Rule-based Cognitive Modeling and Model Tracing for Symbolization in a Math Story Problem Tutor

https://doi.org/10.3991/ijet.v12i04.6592

Nabila Khodeir
Electronics Research Institute, Cairo, Egypt
nkhodeir@eri.sci.eg

Hanan Elazhary
King Abdulaziz University, Jeddah, Saudi Arabia
Electronics Research Institute, Cairo, Egypt
helazhary@kau.edu.sa; hananelazhary@eri.sci.eg

Nayer Wanas
Electronics Research Institute, Cairo, Egypt
nwanas@eri.sci.eg

Abstract—Intelligent Tutoring Systems (ITSs) are intended to help in tutoring students in specific domains, typically by improving their problem solving skills. An important aspect of such ITSs is the ability to solve problems in the same manner that the student would, in addition to interpreting student actions and providing relevant feedback and help in case of any errors. Cognitive models, that mimic the way procedural knowledge is represented in human minds, are excellent means toward achieving this goal. This paper discusses cognitive modeling in the MAth Story problem Tutor (MAST). MAST is a Web-based ITS that can generate probability story problems of different contexts, types and difficulty levels. A major contribution of the paper is the ability of MAST to symbolize the probability word problems and solve them in the same manner that the student would. The paper discusses the model tracing approach of MAST to interpret the student actions in symbolizing the word problems and estimating the required probabilities to provide relevant feedback and help in case of any errors. Evaluation results have shown the ability of MAST to tutor the students and considerably improve their probability story problem solving skills.

Keywords—Cognitive Modeling, Intelligent Tutoring Systems, Model Tracing, Rule-based Systems

1 Introduction

Intelligent tutoring systems (ITSs) aim to assist the students to acquire and enhance their knowledge and problem solving skills in a specific domain [1]. Most ITSs target to analyze the student answers to generated problems to provide instant and adaptive tutoring and feedback, typically without involvement of any human tutor [2, 3]. Therefore, such systems have to employ knowledge representation of their selected domains in addition to inference mechanisms to be applied on the represented knowledge to solve the domain problems [4]. A common approach to achieving those requirements is via developing an expert system module that is able to generate the problem answers and provide the base for interpreting student actions [5]. An excellent means of achieving this goal is by adopting a cognitive approach in implementing the expert system module to mimic the manner by which procedural knowledge is represented in the human mind. This would allow ITSs to respond to problem-solving situations in a manner similar to that of the student [6]. Another important aspect of such ITSs is integrating cognitive modeling with model tracing to be able to interpret the student actions in terms of the knowledge of the cognitive model to diagnose student errors and provide relevant feedback and help [7].

This paper presents MAST, a Web-based ITS of probability story problems. MAST has the ability to generate problems of different contexts, types and difficulty levels for self-paced learning. Additionally, it utilizes natural language generation methods to generate word problems with various syntaxes, wordings and operators. MAST also employs a rule-based cognitive model to independently solve the generated word problems and integrates cognitive modeling with model tracing to interpret the student actions in terms of the cognitive model knowledge to provide relevant help. A major contribution of the paper is in the symbolization of the probability word problems to convert them to the symbolic form, while tracing the students errors. Details of all the different modules of MAST are outside the scope of this paper. The paper is mainly concerned with discussing and evaluating cognitive modeling and model tracing in MAST. Though MAST considers numerous contexts, the paper emphasizes the context of rolling a die and tossing a coin - as an example - due to space limitations and since it is generally the most common amongst the available contexts.

The rest of the paper is organized as follows. Section 2 discusses expert systems in ITSs. Section 3 provides an overview of MAST and how it specifies its word problems types and difficulty level categories. The details of cognitive modeling and model tracing in MAST are provided in Sections 4 and 5 respectively. The results of the empirical evaluation of MAST are discussed in Section 6. Finally, Section 7 presents the conclusion of the paper and directions for future research.

2 Expert Systems Supporting Problem Solving in ITSs

As previously noted, ITSs that aim to guide the students in the problem solving process typically employ domain specific expert modules that are able to solve domain problems. Such an expert system module consists of explicit representation of domain knowledge and an inference mechanism that exploits this knowledge in the problem solving process. Knowledge in artificial intelligence is broadly classified into declarative knowledge or facts and procedural knowledge or problem solving steps. Different knowledge representations can be considered in implementing expert system modules. Nevertheless, the nature of the domain and its knowledge has a direct effect on the selection of the appropriate knowledge representation scheme. For example, semantic networks, conceptual graphs and concept maps are suitable for representing declarative knowledge [8, 9, 10]. On the other hand, If-Then production rules are commonly used for representing procedural knowledge of expert systems modules within the context of ITSs [11, 12, 13].

An expert system module developed in a way that facilitates communication with the student, is called a cognitive model. The goal of cognitive modeling is to develop a simulation of human problem solving in terms of knowledge components that mimic humans [14]. A rule-based cognitive model is typically composed of 3 main modules: (a) a knowledge base, (b) a working memory and (c) an inference engine. The knowledge base includes a set of If-Then production rules. An If-Then rule is composed of one or more conditions and one or more actions that are performed when the conditions are satisfied. The working memory stores a number of facts that are matched by the inference engine against the rules in the knowledge base to decide which rule(s) to execute or fire. In the context of problem solving, the production rules represent solution steps according to problem facts in the working memory in a specific domain.

In the mathematics domain, several ITSs incorporated expert system modules. Nevertheless, most of those systems such as PAT [15], PAT2Math [16] and Aplusix [17] were developed for the Algebra domain. The 3rd Generation intelligent tutor [18] took one step ahead by considering Algebraic word problems and teaching students how to perform symbolization i.e. convert word problems to symbolic problems. Some ITSs [19], [20] consider probability word problems, but to the best of our knowledge, none of those systems addressed the problem of symbolization of those problems and tracing the students errors in this process. This is one of the main contribution of MAST as discussed in the rest of the paper.

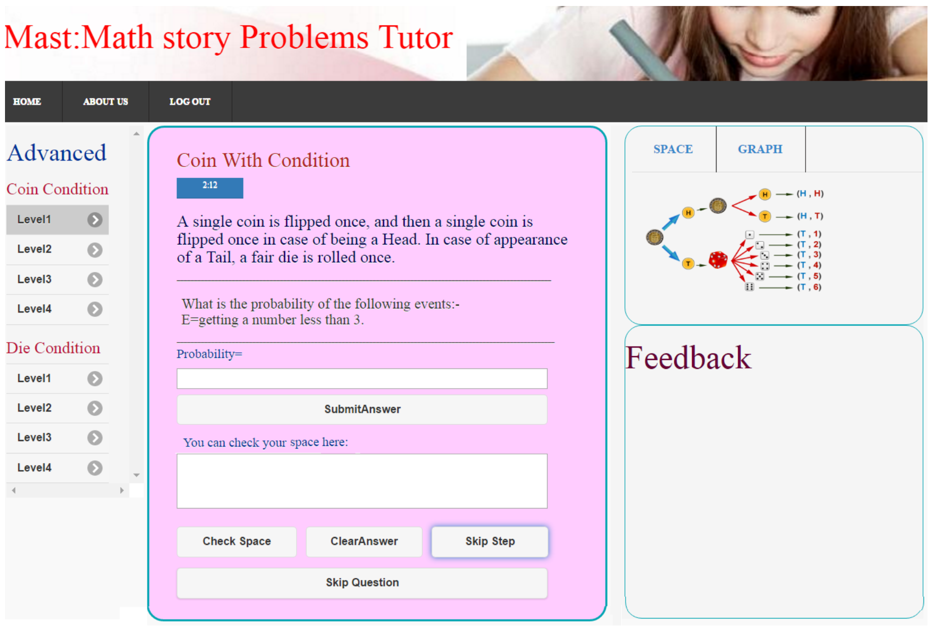
3 The Math Story Problem Tutor (MAST)

As previously noted, MAST is able to generate probability story problems of different contexts, types and difficulty levels for self-paced learning. Examples of problem contexts in MAST are as follows:

- A container contains 5 orange balls and one red ball. If the balls were similar and a person draws a ball randomly, what is the probability that this ball is orange?
- In a class of 18 students, two are girls. If one of the students were randomly selected out of this class, what is the probability that this student is a girl?
- A single die is tossed twice and the numbers on the upper faces are observed, what is the probability of obtaining an odd number?

Due to space limitations, the paper emphasizes the context of rolling a die and tossing a coin since it is generally the most common context. Within each context, the problems are classified according to their difficulty levels. For example, within the context of rolling a die and tossing a coin, the different problem types are classified into four coarse-grained difficulty level categories as follows:

- **Low:** problems involving *one object* such as one coin and one die.
- **Medium:** problems involving *repeated objects* (two or more objects of the same type) such as two dice.
- **High:** problems involving *hybrid objects* (two or more objects of different types) such as one coin and one die.
- **Advanced:** problems involving possibly hybrid objects and a *condition* on the flow of the story (such as rolling a single die or two dice according to the result of tossing a coin).



A single coin is flipped once, and then a single coin is flipped once in case of being a Head. In case of appearance of a Tail, a fair die is rolled once.

What is the probability of the following events:-
E=getting a number less than 3.

Fig. 1. A snapshot of MAST depicting fine-grained difficulty levels of two different problem types within the Advanced difficulty level category.

Within each of those coarse-grained categories, the different problem types are further classified into several fine-grained difficulty levels according to the included operators. Figure 1 shows an example MAST screen depicting two problem types with several fine-grained difficulty levels within the Advanced difficulty level category. The screen also shows an example generated probability word problem. As shown in the figure, the problem is composed of a story that describes a situation involving more than one object with conditions that control the flow of the story. The problem is followed by a question about the probability of an event which is the appearance of a number less than 3 including the operator *less than*. It is worth noting that each MAST problem starts with a question about the sample space. As shown in the figure, the sample space is then displayed graphically throughout the rest of the problem in which the student is provided with one or more probability questions one by one.

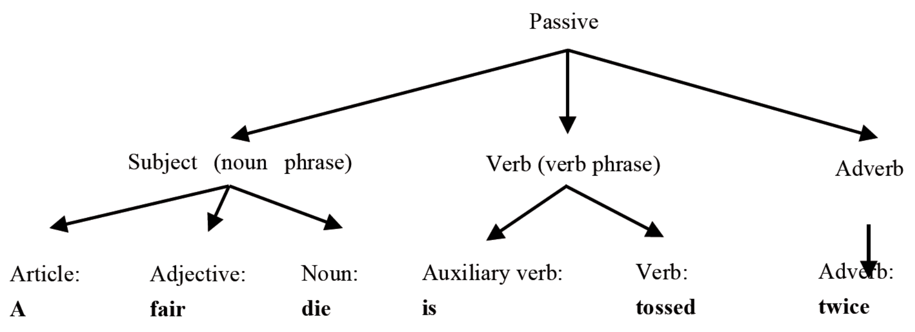


Fig. 2. A parse tree of a passive sentence indicating the grammatical role of each word.

4 Cognitive Modeling and Symbolization in MAST

Typically, the first step to solve a word problem is to represent it using a mathematical model which is also called symbolization [18]. In the probability domain, the mathematical model is the sample space of the problem and the event space (of the event of interest). After symbolization, the probability of the event is estimated based on both the sample space and the event space.

Cognitive modeling simulates human problem solving. In other words, it involves symbolizing a given word problem, and then computing the required probability. As previously noted, MAST includes different categories of word problems with variable difficulty levels. The difficulty levels are based on the number and nature of the objects in the problems and whether their actions are dependent on the outcome of other actions. Cognitive modeling and the corresponding symbolization and probability computation processes depend on the problem category as explained below.

4.1 Generation of the sample space

MAST word problems are generated using the Rhetorical Structure Theory (RST) [21] as a natural language generation technique. Using RST, each possible problem structure is represented using a *rhetorical schema* in which it is recursively decomposed into a set of text spans with identified roles such that the text spans at the lowest level of the schema are single words. A *rhetorical tree* is a realization of a rhetorical schema. In other words, several problems can be described using a single rhetorical schema. Similarly, each text span has several possible structures and each structure (such as a passive sentence for example) can be represented by several possible sentences as long as they are parsed similarly. To illustrate, Figure 2 shows an example parse tree of a passive sentence formed of a noun phrase (representing its subject), a verb phrase (representing its verb) and an adverb. Such a parse tree can describe several sentences that have the same structure but with different wordings such as "**A** fair **die** is tossed **twice**" and "**A** single **coin** is flipped **once**".

The advantage of using RST and identifying possible structures of the text spans is that the grammatical role of each word in a given story problem is known. This allows defining production rules to extract keywords based on their grammatical roles (in their corresponding phrases or sentences). Those keywords are used to identify problem parameters needed for the symbolization process such as the name and count of each included object and any condition on the flow of the problem story. For example, consider the following rule:

If the sentence is a passive sentence

Then the object name is the subject noun and the count of the object is the greatest number represented by the article or the adverb

According to this rule, the important keywords are the subject noun, the article and the adverb. Using those keywords we are able to identify the problem parameters in each sentence, which are the included objects and their count. Similar rules are defined to extract other parameters according to the problem structure.

In the first three categories of problems (Low, Medium and High) the symbolization process of generating the sample space starts by identifying the objects in the problem header and their count. Those parameters are used to trigger corresponding production rules resulting in generating the sample space of each object and then combining the sample spaces of all the objects to generate the final problem sample space. For example, consider the following problem header of a Medium problem category:

A *single* ***die*** *is tossed* ***twice****, then the numbers on the upper faces are observed.*

In this problem, symbolization starts by identifying the object name indicated by the subject noun die. This parameter triggers the corresponding production rule resulting in generating the sample space of the die which is {1,2,3,4,5,6}. Since the adverb indicates two dice, another instance of the die sample space is generated. The two instances are then combined resulting in the following compound sample space:

{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),(3,1),(3,2),(3,3),(3,4),(3,5),(3,6),(4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(5,1),(5,2),(5,3),(5,4),(5,5),(5,6),(6,1),(6,2),(6,3),(6,4),(6,5),(6,6)}

Consider the following example of a High problem category:

Consider an experiment of flipping ***two*** *distinct* ***coins once*** *and tossing* ***a*** *fair* ***die once*** *and afterwards observing the upper faces of the coins and the die.*

In this problem, the keyword *coin* results in generating the sample space of the coin {H,T}. Nevertheless, since the keyword *two* indicates two coins, another instance of this sample space is generated and the two instances are combined resulting in the compound sample space {(H,H),(H,T),(T,H),(T,T)}. Similarly, the keywords *die* and *once* result in generating a sample space of the die {1,2,3,4,5,6}. Those sample spaces are then combined resulting in the hybrid sample space:

{(H,H,1),(H,H,2),(H,H,3),(H,H,4),(H,H,5),(H,H,6),(H,T,1),(H,T,2),(H,T,3),(H,T,4),(H,T,5),(H,T,6),(T,H,1),(T,H,2),(T,H,3),(T,H,4),(T,H,5),(T,H,6),(T,T,1),(T,T,2),(T,T,3),(T,T,4),(T,T,5),(T,T,6)}

Symbolization of an Advanced problem is a little bit harder due to the condition on the flow of the problem story. In such a problem, in addition to the involved object names and count, the extracted problem parameters include the problem condition. For example, consider the following problem header:

***A** single **die** is thrown **once**, and then **a** single **coin** is flipped **once** in case of obtaining an **even** number. In case of obtaining an odd number, **a** single **coin** is flipped **twice**.*

In this problem, the main object, which is the die is first extracted resulting in the generation of the corresponding sample space {1,2,3,4,5,6}. Symbolization of the problem then proceeds as follows:

- The keyword *even* is extracted and used as another parameter to impose a condition on the above sample space resulting in the conditioned space {2,4,6}.
- The keywords *coin* and *once* are extracted resulting in a single instance of the coin sample space {H,T}.
- Those two spaces are combined resulting in the hybrid space {(2,H),(4,H),(6,H),(2,T),(4,T),(6,T)}.
- The complement of the above conditioned space is generated resulting in the space {1,3,5}.
- The keywords *coin* and *twice* are extracted resulting in the generation and combination of two instances of the coin sample space resulting in the space {(H,H),(H,T),(T,H),(T,T)}.
- Those two spaces are combined resulting in the hybrid space {(1,H,H),(1,H,T),(1,T,H),(1,T,T),(3,H,H),(3,H,T),(3,T,H),(3,T,T),(5,H,H),(5,H,T),(5,T,H),(5,T,T)}.
- The two hybrid spaces are then combined generating the problem sample space as follows: {(2,H),(4,H),(6,H),(2,T),(4,T),(6,T),(1,H,H),(1,H,T),(1,T,H),(1,T,T),(3,H,H),(3,H,T),(3,T,H),(3,T,T),(5,H,H),(5,H,T),(5,T,H),(5,T,T)}.

4.2 Generation of the event space

In MAST problems, the students are mainly asked to compute the probabilities of some events. Computing the probability of an event entails generating the event space first. To generate the event space, the operators (combined with their operands) in the corresponding probability question are extracted and used to perform operation on the sample space to select relevant sample space elements. Operations are of three types: simple, compound and complex.

A simple operation is a single operation that checks a corresponding operator directly against the sample space elements to identify those satisfying it. For example, consider the following problem:

*An experiment consists of tossing a fair die once and observing the upper face. What is the probability of obtaining an **odd** number?*

After generating the problem sample space {1,2,3,4,5,6}, the keyword *odd* is extracted using its grammatical role as an adjective. This simple operator triggers a rule for checking the elements of the problem sample space to consider only odd ones resulting in the event space {1,3,5}.

A compound operation is actually a combination of two simple operations that are applied to the problem sample space in turn. Consider the following problem:

*A single die is tossed twice, then the numbers on the upper faces are observed. Determine the probability of the event that the **sum** of the appearing numbers is **less than 8.***

In this problem, the problem sample space is first generated as follows:

{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),(3,1),(3,2),(3,3),(3,4),(3,5),(3,6),(4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(5,1),(5,2),(5,3),(5,4),(5,5),(5,6),(6,1),(6,2),(6,3),(6,4),(6,5),(6,6)}

To generate the event space, the keywords *sum* and *less than 8* (combined with its operand) are generated using their grammatical roles. Those operators are applied on the problem sample space as follows:

- The sum of each element in the above compound space is generated resulting in the set {2,3,4,5,6,7,3,4,5,6,7,8,4,5,6,7,8,9,5,6,7,8,9,10,6,7,8,9,10,11,7,8,9,10,11,12}
- The elements less than 8 are then identified and the corresponding elements in the compound sample space are generated resulting in the event space:

{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,1),(2,2),(2,3),(2,4),(2,5),(3,1),(3,2),(3,3),(3,4),(4,1),(4,2),(4,3), (5,1),(5,2),(6,1)}.

Finally, a complex operation is a combination of two (simple or compound) operations linked by one of De Morgan's operators such as *AND*, *OR* and *AND NOT*. For example, consider the following problem:

*Consider an experiment of flipping two distinct coins once and tossing a fair die once and observing the upper faces of the coins and the die. What is the probability of obtaining a **Tail And** that the appearing number is **even**?*

In this problem, the problem hybrid sample space is first generated as follows:

{(H,H,1),(H,H,2),(H,H,3),(H,H,4),(H,H,5),(H,H,6),(H,T,1),(H,T,2),(H,T,3),(H,T,4),(H,T,5),(H,T,6),(T,H,1),(T,H,2),(T,H,3),(T,H,4),(T,H,5),(T,H,6),(T,T,1),(T,T,2),(T,T,3),(T,T,4),(T,T,5),(T,T,6)}

To generate the event space, the keywords *Tail*, *And* and *even* are generated using their grammatical roles. Those operators are applied on the above problem hybrid sample space as follows:

- The sample space elements including a single Tail are generated resulting in the space {(H,T,1),(H,T,2),(H,T,3),(H,T,4),(H,T,5),(H,T,6),(T,H,1),(T,H,2),(T,H,3),(T,H,4),(T,H,5),(T,H,6)}.
- The sample space elements including an even number are generated resulting in the space
- {(H,H,2),(H,H,4),(H,H,6),(H,T,2),(H,T,4),(H,T,6),(T,H,2),(T,H,4),(T,H,6),(T,T,2),(T,T,4),(T,T,6)}.
- De Morgan's operator *AND* is used to aggregate those spaces by considering the common elements only resulting in the following event space {(H,T,2),(H,T,4),(H,T,6),(T,H,2),(T,H,4),(T,H,6)}.

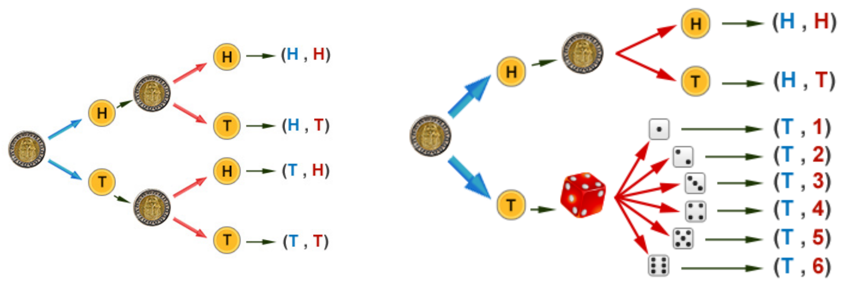


Fig. 3. Visualization of an Advanced category problem where the sample space elements are (a) equiprobable and (b) of different probabilities.

4.3 Probability evaluation

After generating the sample space and event space of a given problem, the probability of the event is computed. In MAST, the answer is written in a fraction form unless it is 0 or 1. MAST is flexible enough to accept any fraction provided that it evaluates to the correct probability.

In the Low, Medium and High problem categories, the probability of each sample space element is computed by dividing one by the number of elements of the sample space. The probability of an event is computed by summing up the probabilities of all the elements of the event space. This also applies in case of some of the Advanced problems such as the problem shown in Figure 3(a). In most cases, this does not apply, because the sample space elements may have unequal probabilities. So, we have to compute the probabilities of the elements in isolation. Consider the following problem:

A fair coin is flipped once, and then a single coin is tossed once in case of obtaining a Head. In case of obtaining a Tail, a single die is thrown once. Calculate the probability of obtaining at least one Tail and not being a perfect square number.

The resulting sample space elements are visualized in Figure 3(b). Since the coin is fair, the probability of each of the first Head and the first Tail is 1/2. On the other hand, the probability of each of the space elements (H,H) and (H,T) is calculated by multiplying 1/2 by 1/2 resulting in 1/4. Similarly, the probability of each of the space elements (T,1) through (T,6) is computed by multiplying 1/2 by 1/6 resulting in 1/12. In this case, the probability of the event is calculated by adding the probabilities of the event space elements {(T,2)(T,3)(T,5)(T,6)} resulting in 1/3.

5 Model Tracing for Diagnosis of Student Errors in MAST

Cognitive modeling allows generating a problem answer through a sequence of steps. Successful matching of the student answer with the cognitive model answer implies that the student may have utilized the same production rules to obtain the answer. On the other hand, an incorrect answer triggers diagnosis techniques of the source of the error. For example, some systems add incorrect rules that represent incorrect actions of the students and assign a relevant feedback message for each such action [22]. MAST employs model tracing for this purpose.

Table 1. Examples of some erroneous student answers and the diagnosis of their errors

Erroneous answers	Interpretation
{(H),(T),(H,H),(H,T),(T,H),(T,T)}	Missing the first object (die)
{(2,H),(4,H),(6,H),(2,T),(4,T),(6,T)}	Missing the third object (coin)
{(1,H),(2,H),(3,H), (4,H),(5,H),(6,H),(1,T), (2,T),(3,T),(4,T),(5,T),(6,T)}	Missing the flipping count of the third object
{(1,H),(3,H),(5,H),(1,T),(3,T),(5,T) (2,H,H),(4,H,H),(6,H,H),(2,H,T),(4,H,T),(6,H,T), (2,T,H),(4,T,H),(6,T,H),(2,T,T),(4,T,T),(6,T,T)}	Exchanging the *even* space elements with its *odd* complement space elements

Model tracing in MAST starts by examining the student symbolization of the story problem. Providing a correct answer to a sample space question means that the student applied all related symbolization rules correctly. On the other hand, providing a wrong answer stimulates the tracing algorithm. This algorithm checks the effect of incorrect application of the problem keywords and their sequence on the student answer. Erroneous answers are generated by several means such as missing one of the objects or exchanging a conditioned space with its complement. In case the student answer matches one of the erroneous answers, the cause of the error is used as an interpretation of the cause of the student error. For example, consider the following problem header:

A single die is thrown once, and then a single coin is flipped once in case of obtaining an even number. In case of obtaining an odd number, a single coin is flipped twice

It is obvious that the correct sample space of this problem is as follows:

{(2,H),(4,H),(6,H),(2,T),(4,T),(6,T),(1,H,H),(3,H,H),(5,H,H),(1,H,T),(3,H,T),(5,H,T),(1,T,H),(3,T,H),(5,T,H),(1,T,T),(3,T,T),(5,T,T)}.

Table 1 shows several erroneous answers to this problem (erroneous symbolization) and their corresponding interpretation.

In case of a probability question, the student answer should be provided in the form of a fraction. The cause of an incorrect answer to such a question is more challenging to interpret. In turn, the student is allowed to check the event space as shown in Figure 1. This leads from the fact that wrong derivation of the event space is amongst the main causes for wrong computation of the probability.

Interpretation of the student error in providing the event space is based on the number of operators in the probability question. In case of one operator (simple operation), the student error is interpreted by checking the student answer elements to identify the elements that do not satisfy the operator. For example, consider the following problem:

Consider an experiment of throwing a single die once and observing the number appearing on the upper face. Calculate the probability of the event of obtaining a ***perfect square*** *number.*

Suppose that the student solves this problem incorrectly and provides the following event space {1,2,4}. Model tracing checks the student answer and figures out that number 2 provided in the space does not satisfy the operator *perfect square*. Consequently, the system provides a feedback indicating this error.

In case of multiple operators (compound or complex operations), on the other hand, the elements of the student answer are checked to indicate whether or not they satisfy all the operations. For example, consider the following problem:

A fair coin is tossed once, then two distinct dice are rolled once. The upper faces of the coin and the dice are observed. Calculate the probability of getting a Tail and not the absolute difference of the appearing numbers is an even number.

Suppose that the student solves this problem incorrectly and provides the following event space {(H,1,2), (T,2,1), (T,1,4), (T,4,1), (T,1,6), (T,6,1), (T,2,3), (T,3,2), (T,2,5), (T,5,2), (T,3,4), (T,4,3), (T,3,6), (T,6,3), (T,4,5), (T,5,4), (T,5,6), (T,6,5)}. By checking the elements of the student answer, the model tracing algorithm interprets that the cause of the error is that the element (H,1,2) does not satisfy the first operator, since it does not include a Tail.

Another example, suppose that the student solves the problem incorrectly and provides the following event space {(T,1,2), (T,2,1), (H,1,5), (T,4,1), (T,1,6), (T,6,1), (T,2,3), (T,3,2), (T,2,5), (T,5,2), (T,3,4), (T,4,3), (T,3,6), (T,6,3), (T,4,5), (T,5,4), (T,5,6), (T,6,5),(T,6,2)}. In this case, the model tracing algorithm infers that the cause of the error are the elements (H,1,5) and (T,6,2) since (H,1,5) does not satisfy the *Tail* operator and both elements do not satisfy the *AND NOT* operator.

6 Evaluation

To evaluate the rule-based expert system we prepared two groups of 15 students each in the age range of 17 to 21. Each group was allowed to take a pre-test exam including a problem from each type in MAST. The first group (experimental group) was allowed to practice using MAST for 2 days at their own pace, while the second group (control group) was allowed to practice using the textbook only for the same period of time. Afterwards, the two groups were allowed to take a posttest exam. The posttest exam included the same number and types of problems as the pretest exam, but the problems and their order were different. The grades of the students in each exam pair were recorded and subtracted to compute the gain. The results are shown in Table 2. As shown in the table, the mean of the gain is significantly higher in the experimental group in comparison to the control group.

Table 2. Analysis of the gain of the grades of the students in the experimental group and the control group

Group	Mean	Variance	Standard deviation	median
Experimental group	11.33	36.66	6.05	10
Control group	3.80	4.02	2.00	3

We also compared the gains of the two groups using *Mann-Whitney test*. We obtained a critical value of *U* of 64 and a *p-value* of 3.33E-04 indicating that the result is significant at $p < 0.05$. In other words, there is a significant difference between the gains of the two groups indicating a positive effect of MAST on tutoring the students and improving their problem solving skills. We also assessed the reflection of the students in the experimental group. The students were asked to provide a response on a *Likert* scale between 1 (very unsatisfactory) and 5 (very satisfactory) and the results are shown in Table 3. As shown in the table, the students believe the system can increase their interest in learning. Additionally, it is conducive to improving self-learning and self-paced learning capabilities.

Table 3. Reflection of the students in the experimental group

Evaluation indicator	average
Increase the student interest in learning	4.934
Conducive to improving self-learning capability	4.667
Conducive to improving self-paced learning capability	4.467

An important aspect of MAST expert system is its ability to handle sample space and event space questions that assess the ability of the students to symbolize probability word problems and convert them into a mathematical format. As previously noted, the expert system not only solves such problems, but also diagnoses the student errors and provides relevant help. Hence, the effect of MAST expert system on tutoring the students this capability was assessed in isolation for the two above groups. In each problem, the students were asked to solve the initial sample space question in addition to the optional event space question corresponding to each probability question within the problem. The grades of the students in solving those questions throughout the problems of the two exams were recorded and subtracted for each pair to compute the gain. We compared the gains of the two groups using *Mann-Whitney test*. We obtained a critical value of *U* of 63 and a *p-value* of 2.29E-04 indicating that the result is significant at $p < 0.05$. In other words, there is a significant difference between the gains of the two groups indicating a significant positive effect of MAST expert system on tutoring the students this capability.

7 Conclusion

This paper presented MAST, a Web-based ITS of probability story problems. MAST is able to generate problems of different contexts, types and difficulty levels. The paper is mainly concerned with cognitive modeling and model tracing in the expert system module of MAST. Symbolization of the word problems and tracing the student errors in this process are the main contributions of the paper. The expert system module is able to interpret the student answer and figure out the cause of errors. The module, thus, can provide the correct answer and personalized feedback according to the student error. Evaluation results have shown the ability of MAST to effect considerable improvement in the probability story problem solving skills of the students.

8 References

[1] Corbett, A., Koedinger, K. & Anderson, J. (1997). Intelligent tutoring systems. In Handbook of Human-Computer Interaction, Helander, T., Landauer, K. & Prabhu, P. (Eds.), Elsevier Science, Amsterdam, pp. 849-874. https://doi.org/10.1016/b978-044481862-1.50103-5

[2] Woolf, B. (2008). Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-Learning, Morgan Kaufmann, San Francisco, CA, USA.

[3] Aleven, V., Roll, I., McLaren, B. & Koedinger, K. (2016). Help helps, but only so much: Research on help seeking with intelligent tutoring systems. International Journal of Artificial Intelligence in Education, 26:205-223. https://doi.org/10.1007/s40593-015-0089-1

[4] Arevalillo-Herráez, M., Arnau, D. & Marco-Giménez, L. (2013). Domain-specific knowledge representation and inference engine for an intelligent tutoring system. Knowledge-Based Systems, 49:97-105. https://doi.org/10.1016/j.knosys.2013.04.017

[5] Nkambou, R. (2010). Modeling the domain: An introduction to the expert module. In Advances in Intelligent Tutoring Systems, R. Nkambou et al. (Eds.), Springer-Verlag, Berlin Heidelberg, pp. 15-32. https://doi.org/10.1007/978-3-642-14363-2_2

[6] Anderson, J. (1982). Acquisition of cognitive skill. Psychological Review, 89(4):369-406. https://doi.org/10.1037/0033-295X.89.4.369

[7] Mitrovic, A., Koedinger, K. & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modeling. In Proceedings of the 9th International Conference on User Modeling, Johnstown, PA, USA.

[8] Garshol, L. (2004). Metadata? Thesauri? Taxonomies? Topic maps! Making sense of it all. Journal of Information Science, 30(4):378-391. https://doi.org/10.1177/0165551504045856

[9] Kumar, A. (2006). Using enhanced concept map for student modeling in programming tutors. In Proceedings of the 19th International FLAIRS Conference on Artificial Intelligence, Melbourne Beach, FL, USA.

[10] Bontcheva, K. & Dimitrova, V. (2004). Examining the use of conceptual graphs in adaptive web-based systems that aid terminology learning. International Journal on Artificial Intelligence Tools, 13(2):299-331. https://doi.org/10.1142/S0218213004001569

[11] VanLehn, K., Lynch, C., Schultze, K., Shapiro, J., Shelby, R., Taylor, L., Treacy, D., Weinstein, A. & Wintersgill, M. (2005). The Andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence in Education, 15(3).

[12] Crowley, R. & Medvedeva, O. (2006). An intelligent tutoring system for visual classification problem solving. Artificial Intelligence in Medicine, 36:85-117. https://doi.org/10.1016/j.artmed.2005.01.005

[13] Koedinger, K. & Aleven, V. (2007). Exploring the assistance dilemma in experiments with cognitive tutors. Educational Psychology Review, 19:239-264. https://doi.org/10.1007/s10648-007-9049-0

[14] Anderson, J. (1988). The expert module. In Foundations of Intelligent Tutoring Systems, Polson, M. & Richardson, J. (Eds.), Hillsdale, NJ: Erlbaum, pp. 21-53.

[15] Koedinger, K. & Sueker, E. (1996). PAT goes to college: Evaluating a cognitive tutor for developmental mathematics. In Proceedings of the International Conference on Learning Sciences, Evanston, Illinois, pp. 180-187.

[16] Jaques, P., Seffrin, H., Rubi, G., de Morais, F., Ghilardi, C., Bittencourt, I. & Isotani, S. (2013). Rule-based expert systems to support step-by-step guidance in algebraic problem solving: The case of the tutor PAT2Math. Expert Systems with Applications, 40:5456-5465. https://doi.org/10.1016/j.eswa.2013.04.004

[17] Nicaud, J., Bouhineau, D. & Huguet, T. (2002). The Aplusix-editor: A new kind of software for the learning of Algebra. In Proceedings of the 6th International Conference on Intelligent Tutoring Systems, 2002, pp. 178-187. https://doi.org/10.1007/3-540-47987-2_22

[18] Heffernan, N., Koedinger, K. & Razzaq, L. (2008). Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. International Journal of Artificial Intelligence in Education, 18(2):153-178.

[19] Holling, H., Bertling, J. & Zeuch, N. (2009). Automatic item generation of probability word problems. Studies in Educational Evaluation, 35:71-76. https://doi.org/10.1016/j.stueduc.2009.10.004

[20] Theune, M., Rookhuiszen, R., Akker, R. & Geerlings, H. (2011). Generating varied narrative probability exercises. In Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications, Portland, Oregon, pp. 20-29.

[21] Taboada, M. & Mann, W. (2006). Rhetorical structure theory: Looking back and moving ahead. Discourse Studies, 8(3):423-459. https://doi.org/10.1177/1461445606061881

[22] Aleven, V., McLaren, B., Sewall, J., Koedinger, K. (2009). Example-tracing tutors: A new paradigm for intelligent tutoring systems. International Journal of Artificial Intelligence in Education, 19(2):105-154.

9 Authors

Nabila Khodeir is a researcher in the Informatics department at the Electronics Research Institute, Cairo, Egypt. Her research interests include intelligent tutoring systems, user modelling and natural language processing. She earned her Ph.D. and ME from the Electronics and Communications department at Cairo University.

Hanan Elazhary earned her B.Sc. and M.Sc. degrees from the Department of Electronics and Communications Engineering, Cairo University. She earned her Ph.D. degree in Computer Science and Engineering from the University of Connecticut, USA. Currently, she is an associate professor in the Computer Science Department, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia and the Computers and Systems Department, Electronics Research Institute, Cairo, Egypt. Here research interests include distributed systems, software engineering and intelligent tutoring systems.

Nayer Wanas is an associate professor of informatics at the Electronics Research Institute, Giza, Egypt. His research interests include data mining and learning systems. He has also worked as a researcher at Microsoft Research on mining social data. Dr. Wanas received his Ph.D. in Systems Design Engineering from the University of Waterloo in 2003, and his Bachelors and ME from the Electronics and Communications department at Cairo University in 1992 and 1996 respectively. He has over 70 technical publications in top journals and conferences in addition to several book chapters and has several patents related to his research and applied work. He has served as a reviewer of several prestigious journals and on several conference program committees and industrial panels. He is a Senior IEEE member and ACM member.

Article submitted 23 December 2016. Published as resubmitted by the authors 16 February 2017.