Enhancing Modularity and Feedback in Computer Aided Assessment

Duarte P., Nunes I., Neto J.P., and Chambel T.
Faculty of Sciences, Lisbon University
pduarte@ptmat.fc.ul.pt , in@di.fc.ul.pt, jpn@di.fc.ul.pt, tc@di.fc.ul.pt

Abstract

We present a model of a computer aided assessment system — CATS (Computer Assessable Task System) — for mathematics, which is both modular and has rich feedback. Modular in the sense that elaborated tasks need not be built from scratch but, instead, from already existing tasks. Modules are made general enough to be reusable in many different contexts. As a consequence, the construction and management of multiple-step problems with complex system-user interaction, is facilitated for non-expert users. By rich feedback we mean that the system automatically generates messages explaining errors, whenever students make algebraic manipulation and logical mistakes. This contrasts with current assessment software where feedback must be hard coded by the teacher, therefore constraining feedback richness by the teacher programming skills.

1. Introduction

Computer Aided Assessment (CAA) is a broad area covering both teacher creation of interactive exercises, and electronic assessment of students in the form of, e.g., homework, automatic graded tests or just a set of questions for self-assessment. This area involves several different concerns as, e.g., pedagogical, usability, or extensibility ones.

There are already many computer systems that assist and assess students through the resolution of mathematical exercises, going from simple true/false, multiple choice or fill-in questions to more elaborated ones where students are asked to give step by step answers to many different sorts of problems. Most of the systems we know of, ask users to specify new problems in such a way that problem reuse is not accounted for — e.g., teachers hardcode, using some high-level editor tool, all the student/system interactivity. Besides compromising reuse, this approach also limits feedback richness since it is the teacher who writes the problem, that gets responsible for specifying feedback.

We present a model of an interactive computer assessment system that allows, on the one hand, students to solve elaborated mathematical problems, while being given feedback on their mistakes, and on the other hand, teachers to easily build new complex interactive problems from a pool of pre-defined modules. The non-trivial part of feedback in our approach is not defined directly by the teacher, but is partially built into basic tasks available for composition.

In section 2.1 we present the main concepts that comprise our model. Section 2.2 describes the model we envisaged for knowledge representation. Section 2.3 describes the intended problem resolution process. The main steps of the task construction process are described in section 2.4. Examples of generic tasks to be used by teachers when creating their own tasks are given in Section 2.5. Related work is discussed in section 3. Finally section 4 concludes and identifies directions for future work.

2. The CATS Model

We present a model of a CAA system which benefits from the following properties:

- Supplying rich feedback information to students;
- Being modular in the sense that the exercises – tasks – that are built can be reused and recombined to create other tasks.

Although feedback is "the most powerful single moderator that enhances achievement" (c.f. Hattie03]), nevertheless, and to the extent of our knowledge, it is poorly supported by almost CAA systems (including those that are based in some Computer Algebra System). We do not consider mere "right/wrong answer" information, or teacher pre-defined, general messages to be a rich feedback. We aim at giving the student clues about incorrect rule application, or application of invalid rules, possibly providing for counter-examples, etc.

The second property aims at an extensible system where the effort spent in building some mathematical task can be capitalized by using that task to build some other, more complex, task. Teachers will be able to create complex exercises that would otherwise be very difficult to define.

2.1. Conceptual Model

The model we devised is based on the following main concepts:

Tasks are the building blocks of our approach. They represent mathematical challenges that students must overcome. A task is more than a simple mathematical exercise — it proposes a problem to the student, and then guides, assesses, and provides feedback, throughout the resolution process. In general, a task is a generic module, depending on parameters, which abstracts some class of problems. Basic tasks (e.g., expression manipulation, equation solving, logic reasoning) are initially created by experts.

Subtasks of a given main task are the tasks the student may choose to perform during the resolution process of the main task.

By default, a task has no subtasks. In this case, the student may have access to all existing tasks sharing the same Context. This gives the student a great freedom to choose the resolution path he likes. In other words, no guidance is provided by the teacher. By enumerating the subtasks of a task, the teacher narrows the set of possible resolution paths, thus helping the student. The teacher can, furthermore, define a precedence graph among subtasks, which amounts to even more guidance through the task resolution process.

By combining existing subtasks, the teacher can build complex multiple step problems, which automatically benefit from their components rich feedback. The teacher may create links among the subtasks, forcing the output of one subtask to be the input of another.

Contexts define a grammar specifying the language representing the knowledge under evaluation, and a meta-language, consisting of a set of meta-functions, for teacher usage.

Meta-Functions are functions designed to evaluate student answers, providing him/her with error explanations.

Oracles are dynamical objects, created for evaluation of student answers. Oracles are associated with Contexts, and are able to provide appropriate feedback to the student's resolution.

Resolution is the sequence of intermediate answers given by the student.

Solution Set comprises the answer type plus a condition to be fulfilled by the answer (the *Success* meta-condition).

Answer is a mathematical term, input by the student, which the system verifies to be in the solution set.

2.2. Knowledge Representation

Each task has an associated *context* which defines the *object-language* and the *meta-language* needed to specify the problem. The object-language is the mathematical speech language. The meta-language consists of the meta-functions used to evaluate student's answers, providing him/her with feedback on errors.

When solving a task, students use the object-language. When creating a task, teachers use meta-functions in order to define task solution conditions and partial progress evaluation.

Meta-functions are the core of our system feedback. Some basic meta-functions evaluate, for instance, whether two algebraic expressions, or two equations, are equivalent, or whether a proposition follows from a given set of premises. These meta-functions identify common errors, like incorrect rule application, application of invalid rules, and small misspellings.

Error messages explain these errors to students, possibly providing counter-examples. Such basic meta-functions may depend on complex algorithms, therefore some of them are provided as the result of expert work. Teachers can easily create their own meta-functions, by recombining and customizing messages of already existing meta-functions, thus creating a higher level of feedback messages.

Each context has an associated *oracle* which is capable of evaluating the context meta-functions and, therefore, the teacher meta-conditions on student answers.

If teachers want to add new meta-functions, e.g., in order to define some specific feedback, they may extend the corresponding context. Contexts are extended through object-language and/or meta-language extension.

All mathematical content is represented as terms, which, in our system, are typified. Term types are defined grammatically within contexts. Some of these types have a random attribute, which means that random values of those types can be generated. Thus, parametric tasks depending on random parameters allow the automatic generation of random instances.

2.3. Task Resolution Process

Solving a task may involve several steps. Each step corresponds to the resolution of a subtask, which itself is a task. At any stage of resolution, the next step may either be predefined in the task, or else selected by the student, among available tasks.

Two additional important concepts are:

Resolution Sheet: contains the sequence of intermediate accepted answers given by the student. More precisely, the resolution sheet is a tree formed by the resolution sheets of all undertaken and completed subtasks.

Draft Sheet: is used to write and draw marginal notes, or to perform auxiliary calculations. The set of available input tools is determined by the task Context. All objects written by the student in the draft sheet are terms of that Context.

During the task resolution process, the system: *(i)* manages the tree of all subtasks under execution; *(ii)* maintains coherence among states of all involved tasks; *(iii)* keeps track of the resolution sheet; *(iv)* provides a draft sheet (common to all subtasks).

2.4. Task Construction

A teacher builds a new task specifying the following items:

- the Context in which the problem is defined; it must be chosen from a pool of available Contexts;
- a set of typed parameters representing terms (optional);
- a statement that describes the problem and issues the challenge; this can be a parametric statement depending on the above parameters — from it, several instances of the problem are generated;
- the problem assumptions as terms of the associated Context (optional);
- the solution set, that is, the answer type, and the *Success* meta-condition (the condition that any answer must fulfill);

Most teachers will create tasks within already existing contexts supplied by experts, although any teacher with some programming skills will be able to make extensions of available contexts.

2.5. Task Examples

Any contextualized problem that needs to be interpreted, equated, and then solved, can be modeled as a task, which makes use of generic subtasks such as:

- stating relations between problem variables;
- solving equations, and systems of equations;
- simplifying algebraic expressions;
- symbolic computing (e.g., computing derivatives, anti-derivatives, and limits);
- performing short deductions and proofs;
- filling-in tables with function values. This is a very general task that can, for instance, be specialized to filling-in a table with the first and second derivatives' signs of some function, or else filling-in a table with the symbolic description of monotonicity and concavities of a given function in a given interval partition;
- draw the graph of a function. The student uses a graphical interface to sketch the graph of a one variable function;
- make some geometric graphical construction.

Tasks listed above are constructed by experts to be used by teachers. Combining them, the teacher can build complex multiple step problems, which automatically benefit from their components rich feedback.

3. Related Work

In order to clearly relate the CATS system with the present state of knowledge in this area, we restrict this analysis to mathematical CAA systems, and stress two important issues – feedback to student's answers, and problem creation – that we identified as being the main issues for which the CATS system contributes.

The kind of questions CAA systems support are commonly divided into closed and open ones, depending on whether corresponding answers belong to a pre-defined solution set, or not. The former class includes Multiple Choice, True/False, Multiple Answer, Ordering, and Matching questions. The latter includes Fill in the Blanks, and other free answering mathematical exercises as, e.g., interactive equation solving.

Although CATS aims at supporting both closed and open questions, we will focus on the state of the art of open ones, because most open research issues fall in this category. The relevant limitations we identified on some existing applications are due to the creation of open questions on the one hand, and their answer interpretation, verification, and feedback on the other.

From a pedagogical point of view, the intuitive idea that feedback is of extreme importance in problem solving has strong support from several studies (see, for example, [3] where it is concluded that "feedback is the most powerful single moderator that enhances achievement"). Therefore, whenever a student is solving a problem in a learning mode, rich feedback concerning a specific error on the student's answer as, e.g., incorrect rule application, application of invalid

rules, possibly providing for counter-examples, or directing the student to a piece of learning material, are important features for every CAA system.

In open questions solving, students may type any symbolic mathematical expression that must be evaluated against the teacher's solution specification.

Some CAA systems are *expression-oriented* in the sense that teachers specify a solution expression which will be checked against the student's answer through some kind of equivalence test. For example, in [2], three kinds of equality comparison are considered: *syntactic equality* checking whether two expressions are syntactically equal; *numerical equality* checking that two numbers are equal, and *semantical equality* checking whether two symbolic expressions are equivalent, that is, that they represent the same mathematical object.

Other CAA systems are *property-oriented* in the sense that teachers define an expression specifying the properties the student's answer must satisfy. This approach is more general than the expression-oriented one because equivalence testing can be specified through some adequate property statement.

Many existing CAA systems use a Computer Algebra System (CAS) to evaluate student's answers: Maple TA [6] and Aim [1] use Maple [5], Stack [7] uses Maxima [8] or Axiom [9], LeActiveMath [10] and MathDox [11] use a variety of CAS and theorem provers, Wallis [12] uses CAS supporting MathML or OpenMath like Maple or Yacas [13].

Although computer algebra systems are invaluable in what function evaluation is concerned, their functions do not provide feedback. Therefore, in CAS-based CAAs, feedback messages have to be hard-coded by the teacher in the process of creating an exercise. This feedback is typically given at the time the student inputs her final answer to the problem in hands. This contrasts with, for example, step-by-step feedback given by the non CAS-based Interactive Equation Solver described in [4].

CATS is not based on any computer algebra system to evaluate student's answers; it will rather provide an initial set of meta-functions on algebraic manipulation and logic reasoning that will be extensively reused through all mathematical contexts. Besides evaluating student's answers, these meta-functions also identify common errors, like incorrect rule application, application of invalid rules, and small misspellings, possibly providing counter-examples. From these evaluation results, the system can automatically generate rich and useful feedback messages. Other meta-functions can be created by teachers, by recombining and customizing messages of already existing meta-functions.

Exercise reusability is supported by existing CAAs in the sense that teachers may reuse already built exercises, stored in some question bank, in order to create exams, or some other exercise collections for student practice and drill. We aim at a broader meaning of reusability: that teachers can assemble complex contextualized problems from available tasks. To the best of our knowledge, existing CAAs do not support this kind of reusability.

4. Conclusions and Further Work

Concepts such as task, sub-task, context, meta-function are key to CATS' model meeting the goal we have stated above: enriching feedback and simplifying task creation through the composition of task modules.

Strong modularity is at the basis of CATS simple architecture and extensibility. We hope, in a near future, to exploit these properties by adding new features such as automatic grading, student knowledge profiling, and adaptive assessment.

Teachers using CATS contexts and tools for task construction will be able to provide specific libraries of problems to guide their students and, eventually, to share their libraries with a community of users.

5. References

[1] Aim, http://maths.york.ac.uk/moodle/aiminfo/ .

[2] G.Goguadze, A.G.Palomo, E.Melis, "Interactivity of Exercises in ActiveMath", in Proceedings of International Conference on Computers in Education (ICCE05), December 2005, Singapore..

[3] John Hattie, "Teachers Make a Difference: What is the research evidence?", in Australian Council for Educational Research Annual Conference on: Building Teacher Quality, University of Auckland, October 2003

[4] Harrie Passier and Johan Jeuring. "Feedback in an interactive equation solver". In Proceedings of the Web Advanced Learning Conference and Exhibition, WebALT 2006.

[5]. Maple, http://www.maplesoft.com/ .

[6]. Maple TA, http://www.maplesoft.com/ .

[7]. Stack, http://eee595.bham.ac.uk/~stack/ .

[8]. Maxima, http://www.maxima.sourceforge.org/ .

[9]. Axiom, http://page.axiom-developer.org/.

[10]. LeActiveMath, http://www.leactivemath.org/ .

[11]. MathDox, http://www.riaca.win.tue.nl/ .

[12]. Wallis, http://www.maths.ed.ac.uk/~wallis/ .

[13]. Yacas http://www.xs4all.nl/~apinkus/yacas.html