# SYSTEM FOR AUTOMATIC GENERATION OF EXAMINATION PAPERS IN DISCRETE MATHEMATICS

Mikael Fridenfalk
*Uppsala University Campus Gotland*
*621 67 Visby, Sweden*

**ABSTRACT**

A system was developed for automatic generation of problems and solutions for examinations in a university distance course in discrete mathematics and tested in a pilot experiment involving 200 students. Considering the success of such systems in the past, particularly including automatic assessment, it should not take long before such systems are widely used in higher mathematics education. The goal of this paper is to encourage such development by sharing some of the experience that was gained along the way in the implementation and application of such system.

**KEYWORDS**

Automatic item generation, discrete mathematics, distance education, examination.

## 1. INTRODUCTION

Automatic generation and grading of examinations is a promising development that may potentially save time and effort for professors within the field of technology, giving room for further improvements in the education, along with allocation of additional time for discussions with the students. Such systems may also be used for the generation of individualized examinations, decreasing the risk for plagiarism while at the same time offering each student higher flexibility on examination dates, followed by decreased grading time and increased grading accuracy (Beevers, 2000; Maple TA, 2013; NCES, 2005; Pead, 2010; Rasila, 2010).

To contribute to this development, an automatic item generator called the Virtual Mathematics Assistant (VMA) was developed by the author and launched at Gotland University (recently merged with Uppsala University, Sweden), with the ability to generate full examination papers and solutions in a distance course in discrete mathematics for computer science students. This generator is presently able to create examination papers covering 2048 courses (with 69 assignments per course occasion) or a total amount of over 100,000 unique assignments within over ten different areas of discrete mathematics. The VMA system was applied during two consecutive years totally covering around 200 students. Compared to previous examinations, the new ones succeeded in addition to reduce the grading time for each student to about one-third compared to previous years without compromising the quality of the course.

## 2. MATERIALS AND METHODS

The VMA system was developed in C++ using Apple Xcode (2013) for the generation of LaTex-code and TeXShop (2013) for generation of the PDF examination documents. C++ is a common programming language for development of software applications. LaTex is the standard typesetting system for mathematical documents and is often used for generation of scientific papers.

The VMA system is designed to operate in two primary modes, development and production. In the development mode the programmer is able to focus on the implementation and validation of a single model or engine at a time and to make sure that each component in the system performs properly. In production mode, the user currently sets a course index between 0 and 2047 and compiles and runs the VMA application, whereby twelve examination papers are generated in LaTex-format plus twelve papers that

include both the problems and the solutions covering a single course in discrete mathematics. In addition a number of examination papers with solutions are generated for the students to study in advance of examinations.

The distance course consists presently of three examinations (with five assignments each) which each have to be completed within a single week. The course is concluded by a 24-hour final examination consisting of eight assignments, which determines the final grade of the course and includes at least two assignments from each examination category. Each examination (including the final) is issued three times per course. To encourage as many students as possible to aim for the top grade, the workload of the students is maximized during the start, thereby giving them sufficient time at the end of the course to perform studies at a more advanced level than is required to pass the initial examinations.

The basic idea behind the design of the VMA system was to break down common problems within the area of discrete mathematics into basic elements and then recombine such elements into a more complex final result (assignments). Each area showed however to be fundamentally different from the others and many problems were in relative terms resolved by *ad hoc* solutions. The VMA system consists currently of 16 assignment generation engines, see Table 1.

Table 1. A list of the VMA assignment generation engines

| Engine | Examination | Category | Models |
|---|---|---|---|
| Truth Table | 1 | Logic | 8 |
| Propositional Logic | 1 | Logic | 16 |
| Predicate Logic | 1 | Logic | 102 |
| Karnaugh Diagram | 1 | Logic | 2 |
| Venn Diagram | 1 | Set Theory | 2 |
| Set Equation | 1 | Set Theory | 12 |
| Radix | 2 | Numeral Systems | 6 |
| RSA Encryption | 2 | Modular Systems | 1 |
| Diophantine Equation | 2 | Modular Systems | 1 |
| Matrix Multiplication | 2 | Arrays | 2 |
| Proof by Induction | 2 | Induction | 2 |
| Minimum Spanning Tree | 3 | Graph Theory | 4 |
| Combinatorics - Level 1 | 3 | Combinatorics | 10 |
| Combinatorics - Level 2 | 3 | Combinatorics | 11 |
| Combinatorics - Level 3 | 3 | Combinatorics | 9 |
| Combinatorics - Level 4 | 3 | Combinatorics | 14 |

In its simplest form, an engine or a model could in theory consist of a number generator and an environmental context generator. The number generator is here defined as the generator of the calculations involved in solving a problem. The environmental context generator is the context in which a problem can occur, typically expressed by words. By the multiplication of the numerical set with the environmental, a matrix of possible assignments can be created. Although this is a straightforward idea it showed to be difficult to implement across the board due to the great differences between the types of problems that had to be formulated and solved. Of all 16 engines the simplest to implement was the matrix multiplication engine (that was included in the course to support students who had not previously studied linear algebra), which only showed to require two generation models. This assignment was also by far the simplest to solve for the students. In such assignment, two matrices are allocated and randomly filled with small integers, such that a predefined number of elements are non-zeros to regulate the difficulty of the assignment. Matrix size is made to vary in addition to the context in which the multiplications are performed.

The most difficult engine to develop was for predicate logic, which showed to require about a hundred models. This assignment type was the only one in the course that was transformed into a multiple-choice section (three questions per assignment with eight choices per question). Traditionally in this course the students were asked to express formal language as predicate logic. The backside for the teacher was that it could take in average 15 minutes to analyze the solution for each student, to grade it and write an appropriate reply. By making this a multiple choice assignment and instead focus the effort to provide for very detailed solutions for the students to study, the average grading and feedback time for this assignment was for each student reduced to below 30 seconds and feedback from the students on the received grade was nearly

eliminated. In addition, since the generation of combinatorics problems was difficult to automate, four engines were devised, each designed to generate assignments with a fixed level of difficulty.

One of the main challenges in designing a system like VMA showed to be balancing. The level of difficulty for each assignment is for instance recommended to reside within predefined limits, else the variation between the examinations could occasionally turn out to be out of proportion. Some natural variation is expected even in manual development of examinations, but in the case of automatic generation of assignments the predictability of the level of difficulty for each individual assignment is even more important for unsupervised generation. As an example the VMA Diophantine equation engine is able to generate potentially a large number of problems, but only selects those with moderately complex solutions, thereby eliminating equations with no solution or only simpler solutions and equations that require relatively extensive calculations. Such trial and error process was used in some engines to ensure that proper level of difficulty was found, but only if it was not possible to predict the complexity of the generated assignment in advance by any other means. Another question was the relative balancing between the difficulty levels of different assignments. A basic inter-assignment moderator was developed for compilation of examinations so that easier assignments would counter more difficult ones, thereby keeping the total level for an examination paper relatively constant.

A system was in addition implemented for engines incorporating the largest number of models to ensure that it would take a few course occasions before the same model was used twice. The main reason for this was that one of the balancing factors in addition was the distribution of assignments with respect to the levels of required creativity to solve a certain problem. The less creative assignments were in this work categorized as “procedural” (such as matrix multiplication) and are typically based on a few models and the more original as “creative” (such as the higher level combinatorics problems), based typically on multiple models. The overall division is currently considered to be relatively even between procedural and creative elements throughout the whole course. As a note, procedural problems are defined as the ones were the student has to carefully follow the instructions of an assignment and double-check its own calculations to maximize its score. The creative problems test the knowledge of the students on a deeper level by the introduction of problems that are usually not included in precise terms in the textbook or can be found on the Internet, but that easily are solved if the essence of the subject is thoroughly understood. As a rule the generation of procedural problems showed to be the most straightforward to automate, while the creative problems showed to be relatively difficult to automate in the sense that multiple models were required and that it typically took at least five times as much time to develop an engine for a creative assignment.

The addition of uncommon (but plausible) constraints to conventional assignments showed in addition to be a useful tool in the design of original assignments. Not only did it make grading in some occasions significantly easier in procedural assignments by making all solutions to look more or less alike (and almost identical to the generated solution provided by the VMA), but it also excluded the possibility that the student could have used an automatic generator on the Internet to solve the problem without a deeper understanding of the subject.

## 3. RESULTS

Although automatic item generation could be considered to be relatively difficult to implement within discrete mathematics, it proved to be possible for a single programmer spending one time unit in the development of this system to save potentially 200 time units for a traditional mathematics educator. No decrease was further detected in the quality of the generated assignments and solutions compared to manually developed ones during previous years in the same course. This assumption is partly based on student evaluations, which consisted of 50 replies out of 200 participating students. One positive side-effect of the application of this new system was that since it was developed with a possible future automatic grading system in mind, the manual grading process was significantly simplified as well. This was appreciated by the students since grades could be received typically within 24 hours after examination deadline. The downside was that the procedural elements in the course could only be solved in a very specific fashion to ensure rapid grading, and since all students received identical assignments, this made it difficult to detect plagiarism. The creative assignments did not suffer from this flaw since they could be solved in multiple ways, but made the grading process more challenging.

## 4. CONCLUSION AND FUTURE WORK

Considering how well this system proved to work in practice (as expected from previous projects similar to this), there is really no reason why such systems should not in the future become a natural integral part of mathematics-based education across the board. Once there are sufficiently many automatic item generators to choose between for a professor so that it takes shorter time for the students to actually learn the subject in question compared with using similar systems to automatically find a solution, this could significantly reduce the required time for the generation of examinations. Since the lifespan of VMA is likely to be considerably shorter than 2048 course occasions, such system could in practice be used to provide each student with an individual assignment, increasing the flexibility by which examinations are issued and at the same time decreasing the risk for plagiarism in both campus and distance courses. While most VMA engines are at this stage considered to be complete, for sustained quality over time, the recommended number of generation models for the combinatorics engines is at a minimum 24 models per engine or preferably the double. In addition, the engine for proof by induction is recommended to be extended by one extra model.

## ACKNOWLEDGEMENT

Many thanks to Dr. Bo Göran Johansson for the main design of the course structure as defined in Table 1, which was of great help during the design of the VMA system.

## REFERENCES

Apple Xcode, Apple Inc., 10 February 2013 <https://developer.apple.com/xcode>.

Beevers, C. & Jackson, D. H., 2000. Automatic Assessment - Mathwise and Beyond. In *Proc Twelfth International Conference on Technology and Collegiate Mathematics*, San Francisco, 1999, 1/01/70, pp. 19-23.

Maple TA, Waterloo Maple Inc., 20 January 2013 <http://maplesoft.com/products/mapleta>.

NCES, 2005. *Online Assessment in Mathematics and Writing: Reports From the NAEP Technology-Based Assessment Project, Research and Development Series*. National Center for Education Statistics, Institute of Education Sciences, U.S. Department of Education, 1990 K Street NW, Washington, DC 20006-5651.

Pead, D., 2010. *On Computer-Based Assessment of Mathematics.* PhD Thesis, University of Nottingham.

Rasila, A. et al, 2010. Automatic Assessment in Engineering Mathematics: Evaluation of the Impact. In *ReflekTori 2010 Symposium of Engineering Education,* Aalto University School of Science and Technology, pp. 37-45.

TeXShop, 24 January 2013 <http://pages.uoregon.edu/koch/texshop>.