Authoring Interactive Exercises in ActiveMath

Giorgi Goguadze [a], Jan Tsigler [b],
[a] *University of Saarland, Saarbrücken, Germany, george@activemath.org*
[b] *German Research Institute for Artificial Intelligence (DFKI) Saarbrücken, Germany, tsigler@activemath.org*

Abstract.
This paper describes an authoring environment for exercises in a web-based learning environment ActiveMath. This tool offers WYSIWYG authoring of multi-step interactive exercises, represented as finite state machines, in which the authors can visually manipulate the graph of an exercise, edit the states and transitions of an exercise and run the edited exercise in each step of its creation. It combines features similar to those of systems such as Maple T.A.^{TM} for creating tests and assignments and Pseudo Tutors created with CTAT - Cognitive Tutors Authoring Tools.

The user interface for editing steps of an exercise with interactive elements is comparable to the one in Maple T.A.^{TM}, also semantic evaluation of the learners answer using a Computer Algebra System (CAS) is possible. Among other features, similar to Maple T.A.^{TM} is a possibility to create parameterized exercises using randomized variables in the exercise solution space.

On the other hand, visualizing the exercise graph and authoring different solution paths by modeling possible learner's answers (correct or incorrect) is similar to the behaviour recorder of CTAT when authoring Pseudo Tutors. Also the highly semantic knowledge representation of an exercise allows for further semantic analysis of the log data after the learner's interaction with an exercise.

1. Introduction

Interactive exercise sub-system is one of the major components of ActiveMath – a web-based interactive learning environment for mathematics. ActiveMath system incorporates several components, tools and external services that make use of underlying semantic knowledge representation of content and form a user-adaptive platform for technology-enhanced learning (see [7]).

The role of interactive exercises is crucial, since the results of interactivity of the system with a learner within an exercise are the main source of information for learner model component that is responsible for assessing the current mastery of the learner w.r.t. the domain concepts. This information about the mastery is then the basis for further adaptive actions of the ActiveMath system, e.g. presenting different additional instructional items and finally suggesting new exercises. This closes the global feedback loop in the system. For more details about the global feedback in ActiveMath see [8].

On the other hand, the exercises provide a large room for adaptive scaffolding on a micro level via usage of different tutorial strategies that define the way how an exercise should be rendered. For example, the strategy can decide whether or not to give the learner a second chance to answer the question in the task in case of a wrong answer, when to provide feedback and which type of feedback is currently the most appropriate, and so on. For more information about tutorial strategies for ActiveMath exercises see [6].

The architecture of ActiveMath exercise sub-system allows for manually authored, as well as fully or partially generated exercises. It also allows for automatically enriching manually authored exercises by applying different tutorial strategies to them [2].

In general, a multi-step solution of a mathematical problem can have one or more correct solution paths. When solving a problem the learner can also follow erroneous paths. The set of all correct and erroneous paths can be viewed as a directed graph. An interactive exercise can be seen as an automaton in which the learner is moving step by step on this graph and each next node he is visiting depends on the answer he provides for the tasks in the previous nodes. The nodes of the graph can be annotated by the teacher with additional feedback, providing hints, help on errors etc. By definition, a mathematical model of a system with discrete inputs and outputs and finite number of states is a finite state machine (see e.g. [3]). In our case nodes of the graph are the states, input is the answer of the learner and the output is the presentation of the feedback and the task in the next state to the learner.

Therefore, an exercise is represented as a finite state machine of states representing tutorial actions of the system and transitions between those states representing reactions upon learner's answers to the tasks in the exercise states. Exercise states and transitions have themselves complex internal structure.

Each exercise state consists of a task to be performed by a learner, possible feedback to the actions of the learner in a previous exercise state and interactive elements such as fill-in-blanks, multiple-choice-questions, orderings or mappings that will be used by the learner in order to provide the answer to the task. In each of the textual parts of the exercise state arbitrary text with formulas, images and other multimedia and formatting elements is allowed.

Tasks and feedbacks are also annotated with domain- and educational metadata which are necessary for updating the learner model and tutorial strategies to be applied to the exercise.

Each transition issued in a state of an exercise consists of a condition to be satisfied by the learner's answer in order for this transition to fire. It also contains diagnosis information that is necessary for updating the learner model of ActiveMath with the information about learners mastery changes. Finally, it contains a pointer to the next state to which the system will change if this transition fires.

Example. Consider a simple exercise, in which the learner has to differentiate the function $f(x) = 2 \cdot x$. The solution space of this exercise is shown in the Figure 1. The first state (`state1`) of our finite state machine defines a task to

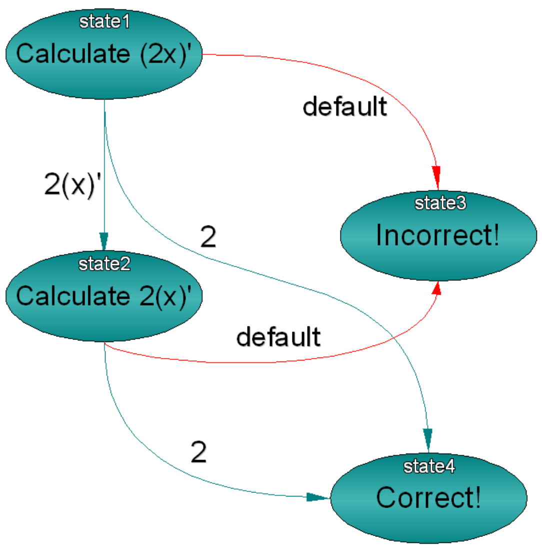

Figure 1. A simple exercise graph

calculate $(2x)'$. If the learner's answer is equal to $2 \cdot (x)'$ the next state will be `state2`, in case the learner entered the final correct answer 2 he is forwarded directly to the state `state4`. In all other cases, the default transition is triggered and the learner is forwarded to the `state3`. The states that do not define any transitions are final. As soon as the final state is reached the exercise terminates. In our example `state3` and `state4` are final.

The authoring tool we present in this paper offers visual authoring for exercise states and transitions, visualizing exercise as whole in form of a graph, as well as providing WYSIWYG editing of micro-structure of states and transitions and the direct testing of intermediate results of authoring via running the authored parts of the exercise in an exercise player.

Since the authoring tool is embedded into the ActiveMath system it has direct access to the exercise sub-system and other components of ActiveMath such as knowledge base, search tool, formula editor and the presentation system of ActiveMath, which are effectively reused in the tool.

2. Main Interface

Similarly to the behaviour recorder of CTAT when authoring Pseudo-Tutors (see [4]), the states of exercises and transitions between them are represented in form of a graph. This graph shows the paths the author defines in a solution space of an exercise. Simultaneously to the graph representation, the authors can edit individual states and transitions in an almost WYSIWYG structure editor. It is

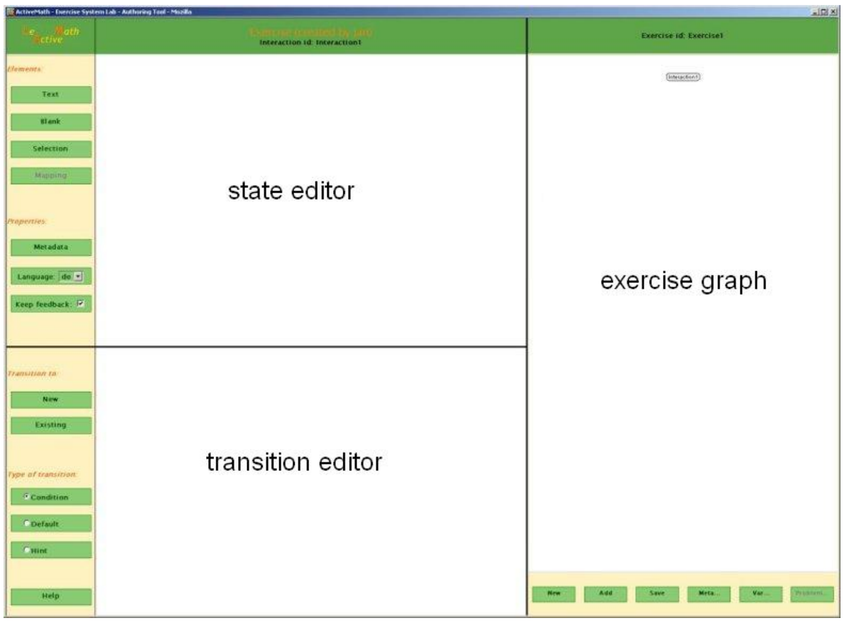

Figure 2. Exercise authoring interface

almost WYSIWYG in sense that additionally to the presentation of the content the markers for its micro structure are also visible, just like in a traditional Mozilla Composer for HTML pages.

The window of the authoring tool consists of 3 main frames, shown in the Figure 2. On the right side is the visualization of the exercise structure in form of a graph, the vertexes of which are representing the states on an exercise and the edges represent the transitions between those states. The upper frame on the left side is for editing the states of the exercise and the lower frame on the left side is for defining transitions. The tool also provides pop-up interfaces for authoring metadata of the relevant parts of the exercises using cascading drop-down menus.

3. Visualizing Exercise Graph

The sample graph of the exercise nodes and transitions between them is shown in the Figure 3. By right-clicking at on the node of the graph the context menu is called in which the author can select one of the options - edit the state, running the exercise starting from this state, or deleting the state. By deleting the state all the transitions to and from this state are deleted as well. When selecting the edit option, the exercise state is loaded into the state editor window and the transitions issued at this state are loaded into the transition editor window.

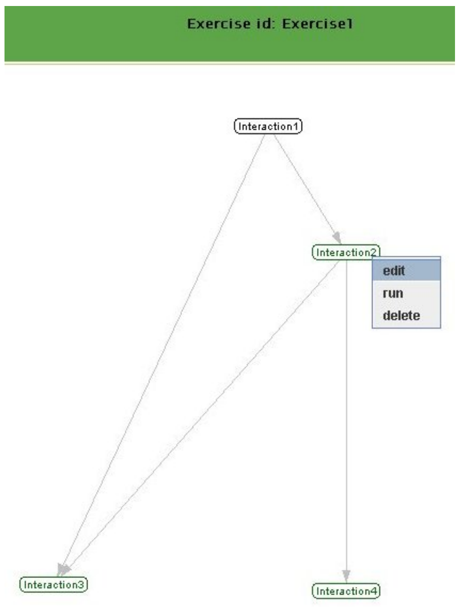

Figure 3. Sample exercise graph

4. Editing Exercise States

In the state editor the author can enter text with formulas, insert images, tables and other usual formatting elements. For doing so, the author calls a pop-up microstructure editor window by clicking on 'text' button in the side-bar. The microstructure editor window is shown in the Figure 4. The author can enter text in several languages by switching to the corresponding tab in the microstructure editor. The formulas are edited in the separate formula editor field on the top left, reusing the formula editor component of ActiveMath. The created formulas can be then inserted into the text.

For inserting images, tables and other media and formatting elements the author clicks on the corresponding icons in the toolbar of the microstructure editor.

In order to author interactive elements the microstructure editor is reused in different ways. For authoring blanks, only the formula editor is used with the special blank symbol in the palette. In case of a simple blank, only the blank symbol is inserted from the palette. The full power of a formula editor is needed in cases dealing with blanks inside complex formulas.

For authoring multiple choice selections, the microstructure editor is used as well enriched with an additional button in the toolbar, separating the already authored choices from new ones. Sample authoring of blanks and selections is shown in the Figure 5

The interface of microstructure editor is similar to the corresponding part of the authoring tool in Maple T.A.$^{\text{TM}}$. The formula editor in ActiveMath looks similar to the traditional Equation editor in Microsoft Word. The difference is, however, that the underlying generic semantic representation for mathematical

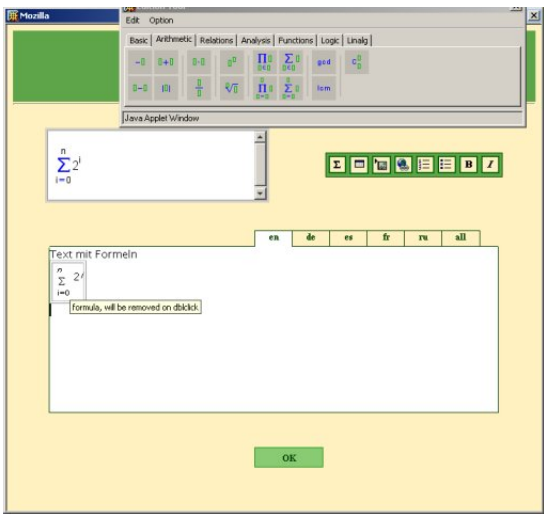
Mozilla
Edit Option
Basic | Arithmetic | Relations | Analysis | Functions | Logic | Linalg
Java Applet Window
en de es fr ru all
Text mit Formeln
formula, will be removed on dblclick
OK

Figure 4. Microstructure editor window

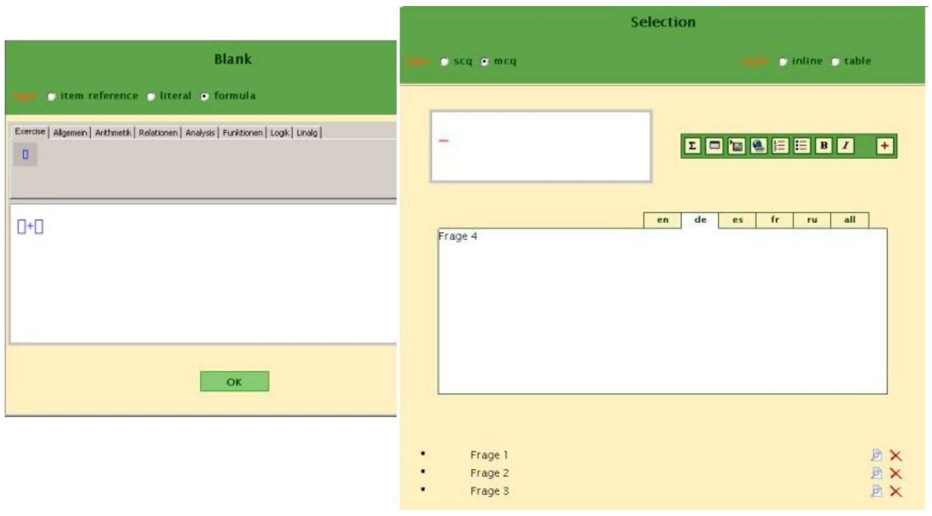
Blank
item reference literal formula
Exercise | Allgemein | Arithmetik | Relationen | Analysis | Funktionen | Logik | Linalg
OK
Selection
scq mcq
inline table
en de es fr ru all
Frage 4
Frage 1
Frage 2
Frage 3

Figure 5. Authoring blanks and selections

formulas – OpenMath [9] makes it possible to evaluate these expressions with external mathematical tools such as computer algebra systems. Underlying representation of formulas produced in Maple T.A.^{TM} editor is using the syntax Maple^{TM}. In contrast, expressions encoded in OpenMath can be evaluated with several different computer algebra and other mathematical reasoning systems, providing these systems are available for usage and the mapping from and to the OpenMath syntax (phrase book) is provided. Within EU project LeActive-Math, phrase books and connections to several Computer Algebra Systems were implemented (see [1]).

5. Defining Transitions

Transitions are defined in the transition editor in which the author specifies the type of a transition, the corresponding diagnosis, and a condition, in which the learner's answer is compared in a certain context to the control expression entered by the author.

The diagnosis information to be authored contains the achievement of the learner w.r.t. to the task in a current exercise state in case the transition fires, relevance of the learner's answer w.r.t. the completion of the task and the relations to the diagnosed misconceptions in case the condition in the transition is matching a particular incorrect answer.

Defining a condition for a transition is not entirely similar to the authoring-by-doing procedure in CTAT behaviour recorder, since the author needs to do more then just simulate the learner's answer. The condition has to be annotated with the context of the evaluation, e.g. whether we expect syntactic (or literal) comparison, or maybe numeric or semantic evaluation using CAS. For more information about the different evaluation contexts in ActiveMath exercises, see [1].

Semantic evaluation of the learner's input using CAS is, naturally, the most powerful feature of Maple T.A.^{TM}. The full power of Maple^{TM} is at author's service and any CAS command can be applied to the learner's answer. In contrast to Maple T.A.^{TM}, we abstract from CAS-dependent commands and perform purely mathematical operations upon the learner's answer, the semantics of which is clear enough to be understood by several different CAS or other mathematical reasoning systems.

Apart from usual conditions in which the expected answer of the learner is a string, mathematical expression, or a result of selecting choices in multiple choice questions, the author can define conditions matching other actions of a learner such as hint requests, giving up on the exercises and so on. Also the default transition, which fires if none of the conditional transitions match can be defined. Authoring sample transitions is shown in the Figure 6.

6. Authoring Parameterized Exercises

Similarly to Maple T.A.^{TM}, the exercise authoring tool provides a possibility to author exercises in which the exercise task and solution space are randomized.

Using the pop-up interface shown in the Figure 7 the authors can define custom variables together with the sets of possible values for them and use these variables in the authored exercise. Currently, the possible values are always integers. In order to add a new possible value the author clicks the red '+' button on the right side which appeares for each newly defined variable. In order to delete existing values the blue '−' button has to be used. This interface has to be extended in the near future, since the exercise system already has support for more complex sets of possible values.

As soon as the variables are defined they appear as special symbols in the pallette of the formula editor and can be used within formulas in all relevant parts of the exercise tasks, feedbacks and conditions.

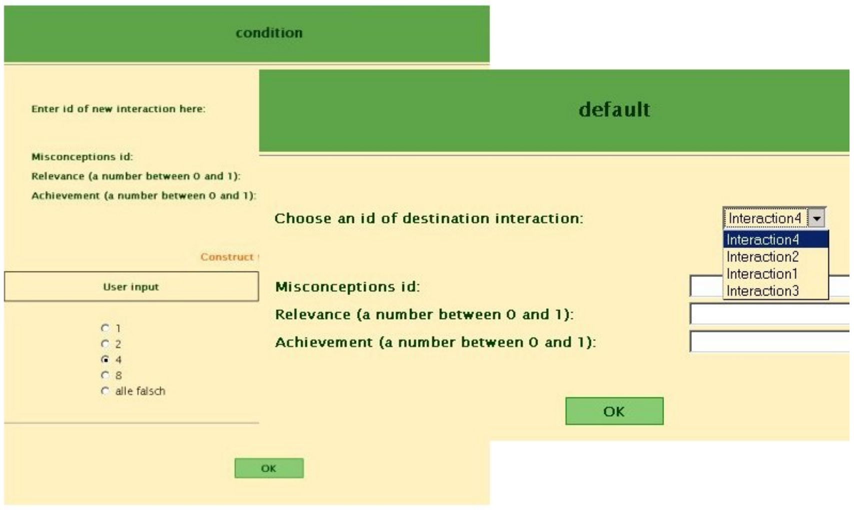
condition
Enter id of new interaction here:
Misconceptions id:
Relevance (a number between 0 and 1):
Achievement (a number between 0 and 1):
Construct
User input
1
2
4
8
alle falsch
OK
default
Choose an id of destination interaction:
Interaction4
Interaction4
Interaction2
Interaction1
Interaction3
Misconceptions id:
Relevance (a number between 0 and 1):
Achievement (a number between 0 and 1):
OK

Figure 6. Defining transitions

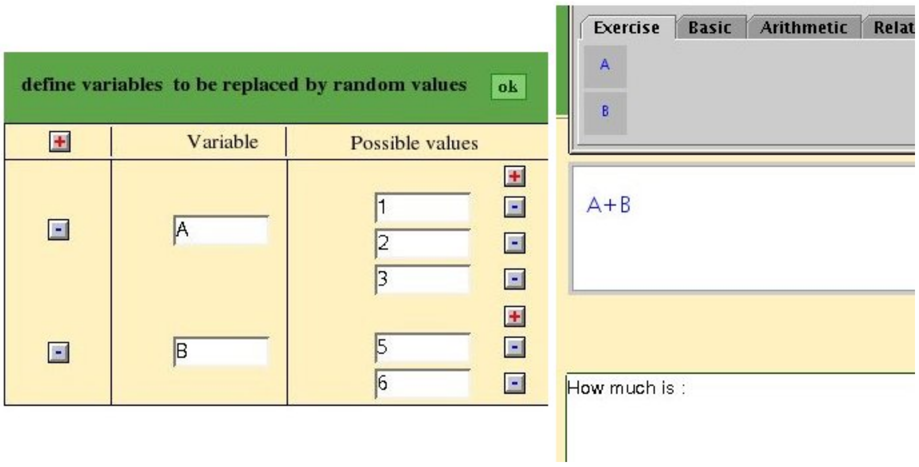

Figure 7. Authoring Randomizer variables

When starting such an exercise, each time the variables to be randomized are instantiated with random values taken from the range defined by the author and these values are propagated throughout the whole solution space of the exercise. This way, the whole class of exercises is authored at once by authoring one highly annotated randomized exercise.

7. Conclusion and Future Work

Summarizing, the authoring tool for exercises in ActiveMath offers visual authoring of solution spaces of complex multi-step interactive exercises annotated with feedback, hints and different types of transitions, using semantic evaluation for reducing the solution space by collapsing the semantically equivalent answers of the learner. Such exercises can be rendered using different tutorial strategies, that facilitate learning.

Now a couple of words about possible future developments.

Due to the highly semantic knowledge representation of exercises, and intelligent reporting facilities of the exercise sub-system of ActiveMath the results of the interaction of many learners with the same exercise can be collected and analyzed. For each exercise state, the answers of different learners that did not match any condition in the transitions issued at this state can be semantically analyzed and compared to each other. This provides a basis for identifying common conceptual and procedural errors of learners. Using the results of such log data analysis, the exercise can be further enhanced by author who can add transitions that diagnose particular misconceptions and provide remedial actions against such errors in the places in the exercise when they likely occur.

Another interesting direction of further development is recording different aspects of behaviour of the author while creating an exercise. For example the author would enjoy the possibility to define the template for the structure of the exercise by creating several first steps together with some transitions and reuse this template for authoring next portion of steps. For instance, author could specify, that for each state of the exercise, containing a new task, he always makes three transitions and three corresponding next states, containing e.g. feedbacks for correct answer, incorrect answer and a hint.

Also, the exercise authoring tool does not yet fully support all the expressive power of the exercise language. For example, still to be implemented is the interface for authoring complex constraints to be satisfied for the learner's answer in case of multiple answers in one interaction that can not be evaluated component-wise, but as a whole.

Finally, the exercise editor has to be continuously extended to support new add-ons to the exercise system providing the authors with features that facilitate exercise authoring, e.g. extensions that allow for randomizing over continuous intervals and sets of functions, etc..

References

[1] G.Goguadze, "Generic CAS Interaction", Deliverable D25, LeActiveMath Consortium, 2005.

[2] G. Goguadze, A. González Palomo, E.-Melis, "Interactivity of Exercises in ActiveMath", *Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences - Sharing Good Practices of Research, Experimentation and Innovation*, Edited by Chee-Kit Looi, David Jonassen, Mitsuru Ikeda, Frontiers in Artificial Intelligence and Applications, Volume 133, pages 109 - 115, 2005,

[3] J.E. Hopcroft, J.D. Ullman, "Introduction to Automata Theory, Languages, and Computation", *Adison-Wesley Series in Computer Science* ISBN 0-201-02988-X, 1979

[4] K.R. Koedinger, V. Aleven, T. Heffernan, B. McLaren, and M. Hockenberry, "Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration", *In the Proceedings of 7th Annual Intelligent Tutoring Systems Conference.* Maceio, Brazil, 2004

[5] Maple T.A.^{TM} Website, <http://www.maplesoft.com/products/mapleta>

[6] E. Melis. Choice of Feedback Strategies. In D.G. Sampson Kinshuk and P. Isaias, editors, *Cognition and Exploratory Learning in the Digital Age (CELDA 2005).* IADIS, iadis, dec 2005.

[7] E. Melis, G. Goguadze, M. Homik, P. Libbrecht, C. Ullrich and S. Winterstein, "Semantic-Aware Components and Services of ActiveMath", *British Journal of Educational Technology*, (ISSN 0007-1013), vol. 37(3), pp. 405-423, Blackwell Pub. May 2006

[8] E. Melis and C. Ullrich, "Local and Global Feedback", *AI in Education AIED-2003*, ed. U. Hoppe, F. Verdejo and J. Kay, pp. 476-478, IOS Press, 2003

[9] The OpenMath Standard, <http://www.openmath.org>