consistently resulted in students scoring AB regardless of track (66% vs 61% receiving AB scores in the high and low tracks respectively). The DDPP group also had a higher percentage of students receiving AB scores on the post-test than the PS/WE group. A Kruskal-Wallis test for one-way analysis of variance showed no significant difference between groups ($p = 0.22$).

The bottom half of Table 7.2 shows the percentage of all students in the tutor who dropped

Table 7.2 Percentage of AB Students and dropped students in PS, PS/WE, and DDPP groups.

Condition		ALL	High Track	Low Track
	n	*% AB Students*		
PS	*26*	65.38	71.43	63.16
PS/WE	*179*	49.72	52.35	36.67
DDPP	*61*	63.93	66.67	61.76
	n	*% Dropped Students*		
PS	*26*	3.85	0.00	5.26
PS/WE	*179*	11.73	6.71	36.67
DDPP	*61*	9.84	11.11	8.82

out, by group and track. Students had a lower dropout rate in both the PS (4%) and DDPP (10%) groups compared to the PS/WE group (12%). Low track students in the PS/WE group had a much higher drop rate than all other groups in the study (37%). Because the tutor gave credit for worked examples, PS/WE group were usually placed in the high track. Therefore, high track placement was not necessarily a marker of high proficiency, but low track placement likely does indicate low proficiency. Students in the PS/WE group on the low track skipped many problems, or demonstrated low alignment of rule usage with that needed for the given problems. Based on the high dropout rate for low-track PS/WE students, and the low AB rate for low-track PS/WE students, these low-performing students who are not intelligently assigned problems based on their problem-solving performance appear to gain little from worked examples.

While it may be tempting to declare problem-solving opportunities with no worked examples as the best performing pedagogical choice among the three groups based on these numbers alone, a look into additional performance metrics gives some more insight. Table 7.3 presents the amount of time spent in tutor and on each problem, as well as the percentage of correct and total rule applications for each group, separated by track. The numbers presented are the median values for each metric, since the distributions of scores were highly skewed and non-normal, and none of the differences were significant due to low sample size within each subgroup.

As shown in Table 7.3, among AB students in all three groups, the total time spent in tutor appears similar, although the mean time for high-track students was lower for DDPP (M = 3.95hr, SD = 6.21hr) compared to PS/WE ($M = 4.46hr$, $SD = 9.13hr$) and PS ($M = 6.66hr$, $SD = 9.91hr$). The mean time for low-track students was lower for PS ($M = 4.63hr$, $SD = 9.55hr$) and DDPP ($M = 5.48hr$, $SD = 5.42hr$) than the PS/WE ($M = 7.74$, $SD = 9.76$). The means of average problem time, percentage of correct rule applications, and number of rule applications were consistent with the median values presented in Table 7.3 across all three groups. Note that low-track students in

Table 7.3 Total Time, Average Problem Time, Percentage of Correct Rule Applications, and Total Rule Applications for AB and CDF students in the PS, PS/WE, and DDPP groups, separated by High and Low Track. The numbers listed are all median values.

		AB STUDENTS				CDF STUDENTS		
		PS	**PS/WE**	**DDPP**		**PS**	**PS/WE**	**DDPP**
HIGH TRACK	***n***	*5*	*78*	*18*	***n***	*2*	*71*	*9*
Total Tutor Time (hr)		2.47	2.37	2.80		12.8	3.75	3.17
Average Problem Time (min)		9.89	11.1	12.1		52.3	18.4	16.0
% Correct Rule Applications		60.8	63.5	58.5		64.1	56.9	62.3
Total Rule Applications		258	214	203		471	255	204
LOW TRACK	***n***	*12*	*11*	*21*	***n***	*7*	*19*	*13*
Total Tutor Time (hr)		1.80	3.33	3.67		17.2	5.96	4.98
Average Problem Time (min)		6.76	15.2	15.0		60.1	25.0	30.4
% Correct Rule Applications		68.8	45.5	57.0		48.7	45.7	47.0
Total Rule Applications		201	404	291		382	394	389

the PS/WE groups had the lowest percentage of correct rule applications, and the highest number of total rule applications among all the groups. This means they are doing more work, but a lower percentage of it is correct.

As shown in Table 7.3, among CDF students in all three groups, the total time spent in tutor is dramatically different, with PS spending 3 to 4 times as long in the tutor than PS/WE and DDPP groups. This ratio is also similar in the average problem time for high and low track students, and the number of total rule applications for high track students. Therefore, while problem-solving only (PS) may have a slightly higher overall success rate in helping students learn proof problem solving and remain in the tutor than the DDPP students, for students who are less prepared, PS results in a much higher time spent in the tutor, with little return on the time investment. Therefore, for students who have a better grasp of the subject matter, pure problem-solving may offer a slightly better option for getting through the assigned tutor, although the differences between problem solving, problem solving and worked examples, and proficiency profiled assigned problem solving and worked examples are minimal. However, for less prepared students, pure problem-solving opportunities offer little to guide students to higher understanding of the material, and in general, the DDPP offers a much better path to completing the tutor in far less time for both AB and CDF students, giving students the opportunity to encounter all the subject matter and have a greater chance of learning the material, resulting in higher overall post-test scores.

Completing the tutor assignment is important for students; however, since it is desired that students are learning the material well, mid-term examination scores are ultimately a higher gauge

Table 7.4 Number of Hints, number of Skips, and number of Rule Reference requests for AB and CDF students in the PS, PS/WE, and DDPP groups, separated by High and Low Track. The numbers listed are all median values.

	PS		**PS/WE**		**DDPP**	
HIGH	**AB**	**CDF**	**AB**	**CDF**	**AB**	**CDF**
# Hint	95	166	12	26	17	32
# Skip	5	16	1	1	0	2
# Ref	151	168	76	145	111	92
LOW	**AB**	**CDF**	**AB**	**CDF**	**AB**	**CDF**
# Hint	30	104	31	44	19	26
#Skip	1	0	30	24	3	15
# Ref	77	224	60	271	55	109

for learning success. Among all the experimental groups in this study, at most 65% of students were performing at A or B grade level on the mid-term examination. Increasing this percentage of AB students is desired, so the question at this point is: Is it possible to predict low exam scores based on in-tutor data for early intervention?

The differences between AB and CDF students is explored in Table 7.3, with the assumption that the DDPP method offers the best overall chance of success for students. For high track students, total tutor time, average problem time, percentage of correct rule applications, and total rule applications are consistent between AB and CDF students. However, for low track students, average problem time, percentage of correct rule applications, and total rule applications show a higher difference. CDF students spent twice as long on average per problem than AB students, and applied rules correctly less than half of the time, while AB students applied rules correctly more than half of the time. CDF students also attempted applying rules 25% more overall than AB students.

Since the performance differences between AB and CDF students are not as apparent for high track students, ancillary tutor behaviors are explored to make a better distinction. Table 7.4 shows the number of requested hints, the number of skipped problems, and the number of rule reference requests (descriptions of logic rule operations) made by students in all groups. For the DDPP group, the most apparent difference among AB and CDF students are the number of hints requested, with the CDF group requesting 32 hints ($M = 50$, $SD = 57$) compared to 17 ($M = 32$, $SD = 42$) for the AB group. This difference in hints requested between AB and CDF students is also consistent across all groups and both high and low track students. Therefore, for high track students, the differentiation between higher and lower proficiency students can be made using hint request behavior, and for low track students, the differentiation between higher and lower proficiency students can be made using the average amount of time spent per problem and the percentage of correct rule applications.

One possible intervention based on these findings is to suggest that a student visit office hours or join a study group, if their hint usage on the high track is high, or if their problem time or incorrect rule applications are too high on the low track.

7.5 Conclusion

In this chapter I compared two classrooms of students using different test conditions of Deep Thought, with different combinations of problem assignment (DDML or DDPP) and the addition of worked examples, for the purpose of understanding how the specific methods of problem assignment and combination of hints and worked examples affect high and low performing students, as evaluated using mid-term examination (post-test) scores. I found that for high post-test (AB) students who have a firmer grasp of the subject matter, problem-solving only offers the best chance of completing the tutor in a timely manner; however, the addition of worked examples does not significantly increase the amount of time spent in the tutor. The method of problem assignment (DDML or DDPP) does not have a noteworthy effect on AB student performance.

For lower post-test (CDF) students, I found that problem-solving opportunities alone with DDML problem assignment offered little to guide students to higher understanding of the material, and greatly extended the amount of time students spent in the tutor with little learning benefit. The addition of worked examples helped these students get through the tutor faster, however these students had a higher drop rate than any other students and lower examination scores. These results demonstrate that updating our data-driven skill estimates equally for viewing or applying rules resulted in students being assigned to the high-track when they were not prepared to solve harder problems. With data-driven proficiency profiling (DDPP) – matching students to previously successful students and the paths they take through the tutor – the amount of time spent in tutor can be reduced, retention can be increased, and worked examples can be better utilized by giving them alongside problems that better match an individual student's proficiency level. This results in similar performance to problem solving alone in terms of retention and knowledge gained, but with a lot less time spent in the tutor for students with lower post-test scores. I conclude that my DDPP method offers the best overall performance, providing higher likelihood of post-test success for students, balanced completion times across low and high tracks and post-test performance groups, and reasonably low dropout rates.

CHAPTER

8

CONCLUSIONS

Conclusions

This dissertation proposed the research question of whether data-driven methods can be used to create an intelligent tutoring system that:

1. improves student performance by selecting appropriate problems and feedback,
2. reduces student dropout and time needed to complete the tutor.

By incrementally implementing mastery learning, hints & worked examples, and proficiency profiling into the Deep Thought logic tutor, I have shown how data-driven methods can be used to develop an adaptive tutoring system in an open-ended problem solving domain, and shown the effectiveness of data-driven methods in improving student performance within that tutoring system.

H1

The data-driven, adaptive and individualized instructional methods presented for mastery learning, assessment, feedback generation, worked example selection, and problem selection will improve

student learning, as measured by performance on rule scores, in an open-ended problem-solving domain.

H2

The data-driven methods presented will reduce student dropout in the Deep Thought tutor.

H3

The data-driven methods presented will reduce the time spent to complete the Deep Thought tutor.

The first study outlined in Chapter 4 tested the effect of a data-driven mastery learning system (DDML) assigning problems of appropriate difficulty in Deep Thought, version DT2-MP. Students completed 33% more of the tutor on average than previous students, and 18% more of the tutor than students using Deep Thought with hints. Students using DT2-MP also have a lower rate of tutor dropout. DT2-MP students are also significantly more likely (by almost 3 times) to complete the tutor when compared to the Control group, and almost 1.7 times as likely to complete the tutor as the Hint group. DDML students in DT2-MP also demonstrate a low rate of incorrect rule applications, and show higher proficiency of tutor concepts.

The second study outlined in Chapter 5 tested the addition of on-demand hints and worked examples to Deep Thought. In this version, version DT3-Random, students were randomly presented with worked examples and problems to solve with on-demand hints. Students using DT3-Random complete 17% more of the tutor on average than students using DT2-MP, and 35% more students using DT3-Random completed the entire tutor. In terms of total time in the tutor, students using DT3-Random spent 83 minutes less in the tutor than students using DT2-MP, saving students' time while increasing completion rates.

A data-driven proficiency profiler (DDPP) outlined in Chapter 6 was added to Deep Thought (DT5-DDPP) to augment the data-driven mastery learning system by replacing expert-based proficiency calculation with a data-driven method. Student scores were clustered and weighted based on PCA. New students were compared to these clusters, and were assigned problems based on prior track assignments for students in the matched cluster. The DDPP was compared to other data-driven methods of proficiency determination based on historical data, then implemented into Deep Thought. The DDPP resulted in the lowest dropout rate among random worked example test conditions in Deep Thought (as shown by comparing the DDML versions DT3-Random and DT5-Random with DT5-DDPP. Additionally, all of the students in the DT5-DDPP group stayed in the tutor until Level 6, where roughly 7% of those students stopped. 93% of the students using the DDPP completed the entire tutor (DT5-DDPP), while only 91% of the students using DT5-Random and

90% of the students using DT3-Random completed the entire tutor. Students using the DT5-DDPP also showed higher correct rule applications.

Chapter 7 is a descriptive analysis of the effect of combined of data-driven methods on student performance. I found that for students scoring higher (AB) on the post-test, problem-solving opportunities offer the best chance of completing the tutor quickly; however, the addition of worked examples does not add much time. The method of problem assignment (DDML or DDPP) does not have a noteworthy effect on high post-test (AB) student performance. However, for low post-test (CDF) students, I found that problem-solving opportunities alone with DDML problem assignment greatly extended the amount of time students spent in the tutor with little learning benefit. The addition of worked examples helped these students get through the tutor faster, however these students had a higher drop rate than any other students and lower examination scores. Updating our data-driven skill estimates equally for viewing or applying rules resulted in students being assigned to the high-track when they were not prepared to solve harder problems. With proficiency profiling – matching students to previously successful students and the paths they take through the tutor – we can reduce the amount of time spent in tutor, lower dropout (i.e. increase retention), and make better use of worked examples by giving them alongside problems that better match an individual student's proficiency level. The DDPP results in similar performance to problem solving alone in terms of retention and knowledge gained, but with a lot less time spent in the tutor for low post-test students. Therefore, the DDPP method offers the best overall performance, providing higher likelihood of post-test success, balanced completion times across low and high tracks and post-test performance groups, and reasonably low dropout rates.

In conclusion, this dissertation presents the following contributions

1. A novel, implemented and tested data-driven knowledge tracing system, that we have called DDML, that incorporates Bayesian knowledge tracing into a problem-solving environment that does not have a skill inventory. This research has shown that the DDML is effective in improving tutor completion and reducing student dropout. DDML combines student performance with expert-set thresholds to determine mastery.

2. An implemented and tested data-driven proficiency profiling (DDPP) system, that results in lower dropout, low time in tutor, and higher likelihood of high post-test scores than DDML. DDPP, unlike DDML, automatically sets thresholds for rule application based on clusters of prior successful students.

3. Results from three studies to determine the impact of different instructional methods (specifically worked examples and/or on-demand hints) on student learning.

4. A fully data-driven intelligent tutoring system for logic that incorporates problem selection, worked examples, on-demand hints, and the automatic setting of thresholds for mastery.

BIBLIOGRAPHY

[AK13] Aleven, V. & Koedinger, K. R. "Knowledge Component (KC) Approaches to Learner Modeling". *Design Recommendations for Intelligent Tutoring Systems*. 2013, pp. 165–182.

[Ale04] Aleven, V. et al. "Toward tutoring help seeking: Applying cognitive modeling to metacognitive skills". *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS 2004)*. Springer. 2004, pp. 227–239.

[And95] Anderson, J. R. et al. "Cognitive Tutors: Lessons Learned". *The journal of the learning sciences* **4**.2 (1995), pp. 167–207.

[Atk00] Atkinson, R. K. et al. "Learning from examples: Instructional principles from the worked examples research". *Review of educational research* **70**.2 (2000), pp. 181–214.

[BS08] Barnes, T. & Stamper, J. "Toward automatic hint generation for logic proof tutoring using historical student data". *Proceedings of the 13th International Conference on Intelligent Tutoring Systems (ITS 2008)*. 2008, pp. 373 –382.

[Bar08] Barnes, T. et al. "A pilot study on logic proof tutoring using hints generated from historical student data". *Proceedings of the 1st International Conference on Educational Data Mining (EDM 2008)*. 2008, pp. 197 –201.

[CA95] Corbett, A. T. & Anderson, J. R. "Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge". *User Modeling and User-Adapted Interaction* **4** (1995), pp. 253–278.

[Cro00] Croy, M. J. "Problem Solving, Working Backwards, and Graphic Proof Representation". *Teaching Philosophy* **23**.2 (2000), pp. 169 –187.

[Cro08] Croy, M. J. et al. "Towards an Intelligent Tutoring System for Propositional Proof Construction". *Current Issues in Computing and Philosophy*. 2008, pp. 145 –155.

[DZ12] Despotovic-Zrakic, M. et al. "Providing adaptivity in Moodle LMS courses". *Educational Technology Society* **15(1)** (2012), pp. 326–338.

[Eag15] Eagle, M. J. et al. "Exploring networks of problem-solving interactions". *Learning Analytics and Knowledge (LAK 2015)*. 2015, pp. 21–30.

[EB12] Eagle, M. & Barnes, T. "Data-Driven Method for Assessing Skill-Opportunity Recognition in Open Procedural Problem Solving Environments". *Proceedings of the 15th International Conference on Intelligent Tutoring Systems (ITS 2012)*. 2012, pp. 615–617.

[EB14a] Eagle, M. & Barnes, T. "Exploring Differences in Problem Solving with Data-Driven Approach Maps". *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*. 2014.

[EB14b] Eagle, M. & Barnes, T. "Modeling Student Dropout in Tutoring Systems". *Proceedings of the 12 International Conference on Intelligent Tutoring Systems*. Springer. 2014, pp. 676–678.

[EB14c] Eagle, M. & Barnes, T. "Survival analysis on duration data in intelligent tutors". *Proceedings of the 12 International Conference on Intelligent Tutoring Systems*. Springer. 2014, pp. 178–187.

[Eag14] Eagle, M. et al. "Interaction Network Estimation: Predicting Problem-Solving Diversity in Interactive Environments". *Proceedings of the 8th International Conference on Educational Data Mining (EDM 2015)*. 2014, pp. 342–349.

[Ehl] Ehle, A. et al. "The Sequent Calculus Trainer – Helping Students to Correctly Construct Proofs". *Proceedings of the Fourth International Conference on Tools for Teaching Logic, TTL 2015*, pp. 35–44.

[Elm12] Elmadani, M. et al. "Data-Driven Misconception Discovery in Constraint-based Intelligent Tutoring Systems". *Workshop Proceedings of the 20th International Conference on Computers in Education (ICCE 2012)*. 2012.

[Fan14] Fancsali, S. E. "Causal Discovery with Models: Behavior, Affect, and Learning in Cognitive Tutor Algebra". *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*. 2014, pp. 28–35.

[Glu14] Gluz, J. C. et al. "A Student Model for Teaching Natural Deduction Based on a Prover That Mimics Student Reasoning". *Proceedings of the 12th International Conference on Intelligent Tutoring Systems (ITS 2014)*. 2014, pp. 482–489.

[Gon10] Gong, Y. et al. "Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting". *Proceedings of the 14th International Conference on Intelligent Tutoring Systems (ITS 2010)*. 2010, pp. 35–44.

[GBM13] González-Brenes, J. P. & Mostow, J. "What and when do students learn? Fully data-driven joint estimation of cognitive and student models". *Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013)*. 2013, pp. 236–239.

[HR09] Hilbert, T. S. & Renkl, A. "Learning how to use a computer-based concept-mapping tool: Self-explaining examples helps". *Computers in Human Behavior* **25**.2 (2009), pp. 267–274.

[Hua14] Huang, Y. et al. "General Features in Knowledge Tracing: Applications to Multiple Subskills, Temporal Item Response Theory, and Expert Knowledge". *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*. 2014, pp. 84–91.

[Kiz13] Kizilcec, R. F. et al. "Deconstructing disengagement: analyzing learner subpopulations in massive open online courses". *Proceedings of the 3rd international conference on learning analytics and knowledge*. 2013, pp. 170–179.

[KM11] Klasnja-Milicevic, A. et al. "E-Learning personalization based on hybrid recommendation strategy and learning style identification." *Computers Education* **56(3)** (2011), pp. 885–899.

[Koe13] Koedinger, K. R. "New Potentials for Data-Driven Intelligent Tutoring System Development and Optimization". *AI Magazine* **34**.3 (2013), pp. 27–41.

[Koe97] Koedinger, K. R. et al. "Intelligent tutoring goes to school in the big city". *International Journal of Artificial Intelligence in Education (IJAIED)* **8** (1997), pp. 30–43.

[Koe04] Koedinger, K. R. et al. "Opening the Door to Non-programmers: Authoring Intelligent Tutor Behavior by Demonstration". *Proceedings of the 7th International Conference on Intelligent Tutoring Systems, ITS2004*. Vol. 7. Springer. 2004, p. 162.

[Koe11] Koedinger, K. R. et al. "Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing". *Proceedings of the 4th International Conference on Educational Data Mining (EDM 2011)*. 2011, pp. 91–100.

[Koe14] Koedinger, K. R. et al. "Data-driven Learner Modeling to Understand and Improve Online Learning: MOOCs and technology to advance learning and learning research (Ubiquity symposium)". *Ubiquity 2014*. 2014.

[LB12] Lee, J. & Brunskill, E. "The impact on individualizing student models on necessary practice opportunities". *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)*. 2012, pp. 118–125.

[Lod] Lodder, J. et al. "A pilot study of the use of LogEx, lessons learned". *Proceedings of the Fourth International Conference on Tools for Teaching Logic, TTL 2015*, pp. 94–100.

[Luk02] Lukins, S. et al. "A tutorial program for propositional logic with human/computer interactive learning". *ACM SIGCSE Bulletin*. Vol. 34. 1. ACM. 2002, pp. 381–385.

[Ma14] Ma, W. et al. "Intelligent Tutoring Systems and Learning Outcomes: A Meta-Analysis." (2014).

[MI11] McLaren, B. M. & Isotani, S. "When Is It Best to Learn with All Worked Examples?" *Artificial Intelligence in Education*. 2011, pp. 222–229.

[McL08] McLaren, B. M. et al. "When and How Often Should Worked Examples Be Given to Students? New Results and a Summary of the Current State of Research". *Proc. 30th*

Conf. Cognitive Science Society. Ed. by Love, B. C. et al. Cognitive Science Society, 2008, pp. 2176–2181.

[McL14] McLaren, B. M. et al. "Exploring the Assistance Dilemma: Comparing Instructional Support in Examples and Problems". *Intelligent Tutoring Systems*. 2014, pp. 354–361.

[MY05] Merceron, A. & Yacef, K. "Educational Data Mining: A Case Study". *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005)*. University of Sydney, Australia. IOS Press, 2005, pp. 467 –474.

[MJ90] Meyers Jr., J. P. "The central role of mathematical logic in computer science". *Proceedings of the twenty-first SIGCSE technical symposium on Computer science education, SIGCSE 1990*. 1990, pp. 22–26.

[Mit13] Mitrovic, A. et al. "The effect of positive feedback in a constraint-based intelligent tutoring system". *Computers and Education* **60**.1 (2013), pp. 264–272.

[MB11] Mostafavi, B. & Barnes, T. "Automatic Generation of Proof Problems in Deductive Logic". *Proceedings of the 4th International Conference on Educational Data Mining (EDM 2011)*. 2011, pp. 289 –294.

[Mur99] Murray, T. "Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art". *International Journal of Artificial Intelligence in Education* **10** (1999), pp. 98–129.

[MA02] Murray, T. & Arroyo, I. "Toward Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems". *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS 2002)*. 2002, pp. 289 –294.

[NM12] Najar, A. & Mitrovic, A. "Should We Use Examples in Intelligent Tutors?" *Proceedings of the 20th International Conference on Computers in Education*. 2012, pp. 5–7.

[Pag03] Page, R. L. *Software is discrete mathematics*. 2003.

[PH11] Pardos, Z. A. & Heffernan, N. T. "KT-IDEM : Introducing Item Difficulty to the Knowledge Tracing Model". *Proceedings of the 19th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2011)*. 2011, pp. 243–254.

[Par11] Pardos, Z. A. et al. "The Sum is Greater than the Parts: Ensembling Models of Student Knowledge in Educational Software". *ACM SIGKDD Explorations Newsletter* **13**.2 (2011), pp. 37–44.

[Per12] Perez, E. V. et al. "Automatic classification of question difficulty level: Teachers' estimation vs. students' perception". *Proceedings of the IEEE Frontiers in Education Conference*. 2012, pp. 1–5.

[RH10] Razzaq, L. & Heffernan, N. T. "Hints: is it better to give or wait to be asked?" *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS2010) Part 1*. Springer. 2010, pp. 349–358.

[Rit07] Ritter, S. "Cognitive Tutor: Applied research in mathematics education". *Psychonomic bulletin review* **14**.2 (2007), pp. 249–255.

[Shi08] Shih, B. et al. "A Response Time Model for Bottom-Out Hints as Worked Examples". *Proceedings of the 1st International Conference on Educational Data Mining, EDM2008*. 2008, pp. 117–126.

[Sie07] Sieg, W. "The AProS project: Strategic thinking & computational logic". *Logic Journal of IGPL* **15**.4 (2007), pp. 359–368.

[Sno14] Snow, E. et al. "Entropy: A Stealth Measure of Agency in Learning Environments". *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*. 2014, pp. 185–192.

[Sta11] Stamper, J. et al. "Experimental Evaluation of Automatic Hint Generation for a Logic Tutor". *Proceedings of the 15th International Conference on Artificial Intelligence in Education (AIED 2011)*. 2011, pp. 345–352.

[SHC14] Steenbergen-Hu, S. & Cooper, H. "A meta-analysis of the effectiveness of intelligent tutoring systems on college students' academic learning." *Journal of Educational Psychology* **106**.2 (2014), p. 331.

[SC85] Sweller, J. & Cooper, G. A. "The use of worked examples as a substitute for problem solving in learning algebra". *Cognition and Instruction* **2**.1 (1985), pp. 59–89.

[Van06] VanLehn, K. "The Behavior of Tutoring Systems". *International Journal of Artificial Intelligence in Education* **16**.2 (2006), pp. 227–265.

[Van11a] VanLehn, K. "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems". *Educational Psychologist* **46**.4 (2011), pp. 197–221. eprint: `http://dx.doi.org/10.1080/00461520.2011.611369`.

[Van05] Vanlehn, K. et al. "The Andes physics tutoring system: Lessons learned". *International Journal of Artificial Intelligence in Education* **15**.3 (2005), pp. 147–204.

[Van11b] VanLehn, K. et al. "The Level Up Procedure: How to Measure Learning Gains without Pre- and Post-Testing". *Proceedings of the 19th International Conference on Computers in Education*. 2011.

[WR06] Watering, G. van de & Rijt, J. van der. "Teachers and students perceptions of assessments: A review and a study into the ability and accuracy of estimating the difficulty levels of assessment items". *Educational Research Review* **1(2)** (2006), pp. 133–147.

[Xu12] Xu, Y. et al. "Comparison of methods to trace multiple subskills: Is LR-DBN best?" *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)*. 2012, pp. 41–48.

APPENDIX

APPENDIX

A

DEEP THOUGHT

A.1 Knowledge Component Definitions

Table A.1 Logical Implication (Inference) Rules

Rule	**Abbreviation**	**Definition**
Modus Ponens	MP	$(p \rightarrow q) \wedge p \Rightarrow q$
Disjunctive Syllogism	DS	$(p \vee q) \wedge \neg p \Rightarrow q$
Simplification	Simp	$(p \wedge q) \Rightarrow p$ **OR** $(p \wedge q) \Rightarrow q$
Hypothetical Syllogism	HS	$(p \rightarrow q) \wedge (q \rightarrow r) \Rightarrow (p \rightarrow r)$
Modus Tollens	MT	$(p \rightarrow q) \wedge \neg q \Rightarrow \neg p$
Addition	Add	$p \Rightarrow (p \vee q)$
Conjunction	Conj	$p \wedge q \Rightarrow (p \wedge q)$
Constructive Dilemma	CD	$(p \rightarrow q) \wedge (r \rightarrow s) \Rightarrow (p \vee r) \rightarrow (q \vee s)$

Table A.2 Logical Equivalence (Replacement) Rules

Rule	Abbreviation	Definition
Double Negation	DN	$p \Leftrightarrow \neg\neg p$
Implication	Impl	$(p \rightarrow q) \Leftrightarrow (\neg p \vee q)$ **OR** $(p \vee q) \Leftrightarrow (\neg p \rightarrow q)$
Equivalence	Equiv	$(p \leftrightarrow q) \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
DeMorgan's	DeM	$\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$ **OR** $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$
Contrapositive	CP	$(p \rightarrow q) \Leftrightarrow (\neg q \rightarrow \neg p)$

Table A.3 Logical Binary Operation Rules

Rule	Abbreviation	Definition
Commutative	Com	$(p \wedge q) \Leftrightarrow (q \wedge p)$ **OR** $(p \vee q) \Leftrightarrow (q \vee p)$
Associative	Assoc	$(p \wedge (q \wedge r)) \Leftrightarrow ((p \wedge q) \wedge r)$ **OR** $(p \vee (q \vee r)) \Leftrightarrow ((p \vee q) \vee r)$
Distributive	Dist	$(p \wedge (q \vee r)) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$ **OR** $(p \vee (q \wedge r)) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
Absorption	Abs	$(p \rightarrow q) \Leftrightarrow (p \rightarrow (p \wedge q))$
Exportation	Exp	$((p \wedge q) \rightarrow r) \Leftrightarrow (p \rightarrow (q \rightarrow r))$
Tautology	Taut	$p \Leftrightarrow (p \wedge p)$ **OR** $p \Leftrightarrow (p \vee p)$
Contradiction	$\varnothing$	$(p \wedge \neg p) \Leftrightarrow \varnothing$

A.2 Problem List

In the following table, the Problem field is in the format L.D.P.A, where:

- L refers to the domain Level of the problem (1-6)
- D refers to the proficiency path Difficulty of the problem (0: Common, -1: Low Proficiency Path, 1: High Proficiency Path)
- P refers to the list position of the Problem in the proficiency path (1-4)
- A refers to the Alternate status of the problem (0: Original, 1: Alternate)

Table A.4 The problem list for Deep Thought. Premises are comma-separated.

Problem	Premises	Conclusion	Expected Rule Use
1.0.1.0	$A \rightarrow (B \wedge C), A \vee D, \neg D \wedge E$	B	Simp, DS, MP
1.0.2.0	$B \wedge (S \vee L), \neg U \rightarrow \neg L, \neg U$	S	Simp, MP, DS
1.0.3.0	$(W \vee \neg R) \wedge N, (G \vee Z) \rightarrow \neg W, Z$	$\neg R$	Simp, Add, MP, DS
1.0.4.0	$P \vee Q, \neg Q, P \rightarrow (R \wedge S)$	$S \vee T$	DS, MP, Simp, Add
2.-1.1.0	$(F \vee G) \rightarrow H, I \vee F, \neg I \wedge J$	H	Simp, DS, Add, MP
2.-1.1.1	$F \rightarrow (G \wedge \neg H), I \wedge F, H \vee J$	$J \vee K$	Simp, MP, DS, Add
2.-1.2.0	$(\neg K \vee L) \rightarrow (M \wedge N), K \rightarrow O, \neg O$	N	MT, Add, MP, Simp
2.-1.2.1	$(\neg O \vee L) \rightarrow (M \wedge \neg N), \neg O, K \rightarrow N$	$\neg K$	Add, MP, Simp, MT
2.-1.3.0	$L \wedge K, (K \wedge O) \rightarrow (Q \rightarrow P), O, P \rightarrow R$	$Q \rightarrow R$	Simp, Conj, MP, HS
2.-1.3.1	$Q \rightarrow K, R \wedge O, K \rightarrow P, Q$	$P \wedge R$	HS, Simp, MP, Conj
2.-1.4.0	$C \rightarrow F, G \vee \neg B, A \wedge (F \rightarrow B), \neg G$	$\neg C$	Simp, DS, HS, MT
2.-1.4.1	$B \vee (A \rightarrow C), B \rightarrow G, C \rightarrow D, \neg G \wedge H$	$A \rightarrow D$	Simp, MT, DS, HS
2.1.1.0	$(\neg T \wedge S) \rightarrow \neg R, \neg T, (\neg Q \vee P) \rightarrow S, Q \rightarrow T$	$\neg R \vee N$	MT, Add, MP, Conj
2.1.2.0	$B \rightarrow (A \rightarrow J), D \wedge \neg(A \rightarrow \neg C), B \vee (A \rightarrow \neg C), J \rightarrow \neg C$	$A \rightarrow \neg C$	Simp, DS, MP, HS
2.1.3.0	$A \rightarrow C, B, C \rightarrow E, D \wedge \neg E$	$\neg A \wedge B$	HS, Simp, MT, Conj
3.-1.1.0	$K \vee J, \neg Z \wedge \neg R, J \rightarrow Z$	$(J \vee L) \vee K$	Simp, MT, DS, Add
3.-1.1.1	$C \rightarrow B, A \wedge \neg B, (\neg C \vee J) \rightarrow \neg(B \wedge A)$	$\neg(B \wedge A)$	Simp, MT, Add, MP
3.-1.2.0	$C \wedge \neg F, \neg C \rightarrow F, \neg F \rightarrow \neg(\neg K \vee \neg Z), D \rightarrow (\neg K \vee \neg Z)$	$\neg C \vee \neg D$	Simp, MP, MT, Add
3.-1.2.1	$Z \rightarrow K, C, \neg Z \vee \neg K, (C \vee D) \rightarrow (\neg F \wedge \neg K)$	$\neg Z$	Add, MP, Simp, MT
3.-1.3.0	$(S \rightarrow Y) \vee (I \wedge Q), (I \wedge Q) \rightarrow D, \neg D, (S \rightarrow Y) \rightarrow D$	Y	MT, DS, MP, Add
3.-1.3.1	$(S \rightarrow D) \vee I, (\neg S \vee Q) \rightarrow Y, \neg D, \neg D \rightarrow \neg I$	Y	MP, DS, MT, Add
3.-1.4.0	$(P \wedge S) \vee T, S \vee Q, T \rightarrow R, \neg R$	$S \wedge \neg R$	MT, DS, Simp, Conj
3.-1.4.1	$B, \neg(B \wedge \neg C) \vee \neg T, \neg C, \neg(P \wedge Q) \rightarrow T$	P	Conj, DS, MT, Simp
3.1.1.0	$\neg B \rightarrow (J \rightarrow I), (\neg J \vee \neg I) \rightarrow (B \rightarrow W), (\neg I \wedge \neg J) \wedge \neg W$	$J \rightarrow I$	Simp, Add, MP, MT
3.1.2.0	$K \rightarrow R, \neg S \wedge (\neg R \vee P), O \rightarrow S, (\neg S \vee T) \rightarrow \neg P$	$\neg K \wedge \neg O$	Simp, Add, MT, MP, DS, Conj
3.1.3.0	$\neg U, (B \rightarrow U) \wedge (L \vee C), \neg L, (C \vee D) \rightarrow H$	$\neg B \wedge H$	Simp, MT, DS, Add, MP, Conj
4.-1.1.0	$\neg G \wedge K, L, J \rightarrow G, (K \wedge L) \rightarrow (N \rightarrow J)$	$\neg N \vee P$	Simp, Conj, MT, MP, Add
4.-1.1.1	$K \rightarrow G, K \wedge \neg L, J \rightarrow L, (G \wedge \neg J) \rightarrow N$	$N \vee O$	Simp, MP, MT, Conj, Add
4.-1.2.0	$(C \rightarrow D) \vee B, B \rightarrow G, (D \rightarrow F) \wedge \neg G$	$C \rightarrow F$	Simp, MT, DS, HS
4.-1.2.1	$C \rightarrow D, \neg B \wedge \neg F, (D \rightarrow F) \vee B$	$\neg C$	Simp, DS, HS, MT
4.-1.3.0	$(\neg F \vee G) \rightarrow (W \vee \neg G), (\neg Y \rightarrow \neg F) \wedge (E \rightarrow G), \neg Y \vee E, \neg W$	$\neg F \wedge \neg G$	CD, MP, DS, Conj
4.-1.3.1	$(A \rightarrow B) \wedge (\neg D \rightarrow F), A \vee \neg D, \neg A \rightarrow (D \vee G), \neg A$	$(B \vee F) \wedge G$	CD, MP, DS, Conj
4.-1.4.0	$(A \rightarrow C) \wedge (\neg D \rightarrow F), A \vee \neg D, (C \rightarrow E) \wedge (F \rightarrow G), \neg E$	G	CD, DS
4.-1.4.1	$(B \rightarrow D) \wedge (H \rightarrow J), T, (D \rightarrow P) \wedge (J \rightarrow Q), \neg T \vee (B \vee H)$	$(P \vee Q)$	DS, CD
4.1.1.0	$(R \wedge N) \rightarrow O, \neg O \wedge (\neg W \rightarrow A), (U \rightarrow K) \vee (R \wedge N), K \rightarrow \neg W$	$U \rightarrow A$	Simp, MT, DS, HS
4.1.2.0	$Z \rightarrow (\neg Y \rightarrow X), Z \wedge \neg W, W \vee (T \rightarrow S), \neg Y \vee T$	$X \vee S$	Simp, MP, DS, Conj, CD
4.1.3.0	$\neg G \rightarrow Q, (Q \rightarrow A) \wedge P, H \rightarrow \neg B, P \rightarrow (\neg G \vee H)$	$A \vee \neg B$	Simp, HS, MP, Conj, CD
5.-1.1.0	$\neg R \vee Z, G \rightarrow \neg Z$	$R \rightarrow \neg G$	Impl, CP, DN, HS
5.-1.1.1	$\neg K \vee J, L \rightarrow \neg J$	$K \rightarrow \neg L$	Impl, CP, DN, HS
5.-1.2.0	$\neg(G \wedge A), B \rightarrow A$	$G \rightarrow \neg B$	DeM, CP, Impl, HS
5.-1.2.1	$\neg(\neg G \wedge B), G \rightarrow D$	$\neg B \vee D$	DeM, Impl, DN, CP, HS
5.-1.3.0	$Y \rightarrow F, \neg\neg\neg F$	$\neg(Y \vee F)$	DN, MT, Conj, DeM
5.-1.3.1	$\neg(Y \vee F), F \vee \neg\neg\neg L$	$\neg Y \wedge \neg L$	DeM, DN, Simp, DS, Conj
5.-1.4.0	$C \rightarrow D, \neg(G \vee \neg C)$	$\neg D \rightarrow F$	DeM, Impl, Simp, DN, MP, DS, Add
5.-1.4.1	$B \vee F, (\neg A \rightarrow B)$	$F \vee \neg A$	Impl, DeM, Simp, MP, DS, Add, DN
5.1.1.0	$(A \rightarrow \neg B) \vee C, \neg C, D \vee B$	$\neg D \rightarrow \neg A$	DS, DN, CP, Impl, HS
5.1.2.0	$K \rightarrow M, Z \rightarrow R, \neg(K \rightarrow R)$	$M \wedge \neg Z$	Impl, DeM, DN, Simp, MP, MT
5.1.3.0	$J \rightarrow (K \wedge L), L \rightarrow M, \neg(K \wedge M)$	$\neg J$	CP, DeM, Impl, HS, MT
6.-1.1.0	$\neg(T \wedge L), \neg T \rightarrow \neg N, \neg(E \vee T)$	$\neg N$	DeM, Simp, MP
6.-1.1.1	$\neg(E \wedge T), \neg E \vee \neg L, T \wedge \neg L$	$\neg E$	DeM, Simp, DN, DS
6.-1.2.0	$J \rightarrow H, \neg W \rightarrow (\neg H \wedge \neg F), J$	W	MP, Add, DN, DeM, MT
6.-1.2.1	$\neg(K \wedge \neg N) \rightarrow W, \neg L, K \rightarrow L$	W	MP, Add, DN, DeM, MT
6.-1.3.0	$Y \leftrightarrow P, \neg Y \rightarrow \neg C, \neg P \leftrightarrow \neg C$	$Y \rightarrow C$	Equiv, Simp, CP, HS
6.-1.3.1	$C \leftrightarrow \neg J, C \vee \neg N, N \rightarrow C$	$N \rightarrow \neg J$	Equiv, CP, Simp, HS
6.-1.4.0	$A \rightarrow \neg(B \rightarrow C), \neg(D \wedge \neg A), \neg B$	$\neg D$	DeM, DN, Impl, HS, Add, MT
6.-1.4.1	$P \rightarrow (\neg\neg Q \wedge \neg R), \neg Q, \neg S \vee P$	$\neg S$	DeM, Impl, HS, Add, MT, DN
6.1.1.0	$S \vee B, B \rightarrow D, S \rightarrow G$	$D \vee G$	CP, DN, Impl, JS, Conj, CD
6.1.2.0	$B \leftrightarrow \neg J, \neg N \vee J, B \vee \neg N$	$B \rightarrow \neg N$	Equiv, Simp, DN, Impl, HS
6.1.3.0	$\neg Z \leftrightarrow P, \neg(P \rightarrow V), V \vee \neg Q$	$\neg(Z \vee Q)$	Equiv, Impl, DeM, Simp, DN, MP, DS, Conj

A.3 Operation Screenshots

Operation Screenshots for Deep Thought problem 1.0.1.0.

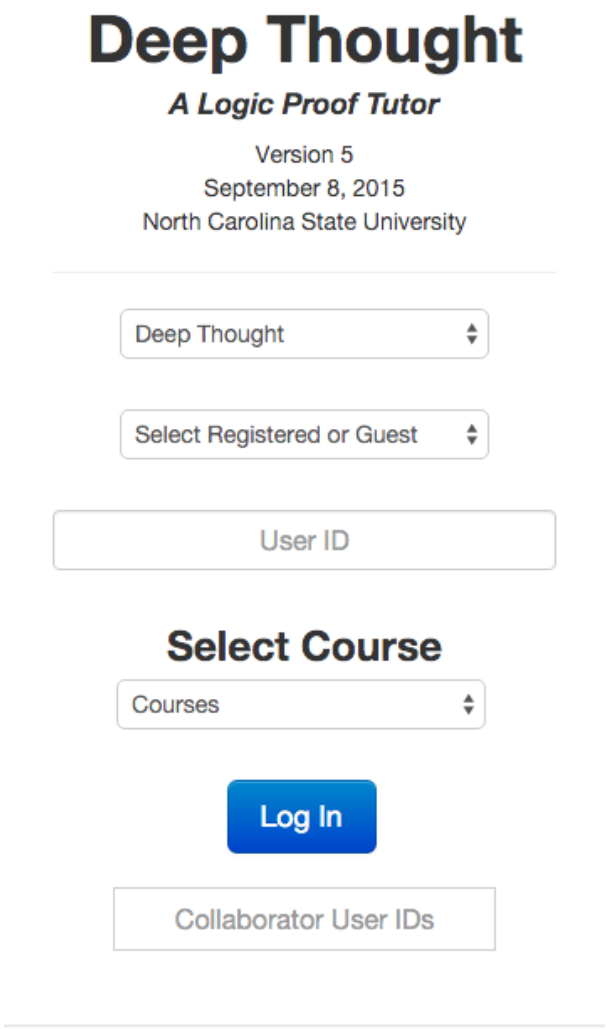

Figure A.1 The Deep Thought login screen.

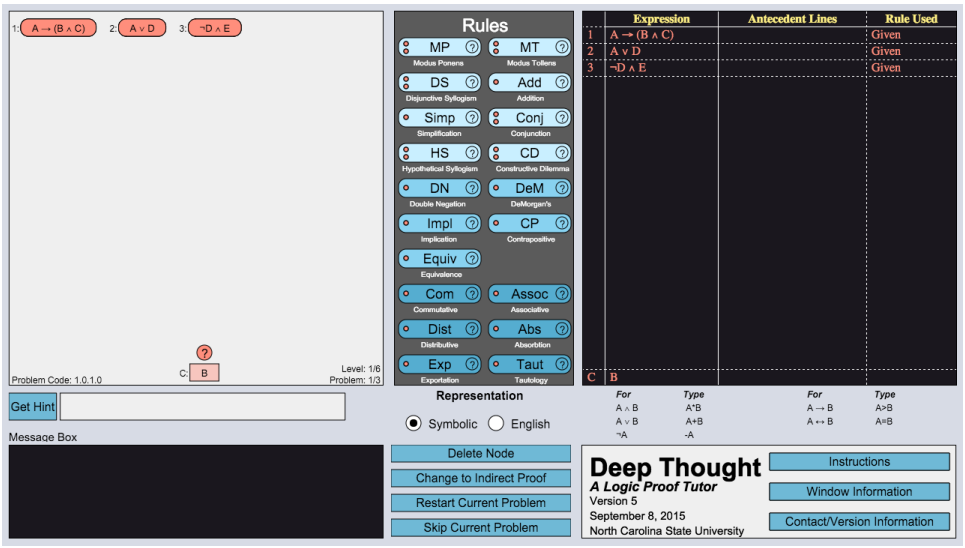

Figure A.2 The Deep Thought main problem window.

Figure A.3 The Deep Thought main problem window (English representation).

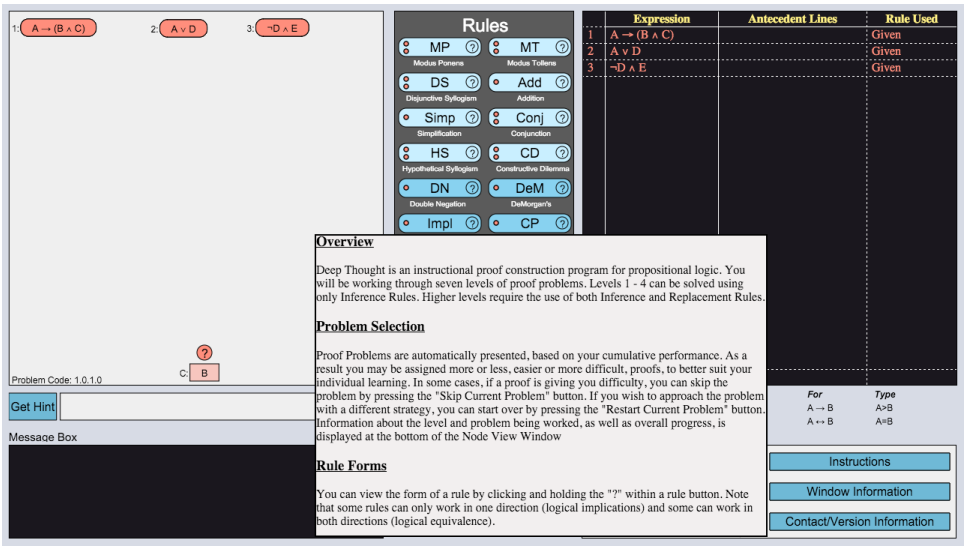

Figure A.4 The Deep Thought instructions pop-up.

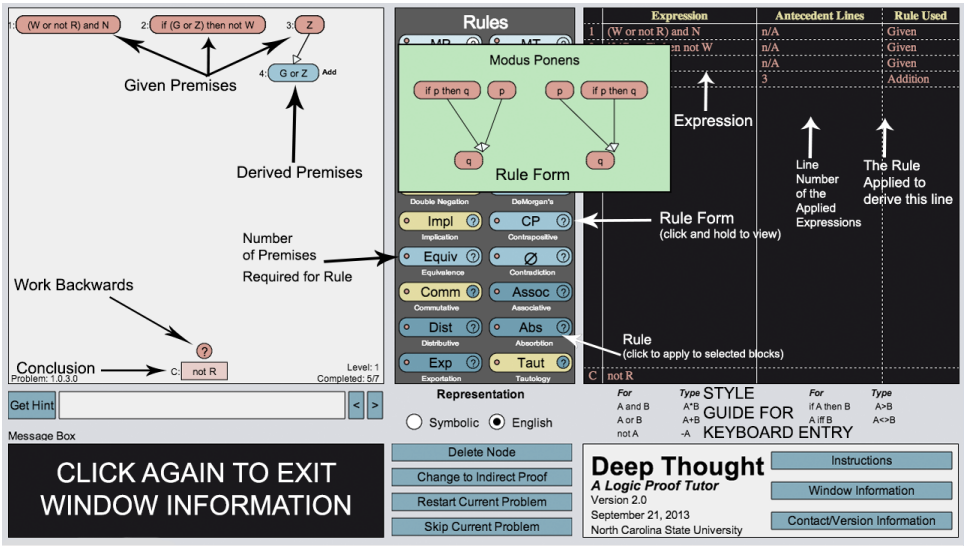

Figure A.5 The Deep Thought window information pop-up.

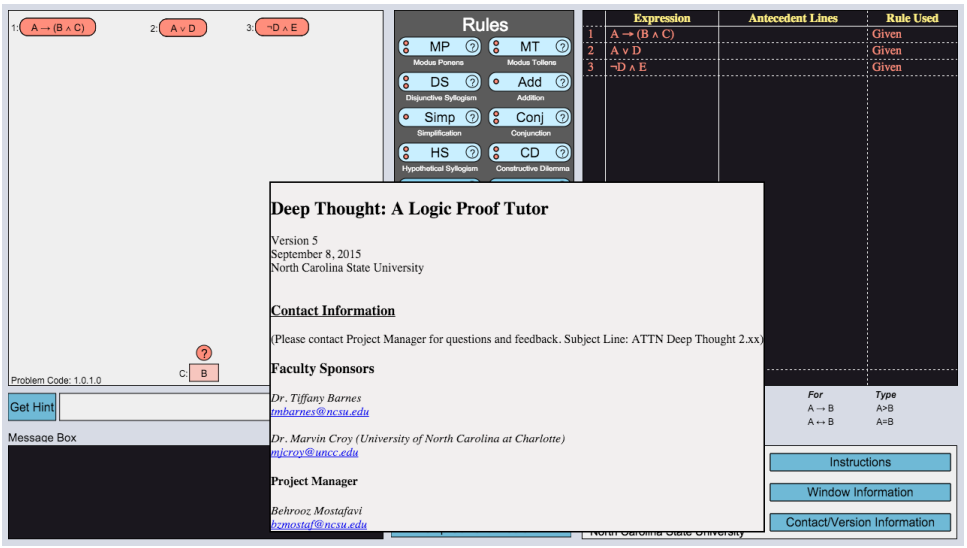

Figure A.6 The Deep Thought contact information pop-up.

Figure A.7 The Deep Thought main problem window (indirect proof mode).

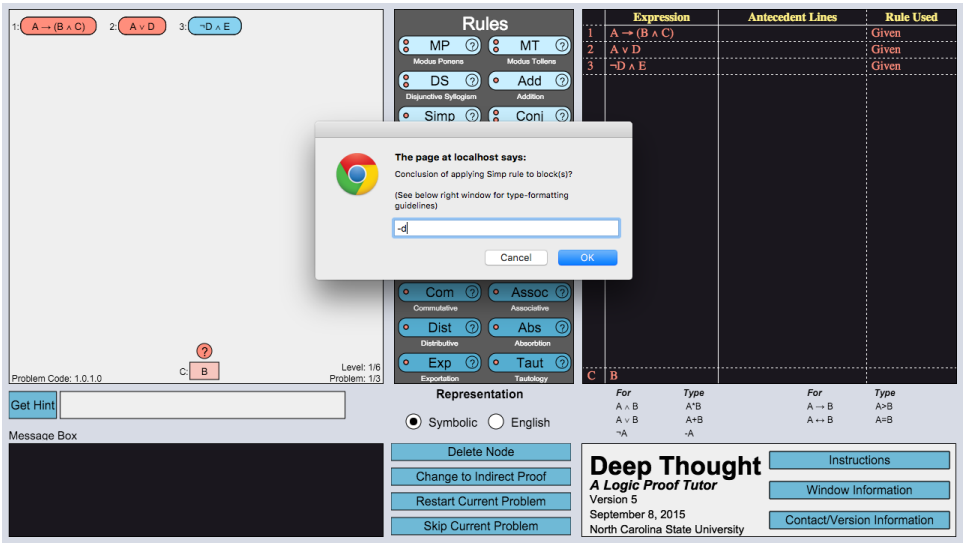

Figure A.8 The Deep Thought text entry window for rule application.

Figure A.9 A Deep Thought rule reference pop-up (for Simplification).

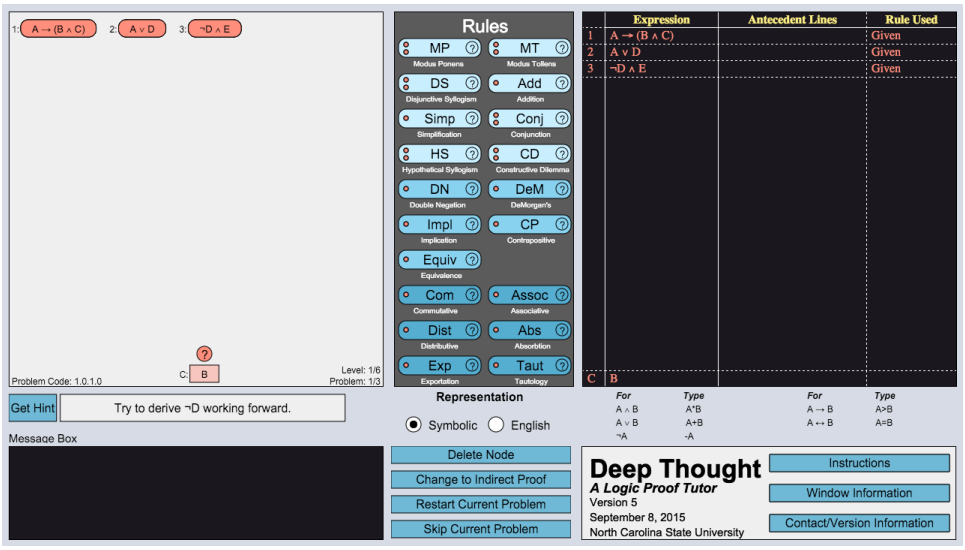

Figure A.10 The first Hint Factory hint for the current screen state.

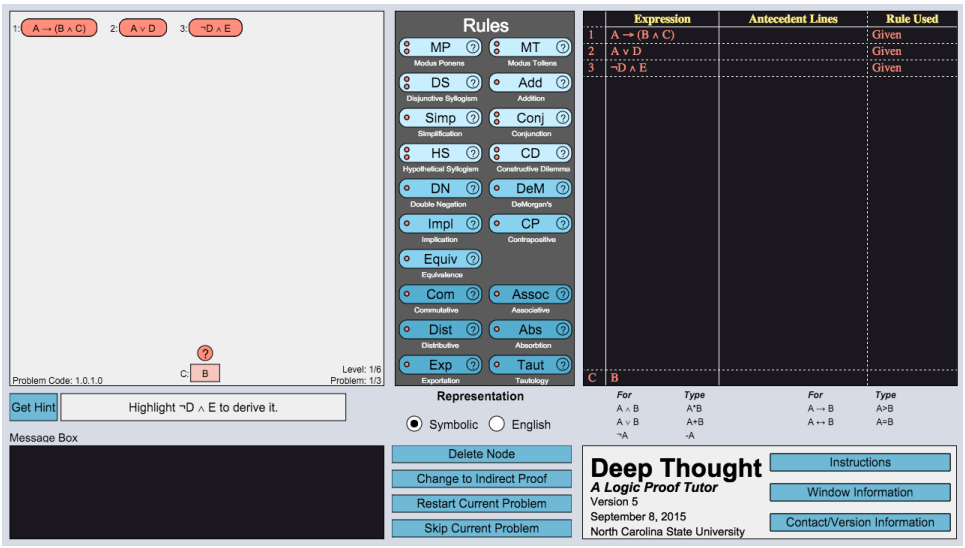

Figure A.11 The second Hint Factory hint for the current screen state.

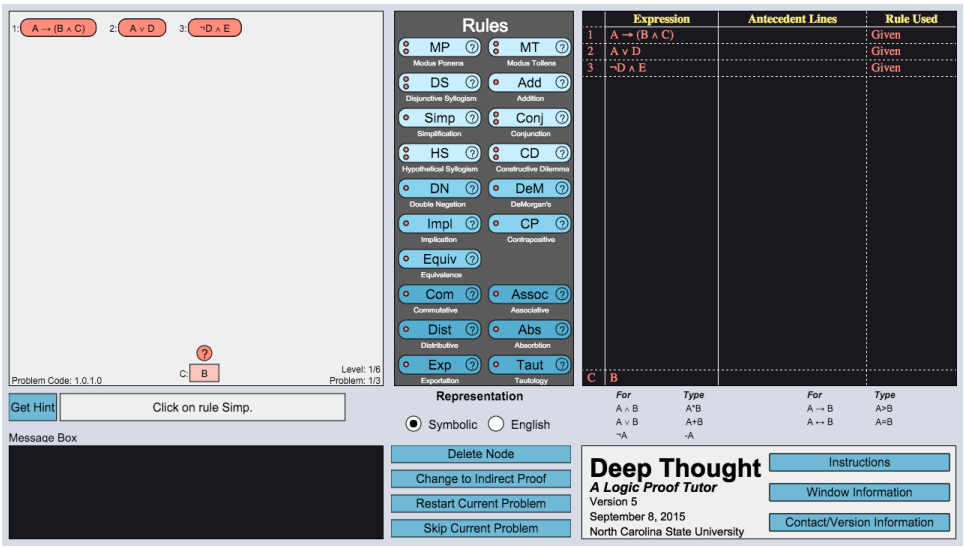

Figure A.12 The third Hint Factory hint for the current screen state.

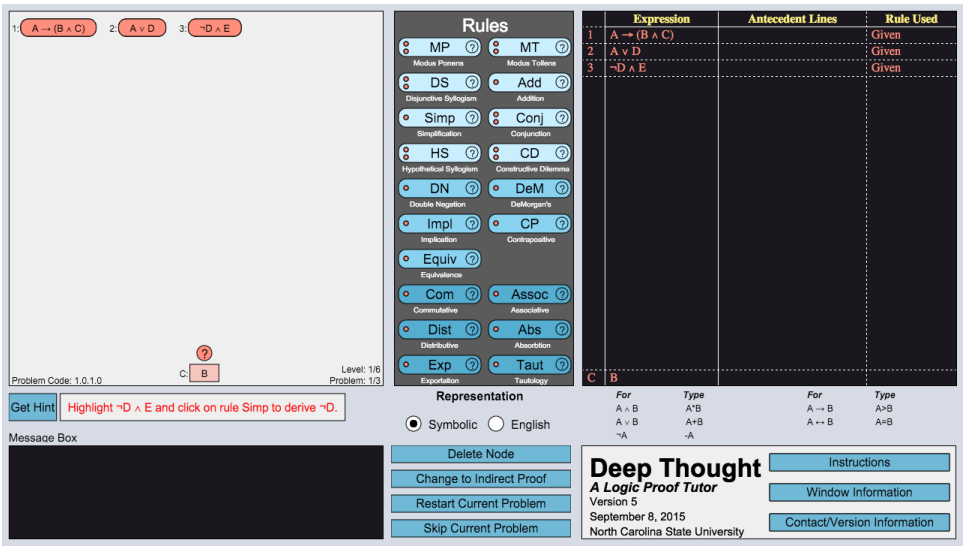

Figure A.13 The fourth and final (bottom out) Hint Factory hint for the current screen state.

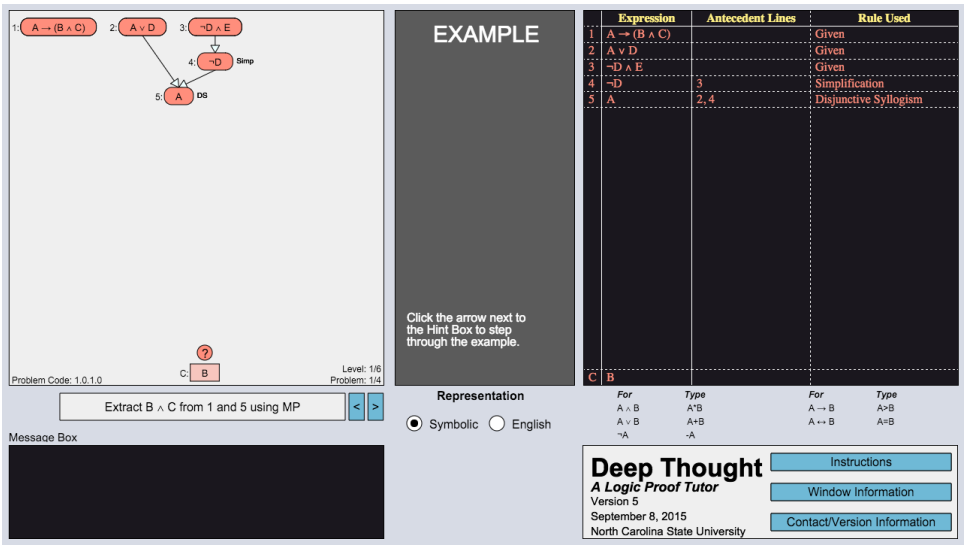

Figure A.14 The worked example (in process) for problem 1.0.1.0.

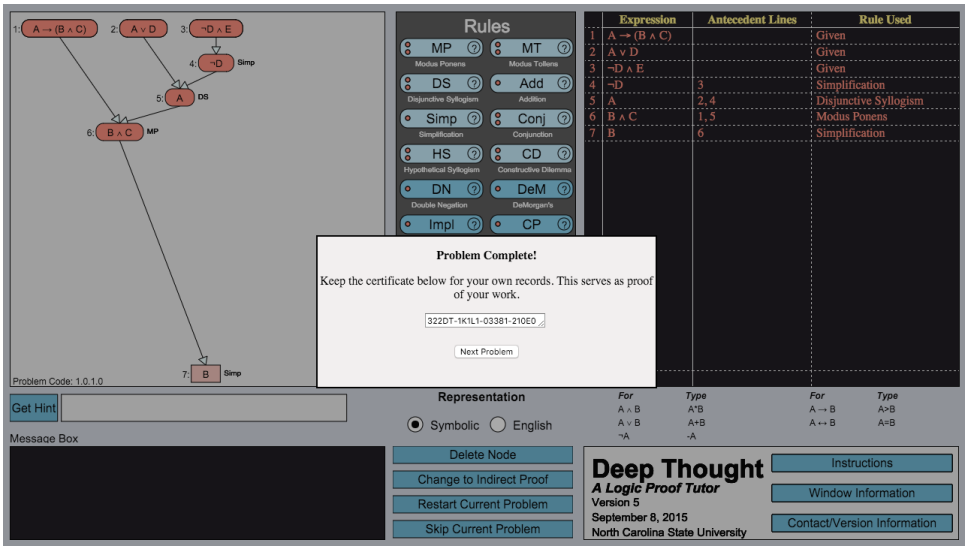

Figure A.15 The Deep Thought problem complete screen.