Fine-Grained Cognitive Assessment Based on Free-Form Input for Math Story Problems

Bastiaan Heeren[1(✉)], Johan Jeuring[1,2], Sergey Sosnovsky[2], Paul Drijvers[3], Peter Boon[3], Sietske Tacoma[3], Jesse Koops[4], Armin Weinberger[5], Brigitte Grugeon-Allys[6], Françoise Chenevotot-Quentin[6], Jorn van Wijk[1], and Ferdinand van Walree[1]

[1] Faculty of Management, Science and Technology, Open University of the Netherlands, P.O. Box 2960, 6401 DL Heerlen, The Netherlands
bastiaan.heeren@ou.nl
[2] Department of Information and Computing Sciences, Universiteit Utrecht, Utrecht, The Netherlands
[3] Freudenthal Institute, Utrecht University, Utrecht, The Netherlands
[4] Cito, Arnhem, The Netherlands
[5] Department of Educational Technology, Saarland University, Saarbrücken, Germany
[6] Laboratoire de Didactique André Revuz, Université Paris Est Créteil, Paris, France

Abstract. We describe an approach to using ICT for assessing mathematics achievement of pupils using learning environments for mathematics. In particular, we look at fine-grained cognitive assessment of free-form answers to math story problems, which requires determining the steps a pupil takes towards a solution, together with the high-level solution approach used by the pupil. We recognise steps and solution approaches in free-form answers and use this information to update a user model of mathematical competencies. We use the user model to find out for which student competencies we need more evidence of mastery, and determine which next problem to offer to a pupil. We describe the results of our fine-grained cognitive assessment on a large dataset for one problem, and report the results of two pilot studies in different European countries.

Keywords: Math story problems · Step-based assessment
Free-form input · Solution strategies · User modelling

1 Introduction

Competence in mathematics has been identified at EU level as one of the key competencies for personal fulfilment, active citizenship, social inclusion, and employability in the knowledge society of the 21st century.[1] In 2009, concerns

[1] Mathematics Education in Europe: Common Challenges and National Policies, 2011.

V. Pammer-Schindler et al. (Eds.): EC-TEL 2018, LNCS 11082, pp. 262–276, 2018.
https://doi.org/10.1007/978-3-319-98572-5_20

about low student performance led to the adoption of an EU-wide benchmark in basic skills, which states that 'by 2020 the share of 15-year-olds with insufficient abilities in reading, mathematics and science should be less than 15%'.[2] An extensive review of research evidence on 'what works for children with mathematical difficulties' has concluded that 'interventions should ideally be targeted towards an individual child's particular difficulties' [4].

In a time where many European countries face shortages of teachers,[3] it is not to be expected that time of teachers available for assessment and determining competencies will increase. ICT can be of help here. The actions of a pupil when working in a digital environment can be collected in, and interpreted by, a so-called user model. The model describes the current level of mathematics achievement of a pupil. It is essential that this model not only analyses final answers, but also intermediate steps [13]. Intermediate steps contain essential information about how a pupil arrived at an answer (the method that was used), and can be used to identify misconceptions and pinpoint errors. We thus want to analyse intermediate steps to get precise diagnostic information.

This paper describes an approach to fine-grained cognitive assessment, including information about steps, of free-form input to math story problems on the domain of 'Relationships' targeting 15-year-olds. Figure 1 shows such a problem in Numworx, which we use as our digital assessment environment. We recognise steps and solution approaches in free-form input and use this information to update a user model of mathematical competencies. We use the user model to find out for which competencies we need more evidence of mastery, or proof of absence of mastery, and determine which next problem to offer to a pupil. We describe the results of our fine-grained cognitive assessment on a large dataset for one problem, and report the results of two pilot studies in different European countries. The contributions of this paper are:

- a novel approach to fine-grained cognitive assessment of free-form solutions to math story problems;
- a novel user modelling approach that uses the results from the fine-grained assessment;
- the results of applying our fine-grained assessment to a large data set for a single task, and for several smaller datasets containing solutions for multiple tasks.

This paper is organized as follows. In Sect. 2 we review several cognitive assessment methods, and motivate our focus on free-form input. Section 3 describes a high-level architecture for our approach. Section 4 gives the competencies we assess, and illustrates these with one particular task: the 'Magical trick'. Section 5 illustrates the various components of our architecture with instances for this task, and Sect. 6 discusses the results of applying our components to a large dataset for the magical trick task, and the results of two small-scale pilots with multiple tasks. Section 7 concludes the paper and discusses future work.

[2] Strategic Framework for European Cooperation in Education and Training, ET 2020.
[3] Key Data on Education in Europe 2012.

2 Supporting Fine-Grained Cognitive Assessment

Conventional assessment tests developed in the traditions of psychometrics aim at supporting high-stakes decisions such as selection, placement, or licensing. In these circumstances, the focus of the test design is made on characteristics such as validity and reliability. The results of such assessment tests are usually unidimensional: a single value on a single scale [1]. While reliable ranking of test takers is important, a single score provides little information about the source of potential learning problems that have prevented a test taker from scoring high on the test. In this paper, the focus is made less on the absolute reliability of the test, and more on obtaining the detailed picture of a student's strengths and weaknesses. As we will explain later, such a fine-grained cognitive assessment needs to be organized on the basis of free-form answers students give to algebraic story problems, which adds another layer of complexity. While the inference of user knowledge based on the evidence produced by the result of solution analysis is rather straightforward, it is the analysis of the free-form student input that poses the biggest challenge for the cognitive assessment mechanism.

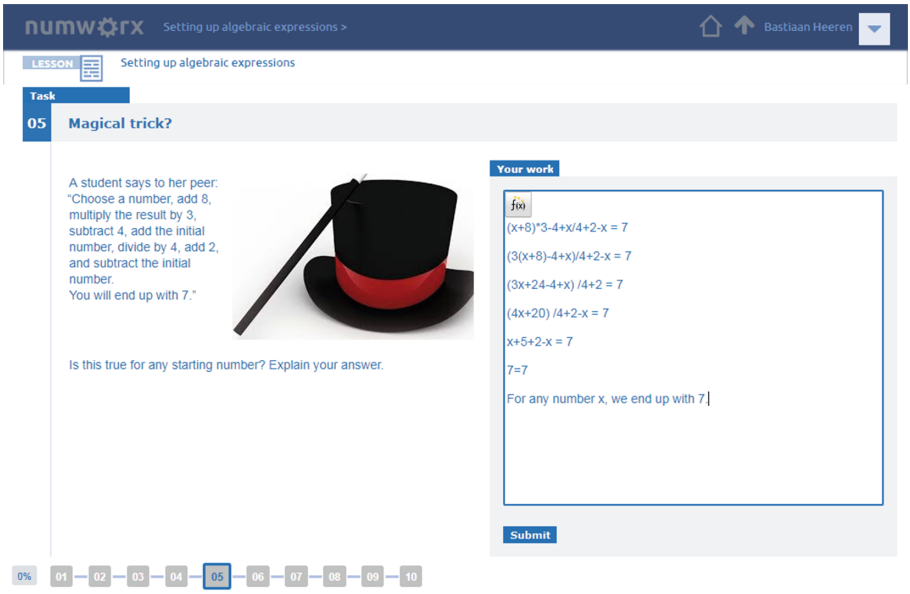

Fig. 1. The 'Magical trick' math story problem in a digital assessment environment

A direct way to facilitate fine-grained cognitive assessment at the solution analysis phase is to use plain assessment exercises that solicit easily verifiable input from students, such as multiple-choice questions. In this case, the potential uncertainty at the analysis phase is minimal: a student answer is either correct or

not, and it is clear which concept is responsible for the outcome. One example of building an adaptive cognitive assessment based on such exercises is SIETTE [3].

However, it is often the case that the underlying assessment method requires a more advanced type of exercises that engages students in complex tasks and requires application of multiple concepts over several solution steps. One way to address this problem is to keep the solution analysis phase simple by allowing students to do all the intermediate computations outside the system, and requiring only the final answer. The uncertainty is then transferred to the part of the system intelligence that defines the concepts responsible for student mistakes. Often, the 'blame' is simply shared among the related concepts. For example, a single self-assessment exercise may involve more than a dozen of concepts [11].

A more reliably way to detect the concept(s) responsible for a mistake made in multi-step exercises is to structure the interaction between the student and the system. In this case, a student is restricted by the interface in what she can enter, and the source of a mistake is easier to identify. Narciss et al. [9] provide one example of such an approach based on a dedicated interface element, where a student chooses the type of the operation before performing it. The Andes system structures the entire student solution so that at every step the operation, its operands, and the purpose of the operation is known to the system [2].

Finally, a system can ask additional follow-up questions that address intermediate steps of the student solution, thus identifying the source of a possible mistake. ASSISTment is one of the systems that employ this approach [5].

Unfortunately, neither of the listed methods is fully suitable. 'Sharing the blame' between potentially responsible concepts does not necessarily guarantee modelling accuracy. Structuring and restricting interaction inescapably provides scaffolding to a student, which is fine for a tutoring system, but is less desirable in an assessment scenario. Finally, follow-up questions unnecessarily extend the assessment session, and can also potentially reveal knowledge to test takers. Our research attempts to go beyond the state of the art by employing a range of AI techniques for analysing free-form student input to math story problems in an assessment setting.

3 A High-Level Architecture

Our goal is to support fine-grained cognitive assessment based on free-form input for math story problems. For this purpose, we analyse free-form student input to find out which steps a student takes when solving a task, and we use the information we find to update the user model, and to determine which next task to offer to the student. This section introduces the main components of the pipeline we use for assessment.

Tutoring systems consist of an outer loop and an inner loop [12]. The inner loop supports a pupil solving a task by taking steps to reach a solution. In the setting of assessing competencies in mathematics, the inner loop is responsible for analysing the steps that a pupil takes, and recognising the high-level solution approach used by the pupil. Giving hints and feedback, thereby providing

opportunities for learning, is not part of an assessment system. The outer loop selects the next task to offer to a pupil. This selection is based on information that is stored in a user model. In the case of assessment, the goal is to quickly find out competencies and misconceptions from a pupil by collecting evidence of mastery, or absence of mastery. Figure 2 identifies the main components needed for assessment, and the flow of information between these components. Next, we describe each of the components.

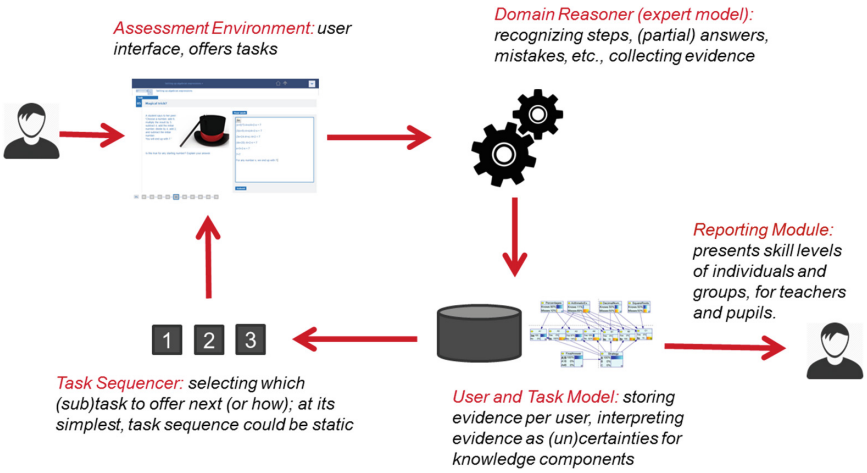

Fig. 2. Information flow in the outer loop

The *assessment environment* is the user interface responsible for offering the tasks to pupils. Such an environment manages user accounts for teachers and pupils, and may offer a teacher area with access to student and class work. For tasks in mathematics, some special tools are needed, for example a formula editor, a calculator, and a graphing tool.

The *domain reasoner* component (also known as the expert knowledge module [10]) contains expert knowledge in the task domain (i.e., the domain of relationships) and uses this knowledge for reasoning about the steps that together form the answer. Input is analysed at different levels of granularity, in particular at the fine-grained step level at which the use of variables, calculational mistakes, sloppy notation, and precedence errors can be detected (among other types of steps), and the high-level solution approach, which may be algebraic, numerical, only partially correct, etc. The parts that are recognised are translated to evidence for the competencies in which we are interested.

The *user and task models* are responsible for the inference, storage and update of student knowledge based on the evidence collected by the domain reasoner. To help manage the potential uncertainty of the diagnosis produced by the domain reasoners, these models are represented as Bayesian Networks (BNs). The user model structures concepts and competencies into a hierarchy,

and the task models relate concepts and competencies involved in a particular assessment task to its solutions steps and the characteristics of the final answer.

The *task sequencer* determines which task should be used next in the assessment. For this, information from the user model is used to calculate which task can best be used to remove uncertainties (in the user model) about competencies. At its simplest, a task sequencer offers tasks in a predetermined order.

The *reporting module* presents the skill levels of individuals in an easy-to-understand way (such as a skill-meter), targeting both teachers and pupils. The module can aggregate information and display information about groups.

A learning environment is a highly complex educational software applications [8]. The technology for calculating diagnostics and building a detailed user model is offered as an open set of services that can be used by multiple learning systems. A service-oriented approach promotes large-scale reuse and counteracts the complexity found in educational software applications [7]. The service-oriented approach, and the integration of these services into existing systems, is one of the innovative aspects of our approach.

4 Tasks and Competencies

Our assessment is based on the answers of a pupil on a set of tasks. The tasks concern the domain of Relationships and target 12–15 years old pupils. In particular, the tasks involve setting up algebraic expressions, equations, and inequalities, as well as simplifying and solving them. The tasks require multi-step solutions, and usually there are multiple ways to solve these tasks. The assessment presented here consists of ten tasks in the domain of Relationships. In this section we introduce the tasks, the competencies they address, and the ways they can be solved by means of an example.

We want to assess the learning goals R1–R3 in the domain of Relationships listed in Table 1. The diagnostic assessment analyses each pupil's answer on the three dimensions described above, but not only by means of correct/incorrect. The diagnostic system provides a set of codes that characterize the answer according to an a priori analysis.

Next, for each of the tasks, the possible solution steps and alternative routes are described, as well as the mistakes students might make. Consider, for example, the 'Magical trick' task, which is also shown in Fig. 1:

> A student says to her peer: 'Choose a number, add 8, multiply the result by 3, subtract 4, add the initial number, divide by 4, add 2, and subtract the initial number. You will end up with 7.'
> Is this true for any starting number? Explain your answer.

The magical trick task is a rich task from a diagnostic point of view. Its goal is to identify whether or not a student is able to generalize and prove a property (R1 and R3) with algebraic strategies. It also provides information about the types of connections (R2) between two representations (in this case a numerical and an algebraic representation), and about the arguments used by a student.

Table 1. Codes for characterizing answers in the domain of Relationships [6]

R1: Construct algebraic objects, e.g., set up expressions, formulas, equations
- Correct global (R11), or step-by-step (R12) algebraization
- Incorrect algebraization (R13)
- Numerical solution (R14), either global (R141) or step-by-step (R142)

R2: Recognise and relate different representations of a mathematical relationship
- Correct relation between two representations with congruent rules (R21)
- Incorrect relation between two representations without reformulation (when non-congruent) (R23)
- Incorrect relation between two representations with schematization (R24)

R3: Calculate and simplify, e.g., expand/factor algebraic expressions, solve equations
- Correct calculation with argumentation and correct semantic and syntactic rules (R31), or without argumentation (R32)
- Incorrect with false rules (R33): errors of parentheses (R332), errors of signs (R333)
- Incorrect with operator priority or concatenation (R34)

There are (at least) two strategies to solve this task: an arithmetic strategy using a particular number (Table 2), and an algebraic strategy that involves a variable (Table 3). For both strategies, we distinguish between a global approach (first set up the complete expression, then perform the simplifications) and a step-by-step approach (set up the expression for the next step and simplify). The tables also present some mistakes that illustrate incorrect techniques.

5 Components

This section discusses two components for assessing solutions in more detail.

5.1 Domain Reasoner

The domain reasoner receives free-form input, determines which steps are taken, and tries to recognise the solution strategy. Based on this analysis, competencies and misconceptions are determined. The analysis proceeds in three phases:

1. Extract mathematical expressions from the textual input, ignoring most of the natural language;
2. Parse the extracted expressions and equations into (structured) mathematical objects;
3. Recognise the solution strategy by parsing the sequence of mathematical objects (i.e., approach strategy recognition as a parsing problem).

Numworx has a formula editor for entering mathematical content. Pupils are allowed to use this editor, which is particularly useful for entering square roots, powers, and other operations that do not have an obvious textual equivalent. Each phase will be described in more detail below.

Math Extraction (Phase 1). A pupil enters input as free text and may use a formula editor that contains mathematical symbols. Learning environments have precise information of what a pupil wrote. We use the MathML standard for transferring the semantics of these graphical mathematical representations from the learning environment to the services.

Table 2. Arithmetic strategy (for number 5) and possible mistakes

Solution	Reasoning	Coding
`((5+8)*3-4+5)/4+2-5 = 7`	Correct arithmetic strategy with global expression, but with a missing generalization for any starting number	R141, R21, R31
`5+8 = 13; 13*3 = 39; 39-4 = 35; 35+5 = 40; 40/4 = 10; 10+2 = 12; 12-5 = 7`	Correct arithmetic strategy with step-by-step approach, but with a missing generalization for any starting number	R142, R22, R31
`5+8*3-4+5/4+2-5 = 7`	Erroneous arithmetic strategy with global expression without parentheses	R14, R23, R33
`5+8 = 13*3 = 39-4 = 35+5 = 40/4 = 10+2 = 12-5 = 7`	Erroneous arithmetic strategy with step-by-step calculations	R14, R24

Since the input from a pupil may contain malformed mathematical expressions, it is not always possible to represent the pupil's input in a standard mathematical representation such as MathML Content. Thus, we allow a learning environment to send input following a subset of the MathML Presentation language. Lexical analysis converts MathML code back to plain text. In this way, a pupil can use specialized symbols such as the root symbol in expressions, while we handle everything as plain text during the recognition phase.

For each task, parts of the math extraction phase can be specialized. For example, some tasks suggest the use of multi-letter variables such as `dist` and `cost`: these variables can be whitelisted. In other tasks there is no distinction between uppercase and lowercase characters, hence we allow both. Furthermore, depending on the language specified in the request, some pre-processing is performed. For example, German words such as `mal`, `plus`, `quadrat`, and `hoch` are converted to their mathematical representations ($*$, $+$, `^2`, and `^`, respectively). Currently, we support English, German, French, and Dutch.

Certain symbols have multiple interpretations. For instance, compare `x+3` (`x` is a variable) with `3x5=15`, in which `x` is probably used to denote multiplication. Similar ambiguities arise for certain punctuation symbols. We are careful not to blow up the search space by considering all interpretations, but instead use heuristics to resolve most of the ambiguous interpretations. For example, any use of `x` surrounded by numbers is interpreted as multiplication. This approach seems to work quite well, but does not prevent misinterpreting expressions such as `1/2 x a x b`. Tasks for which variable `x` has no obvious meaning can be configured to always interpret it as multiplication.

Parsing Expressions (Phase 2). Parsing expressions is rather straightforward, although we do have to take care of equations that are incorrectly chained, such as `5+8 = 13*3 = 39` (see the possible mistakes in Table 2), which we split into two equations with an annotation for the incorrect chaining. Parsed equations are particularly helpful, because they can be checked for equality to spot mistakes (e.g., `5+8 = 40`).

Table 3. Algebraic strategy and possible mistakes

Solution	Reasoning	Coding

`(x+8)*3-4+x)/4+2-x`
`= (3x+24-4+x)/4+2-x`
`= (4x+20)/4+2-x`
`= x+5+2-x`
`= 7`
	Algebraic proof with global expression	R11, R21, R31

`(x+8)*3 = 3x+24`
`3x+24-4 = 3x+20`
`3x+20+x = 4x+20`
`(4x+20)/4 = x+5`
`x+5+2 = x+7`
`x+7-x = 7`
	Algebraic proof with step-by-step approach	R12, R31

`(x+8)*3-4+x/4+2-x` or `(x+8*3-4+x)/4+2-x` or `x+8*3-4+x/4+2-x`	Order or priority of operations is missed	R13, R23

`x+8*3-4+x/4+2-x`
`= 2x+20/4+2-x`
`= 2x+5+2-x`
`= x+7`
	Simplification mistakes, such as ignoring parentheses or priority rules	R13, R23, R33

`(x+8)*3 = 3x+24 = 27x`
`27x-4+x = 24x`
`24x+2-x = 23x+2 = 25x`
	Simplification mistakes, such as dilemma process-product:
	`a+b` $\rightarrow$ `ab`
		R13, R21, R341

Strategy Recognition (Phase 3). In the third phase we try to recognise solution strategies. The general approach is to consider recognition as a parsing problem, and to express the solution strategies as context-free grammars. During the parsing, we keep track of variables that are introduced, numbers that are chosen, and definitions that can be propagated (e.g. `d = a^2+14`, followed by $4 * \texttt{d}$).

The recogniser must be flexible enough to recover from different types of mistakes and imperfections by providing some error correction, for example steps that are taken implicitly, (basic) calculation mistakes, algebraic misconceptions, missing parentheses or incorrect options when modelling, and so on. See Tables 2 and 3 for more mistakes that are recognised during this phase. Note that there is a trade-off between flexibility and computation time: we use real data, collected from earlier user studies and pilots, to calibrate the recogniser.

5.2 Task Model and User Model

Not every student answer provides high-quality input for the domain reasoner to analyse. Hence, the evidence produced by the analysis step can be scattered. It is often clear whether or not a student has given a correct final answer and which strategy a student has applied. Sometimes, individual steps are clearly indicated in the solution and can be easily diagnosed. However, as a rule, we have assumed (and the following evaluation confirms this) that the diagnosis evidence is not guaranteed for any step of the solution, nor for the overall answer. Yet, the implemented solution has to produce a detailed cognitive assessment under such uncertainty. Due to these consideration, we have decided to employ BNs as a well-known mechanism to support inference under uncertainty.

BNs are widely used for modelling students' knowledge based on students' responses to multi-step learning exercises [1,2]. We follow this tradition by representing both the model of student knowledge and the task models as probabilistic networks. However, the fact that users in the assessment environment produce unrestricted, unstructured, unscaffolded input, adds another layer of uncertainty into our inference pipeline and reflects on the design of the task modelling BNs.

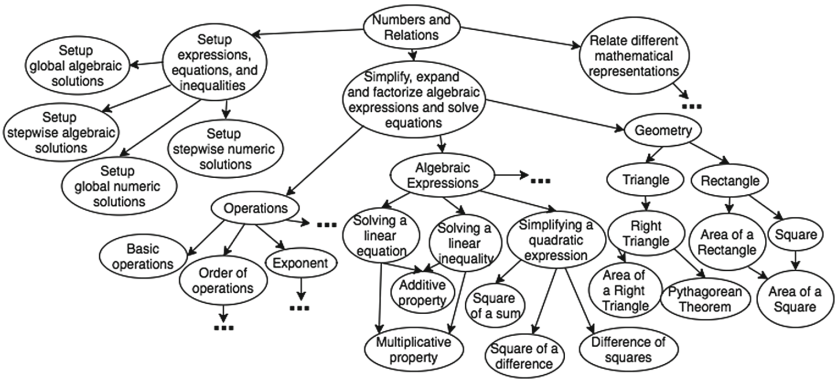

Fig. 3. Part of the user model

Figure 3 visualizes the upper-level structure of a first version of the user model. At the top of the hierarchy, we have three learning goals (R1–R3, see Table reftable:codes). They are further categorized into smaller competencies and concepts that participate in individual models of tasks. For example, Fig. 4 represents a tentative BN for the 'Magical trick' exercise. The top nodes represent competencies corresponding to setting up a solution (R1): these nodes connect the task model with the user model. The top nodes in Fig. 4 are connected to more general characteristics of the solution (correctness and properties of the chosen strategy). These characteristics are more likely to be diagnosed. At the same time, every combination of these characteristics is probabilistically related to a certain sequence of solutions steps (there are four possible strategies to

solve the 'Magical trick' exercise, corresponding to four such sequences of steps). Within each sequence, the probability of a next step to be applied correctly depends on the previous step and the corresponding concept nodes that model the probability that a student has mastered these concepts. When a student starts working on a task, the prior probabilities for all concept and competency nodes are copied from this student's user model. Once a student submits a solution, some of the nodes are set to 1 or 0 in the task model, depending on the results of the analysis phase. This triggers the probability update of the concept and competency nodes given the new evidence. Finally, the updated probabilities are carried over to the user model; they will inform the task model of the next assessment task the student will attempt.

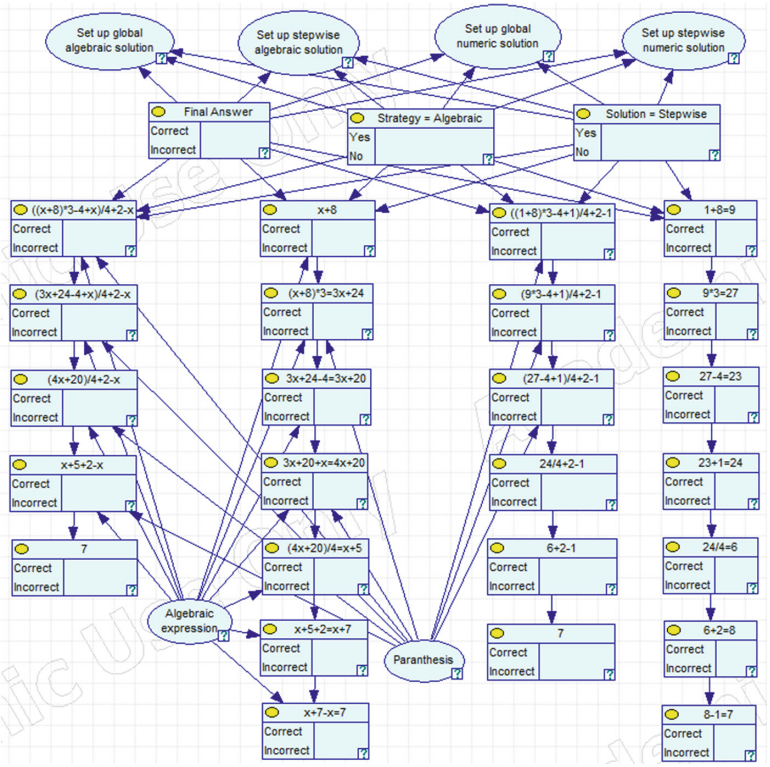

Fig. 4. Magical trick task model

6 Evaluation

We have evaluated our assessment technology in two ways. First, we tested the domain reasoner on a large collection of free-form answers for the Magical trick task, which we describe in Sect. 6.1. Second, we conducted two small-scale pilot studies in 2018, and report about these pilots in Sect. 6.2.

6.1 Magical Trick Dataset

Between 2011 and 2015, several experiments were conducted on the LaboMep platform, which was developed by Sésamath, a French maths' teachers association.[4] The goal of these experiments was to study the integration of automated diagnosis tools in the usual teaching practices, and to study the evolution of students' cognitive profiles in algebra during several years [6]. This has resulted in a large collection of student responses (grade 8–10). From the dataset, we analysed 2956 free-form answers for the Magical trick task with the domain reasoner. Running the analysis took 155 s, which is fast enough for online assessment. From the 2956 answers, we extracted 18,302 mathematical expressions: 99.2% of these expressions could be parsed. Most of the parse errors are caused by unbalanced parentheses, for example `[(x+8)*3-4+x)]/4+2-x = 7`.

The results of recognising the solution strategy are as follows: 115 answers (3.89%) only contain natural language, and no mathematical content; 677 answers (22.90%) follow the algebraic strategy; 1527 answers (51.66%) follow the arithmetic strategy; 637 answers (21.55%) could not be recognised.

We also found combinations of solution strategies, e.g. the arithmetic strategy with different numbers, or the algebraic strategy followed by the arithmetic strategy. Some analysis results were checked ad hoc, but since we do not have a golden standard to compare against, we cannot rule out false positives or true negatives. Nevertheless, the results indicate that a substantial part of the free-form input could be analysed automatically, including the assessment of high-level learning goals based on the solution strategy that was followed.

6.2 Analysis of Pilots

In the context of the Erasmus+ Advise-Me project, we organized two small-scale pilots to test our assessment method and the free-form input for math story problems. The first pilot (N = 19) was organized on March 15, 2018, in Germany, and the second pilot (N = 22) was organized on April 9, 2018, in the Netherlands. Pupils were asked to solve ten math story problems, and to answer eight statements about the tasks, the software, and their attitude towards mathematics in a questionnaire. The experiments were carried out in 90 min: 10 min for logging-in, briefing, and the first questionnaire, 70 min for doing the tasks, and 10 min for the final survey. Some tasks have multiple parts (a–c). All interactions with the assessment environment were logged for further analysis.

From the questionnaires, we learned that pupils found the tasks clear and that they think they did well in the test. Doing well typically improves the experience. They found it relatively easy to use the assessment software and the text editing field. They do not often do math tasks on the computer.

Table 4 summarizes how often the algebraic or arithmetic solution strategy was recognised for the pilot studies. The 'graphical' strategy corresponds to

[4] http://www.labomep.net.

Table 4. Recognising solution strategies in pilots (Al = Algebraic, Ar = Arithmetic, Gr = Graphical, all as proportions); we also report the proportion of empty answers (Em) and unrecognised answers (Un).

Task		German pilot						Dutch pilot				
		N	Al	Ar	Gr	Em	Un	N	Al	Ar	Em	Un
1	Making a square	19	.53			.05	.42	22	.55			.45
2	Matryoshka	17		.76		.06	.18	12		.17	.33	.50
3	Car rental	18	.39		.39	.06	.17	22	.82			.18
4	Pattern	18	.11	.67			.22	19	.11	.53		.37
5	Magical trick	18	.06	.11		.56	.28	20			.70	.30
6a	Rectangle area	18	.94			.06		22	.95			.05
6b.	Rectangle area	18	.67			.06	.28	22	.77			.23
6c.	Rectangle area	18				.67	.33	22	.36		.55	.09
7b.	Theatre rate	18	.33		.11	.50	.06	21	.76		.05	.19
9a.	Area of triangle	15	.33			.47	.20	22	.77		.05	.18
9b.	Area of triangle	15	.20			.60	.20	22	.73		.05	.23
9c.	Area of triangle	15	.13			.73	.13	22	.77		.23	
10.	V-pattern	15	.53			.27	.20	22	.73		.05	.23
	Overall	222	.33	.12	.04	.30	.21	270	.59	.04	.14	.22

approximating the solution directly from a graph. Task 7a (Theatre rate) and task 8 (Area and expression) are omitted: the former requires an answer in natural language, and for the latter, pupils have to click areas instead of writing mathematical expressions. Overall, the solution strategy could be recognised for nearly 80% of the answers. For the remaining 20%, recognised steps and the final answer can still provide valuable information. A closer inspection of the unrecognised answers resulted in the following categorization of difficulties in recognising answers: unclear or ambiguous notation, use of natural language, and unanticipated errors.

7 Conclusion and Future Work

We have developed a framework for fine-grained cognitive assessment of free-form solutions to math story problems for pupils of around 15 years old. The framework uses a domain reasoner to analyse the input from pupils. The domain reasoner extracts the mathematics from the free-form input, parses the mathematical expressions, and then tries to recognise a solution strategy in the solution. The diagnosis from the domain reasoner is taken as input by a Bayesian task model, which in its turn is used to populate a user model.

We have evaluated our framework in various ways. We have tested that one of our domain reasoners for a particular task can analyse more than 80% of pupil

solutions in a dataset of almost 3000 solutions. In a number of small pilots we determined the main causes for our domain reasoner to fail to recognise pupil solutions. Some of the main causes are pupils using only natural language, or mixing up notation. In a qualitative evaluation, pupils were mildly positive about solving this kind of tasks in an online assessment system.

In the future, we want to analyse the quality of the user models resulting from our analyses. We want to determine ways to deal with situations in which we do not recognise a solution from a pupil. Here we envisage several approaches: we might ask a pupil simpler questions, or we might combine our analysis with natural language processing software for recognising mathematical language. Furthermore, we want to perform more extensive evaluations with our framework and compare the fine-grained cognitive assessments with a manual analysis produced by experts.

Acknowledgements. The Advise-Me project has received funding from the European Union's ERASMUS+ Programme, Strategic Partnerships for school education for the development of innovation, under grant agreement number 2016-1-NL01-KA201-023022. For more information, visit http://advise-me.ou.nl.

References

1. Almond, R.G., Mislevy, R.J., Steinberg, L.S., Yan, D., Williamson, D.M.: Bayesian Networks in Educational Assessment. Springer, New York (2015). https://doi.org/10.1007/978-1-4939-2125-6
2. Conati, C., Gertner, A., VanLehn, K.: Using Bayesian networks to manage uncertainty in student modeling. User Model. User Adapt. Interact. **12**(4), 371–417 (2002)
3. Conejo, R., Guzmán, E., Millán, E., Trella, M., Pérez-De-La-Cruz, J.L., Ríos, A.: SIETTE: a web-based tool for adaptive testing. Int. J. Artif. Intell. Educ. **14**(1), 29–61 (2004)
4. Dowker, A.: Children with Difficulties in Mathematics: What Works?. DfES Publications, London (2004)
5. Feng, M., Heffernan, N., Koedinger, K.: Addressing the assessment challenge with an online system that tutors as it assesses. User Model. User Adapt. Interact. **19**(3), 243–266 (2009)
6. Grugeon-Allys, B., Chenevotot-Quentin, F., Pilet, J., Prévit, D.: Online automated assessment and student learning: the *PEPITE* project in elementary algebra. In: Ball, L., Drijvers, P., Ladel, S., Siller, H.-S., Tabach, M., Vale, C. (eds.) Uses of Technology in Primary and Secondary Mathematics Education. ICME-13, pp. 245–266. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76575-4_13
7. Heeren, B., Jeuring, J.: Feedback services for stepwise exercises. Sci. Comput. Program. **88**, 110–129 (2014)
8. Murray, T.: An overview of intelligent tutoring system authoring tools: updated analysis of the state of the art. In: Murray, T., Blessing, S.B., Ainsworth, S. (eds.) Authoring Tools for Advanced Technology Learning Environments, pp. 491–544. Springer, Dordrecht (2003). https://doi.org/10.1007/978-94-017-0819-7_17
9. Narciss, S., et al.: Exploring feedback and student characteristics relevant for personalizing feedback strategies. Comput. Educ. **71**, 56–76 (2014)

10. Nwana, H.: Intelligent tutoring systems: an overview. AI Rev. **4**(4), 251–277 (1990)
11. Sosnovsky, S., Brusilovsky, P., Lee, D.H., Zadorozhny, V., Zhou, X.: Re-assessing the value of adaptive navigation support in e-Learning context. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 193–203. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70987-9_22
12. VanLehn, K.: The behavior of tutoring systems. J. AIED **16**(3), 227–265 (2006)
13. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educ. Psychol. **46**(4), 197–221 (2011)