Math-Bridge Deliverable D 6.2: Diagnostic Tools Improved and Integrated in Math-Bridge

Johan Jeuring, Bastiaan Heeren, George Goguadze

August 26, 2011

1 Introduction

MATH-BRIDGE offers remedial mathematics material to students all over Europe. The material is used in several pedagogical scenarios: it can be used to support a physical course, but it can also be used by students who want to remedy mathematics on their own.

Learning and practicing are tightly connected, and for that purpose, students solve exercises when learning mathematics. By solving exercises correctly, a student shows that (s)he masters a particular subject. MATH-BRIDGE maintains a student model based on the answers of a student to exercises.

Exercises appear in various formats: they may be multiple-choice exercises, exercises to which a student has to give a final answer, or exercises which can be solved stepwise, interactively. All of these exercises are supported by and integrated in the learning environment MATH-BRIDGE, see [1]. However, developing stepwise, interactive exercises in MATH-BRIDGE is quite a lot of work, and giving feedback other than correct or wrong on intermediate steps based on the actions of a student is practically impossible. MATH-BRIDGE uses both Computer Algebra Systems and external domain reasoners to support giving feedback to students when solving exercises. Computer Algebra Systems are mainly used to check correctness of final answers, and can be used to check correctness of intermediate steps of interactive exercises by comparing them with the final answer. However, they cannot be used to give hints at intermediate steps, they cannot recognize the application of common errors, and they cannot derive stepwise solutions for an exercise. For that purpose we use domain reasoners in MATH-BRIDGE. At the start of the MATH-BRIDGE project, a single domain reasoner, a domain reasoner for derivatives [6], was integrated in MATH-BRIDGE.

Since 2007, Johan Jeuring, Bastiaan Heeren, and Alex Gerdes from the Open Universiteit of the Netherlands have worked on developing domain reasoners for exercises, based on a concept called *rewrite strategies* [5]. A class of interactive exercises, such as solving quadratic equations, is specified by means of a rewrite strategy, which is used to automatically derive a stepwise solution for an exercise, to give hints at any point in a solution of an exercise, and to give feedback to students at intermediate steps when solving an exercise.

This deliverable 6.2 describes which domains have been further developed for inclusion in the MATH-BRIDGE-service, and how the domain reasoners have been integrated in MATH-BRIDGE.

2 Domain reasoners

In this section we briefly introduce domain reasoners, describe how we have selected the domains for which we have built domain reasoners, and we discuss some issues in the development of domain reasoners.

2.1 What is a domain reasoner?

A domain reasoner supports solving mathematical exercises interactively. A student can solve an exercise by making small steps until she reaches a solution, just as she would do when using pen-and-paper. She can ask for a hint at any step in the development of the exercise, and the domain reasoner then gives a hint about the best step to take at this point to arrive at a solution to the exercise. Figure 1 shows an example in which a student solves a linear equation, and the domain reasoners gives a hint about the next step to take.

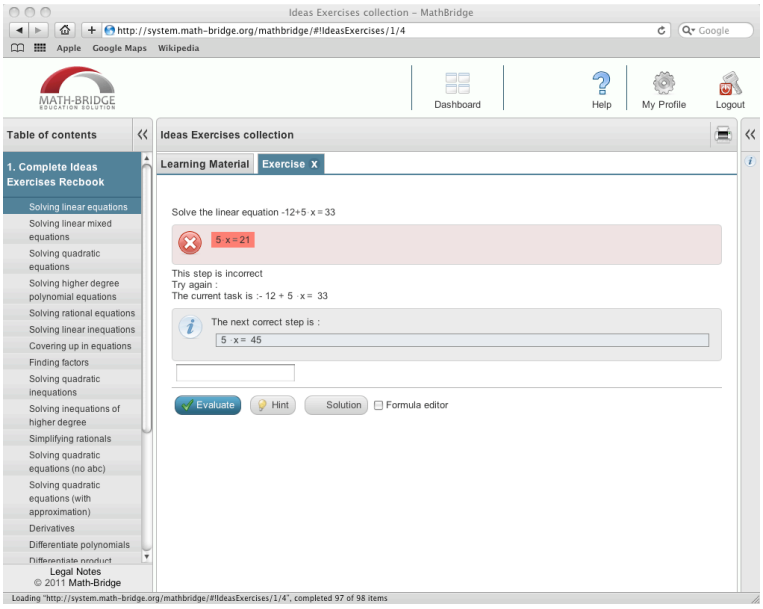

Figure 1: A hint for solving a linear equation

Furthermore, a domain reasoner can show a complete solution to an exercise. Figure 2 shows a worked-out solution produced by the domain reasoner for quadratic equations developed within the project. A student

can ask to get such a worked-out solution of an exercise.

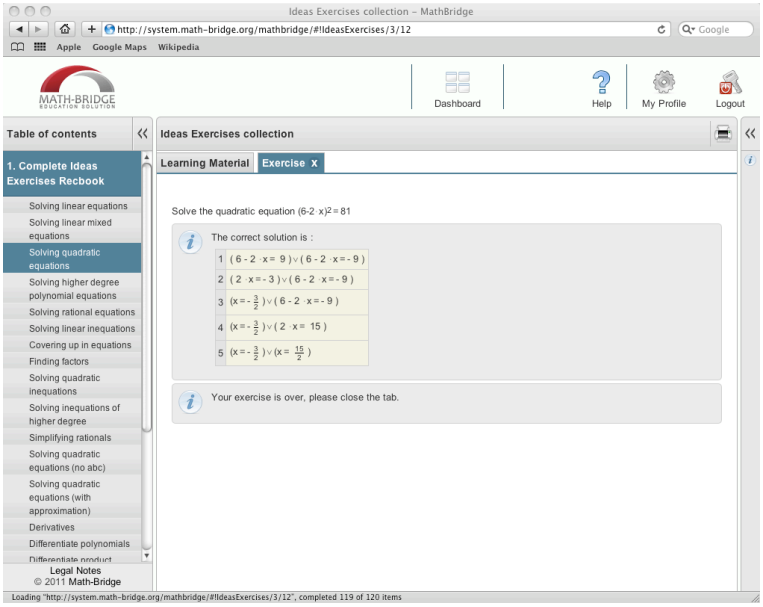

Figure 2: A derivation of a quadratic equation

Another service offered by our domain reasoners is giving feedback on user input. For example, if a student rewrites a linear equation $2 \cdot (x+3) = 5$ to $2 \cdot x + 3 = 5$, we report that we see an application of a common error in which multiplication is not correctly distributed over addition.

The domain reasoners offer functionality to determine worked-out solutions, hints (in various forms and level of detail), and feedback. Furthermore, they come with a generator that can generate an arbitrary exercise in the domain, and most domain reasoners come with a set of predefined exercises. The way the output of a domain reasoner, such as a hint or a worked-out solution, is presented to a student depends on the learning environment, Math-Bridge in our case. The domain reasoners are offered as web-services, and are used by multiple learning environments, among which Math-Bridge. They are licensed under the open-source GPL license.

2.2 Selecting domains

We have selected the domains for which to build domain reasoners in four ways.

- We have explained the concept of domain reasoners to the MATH-BRIDGE-partners by means of examples, documentation, and scientific articles, and asked them to send us domains for which they want to get help from domain reasoners;
- We have studied the kind of exercises that are offered in several mathematical learning environments, such as MATH-BRIDGE, MathDox, and the Freudenthal Mathematical Environment, and investigated which of these could be handled by means of a domain reasoner;
- We have studied the domains covered in the Dutch secondary school mathematics programme, and investigated which of these could be handled by means of a domain reasoner;
- We had already a number of domain reasoners available, which we also made available for MATH-BRIDGE.

Based on this input, we developed strategies for the following exercises:

1. Simplifying fractions
2. Calculating powers
3. Derivatives
4. Differentiate polynomials
5. Differentiate products
6. Differentiate quotients
7. Gaussian elimination
8. Gram Schmidt
9. Solving systems of linear equations using matrices
10. Finding factors
11. Simplifying powers
12. Writing with non-negative exponents
13. Writing as a power
14. Simplifying rationals

15. Solving equations
16. Solving linear equations
17. Solving linear equations with mixed fractions
18. Solving quadratic equations
19. Solving quadratic equations (no quadratic formula)
20. Solving quadratic equations (with approximation)
21. Solving higher degree polynomial equations
22. Solving exponential equations
23. Solving logarithmic equations
24. Solving power equations
25. Solving rational equations
26. Solving systems of linear equations
27. Solving linear inequations
28. Solving quadratic inequations
29. Solving inequations of higher degree

Using the ontology for mathematics used in the MATH-BRIDGE-project, we can roughly categorize these strategies in the following six domain reasoners:

- Numbers and computation: 1-2.
- Differentiation: 3-6.
- Linear algebra: 7-10.
- Algebraic manipulation: 11-15.
- Solving equations: 16-26.
- Solving inequations: 27-29.

What constitutes a domain reasoner is not precisely defined: if we take the lowest level in the ontology for mathematics, we would have to separate the domain reasoner for equations into seven domain reasoners for linear, quadratic, polynomial, rational, exponential, logarithmic, and systems of equations. If we take the lowest but one level in the ontology, we get domain reasoners for fractions, exponents, algebraic manipulation, equations, linear algebra, and single-variable calculus. Since our main effort is spent on developing functionality for solving classes of exercises, we see the list of 29 exercises given above as our domain reasoner contribution. Of these, the linear algebra exercises existed at the start of the project, the others have been developed within the MATH-BRIDGE-project. The number of predefined exercises developed for the project is 1306 at the moment.

2.3 Developing domain reasoners

The domain reasoners use the IDEAS framework developed by Heeren, Jeuring, and Gerdes over the last five years. The framework is built around the concept of rewrite strategies for exercises. It is developed in the functional programming language Haskell, and consists of around 20.000 lines of code.

Before the start of the MATH-BRIDGE project, domain reasoners for logic and linear algebra had already been developed at the Open Universiteit the Netherlands. The mathematical domains used in MATH-BRIDGE introduced some extra challenges to the framework, in particular related to normal forms of mathematical expressions. For example, if $x = 1\frac{4}{7}$ is the desired answer to an exercise, is $x = 11/7$ then a good answer too, or not? And are $\frac{\pi}{2}$ and 0.5π considered to be the same answer? To deal with such questions we developed so-called views, which we use to specify particular normal forms of mathematical expressions [2].

Another challenge we faced was developing technology for cross-domain reasoning. For example, one of the ways to solve a quadratic equation is to factorize the quadratic expression, and solve the two linear equations obtained thus. It follows that we need to use the strategy for linear equations in the strategy for quadratic equations. In principle, our strategy framework allows us to reuse strategies in other strategies, but doing so often imposed a rather strict order for performing exercise steps upon students. To make reusing strategies in other strategies such that students can switch between the several subcomponents, we introduced an interleaving component into our strategy framework [4].

Since we expect teachers to want to specify or change particular feedback for particular classes of exercises, we started on developing technology

for adapting domain reasoners. We have added constructs to our strategy language that can be used to collapse or expand strategies (for example: a student can solve a linear equation in a single step if she has done this often before, but might need feedback about individual steps if she is relatively new to the subject), to add rewrite rules to a strategy, etc. [3] After we perform the first experiments with our mathematical domain reasoners, we hope to offer the functionality to adapt domain reasoners to teachers. Such functionality needs to be developed in close cooperation with the learning environments that use our domain reasoners.

Developing a domain reasoner is still a substantial amount of work, but we believe that with the many domains that are available now, it is much easier to develop a new domain reasoner than it was at the start of the project.

3 Integration in Math-Bridge

The domain reasoners have been integrated in MATH-BRIDGE. Deliverable D6.1 describes the details of how MATH-BRIDGE integrates external tools. On the side of the domain reasoners, we have to ensure that the domain reasoners understand input and produce output in the omdoc format, and that they use the metadata as used in the MATH-BRIDGE project, such as ontology and difficulty for the student model, and title and problem statement in various languages for presenting exercises. The domain reasoners have been extended accordingly. Besides these details about exercise texts and metadata, we need to provide the feedback provided by the domain reasoners in various languages. This has been solved both at the MATH-BRIDGE side as well as the domain reasoner side, and we have yet to determine which of the two ways is most convenient to use.

References

[1] George Goguadze. *ActiveMath - Generation and Reuse of Interactive Exercises using Domain Reasoners and Automated Tutorial Strategies.* PhD thesis, Universität des Saarlandes, May 2011.

[2] B. Heeren and J. Jeuring. Canonical forms in interactive exercise assistants. In *MKM'09*, volume 5625 of *LNCS*, pages 325–340. Springer, 2009.

[3] B. Heeren and J. Jeuring. Adapting mathematical domain reasoners. In *Proceedings of the 9th MKM international conference*, MKM'10, pages 315–330. Springer, 2010.

[4] B. Heeren and J. Jeuring. Interleaving strategies. In *MKM'11*, volume 6824 of *LNAI*, pages 196–211. Springer, 2011.

[5] B. Heeren, J. Jeuring, and A. Gerdes. Specifying rewrite strategies for interactive exercises. *Mathematics in Computer Science*, 3(3):349–370, 2010.

[6] Claus Zinn. Supporting tutorial feedback to student help requests and errors in symbolic differentiation. In M. Ikeda, K. Ashley, and T.-W. Chan, editors, *ITS 2006*, volume 4053 of *LNCS*, pages 349–359. Springer, 2006.