

ACTIVEMATH - Generation and Reuse of Interactive  
Exercises using Domain Reasoners and Automated  
Tutorial Strategies

Giorgi Gogvadze

Dissertation

zur Erlangung des Grades Doktor der Ingenieurwissenschaften  
(Dr.-Ing.) der Naturwissenschaftlich-Technischen Fakultät I  
der Universität des Saarlandes

Saarbrücken, 2010

|                     |   |
|---------------------|---|
| <b>Dekan</b>        | Prof. Dr. Holger Hermanns, Universität des Saarlandes   |
| <b>Vorsitzender</b> | Prof. Dr. Raimund Seidel, Universität des Saarlandes  |
| <b>Gutachter</b>    | PD. Dr. Erica Melis, Universität des Saarlandes<br>Prof. Dr. Jörg Siekmann, Universität des Saarlandes<br>Prof. Johan Jeuring, Universiteit Utrecht |
| <b>Beisitzer</b>    | PD. Dr. Helmut Horacek, Universität des Saarlandes  |
| <b>Kolloquium</b>   | 30. Mai 2011  |

## Abstract

In this thesis we present an approach to interactive exercises for mathematical domains. We have defined a generic knowledge representation for multi-step interactive exercises and developed the exercise sub-system within the ACTIVEMATH learning environment. In order to achieve deep domain diagnosis and automated feedback generation in multiple mathematical domains, a distributed semantic evaluation service architecture was developed. This architecture allows to remotely query multiple domain reasoning services using a domain-independent query format.

In our approach, we allow for hand-crafted exercises for which we provide authoring tools, as well as fully or partially automatically generated exercises. We have also built an extensible framework for tutorial and presentation strategies, which allows for reusing the same exercise with different strategies.



## Kurzzusammenfassung

In dieser Arbeit wird ein Ansatz für interaktive Übungsaufgaben für mathematische Domänen vorgestellt. Wir haben eine generische Wissensrepräsentation für die mehrschrittigen, interaktiven Aufgaben definiert und ein Übungsaufgabenmodul der Lernplattform ACTIVE MATH entwickelt. Um die tiefe Diagnose der Lerneraktionen und die automatische Feedback-Generierung für eine Mehrzahl der mathematischen Domänen zu erreichen haben wir eine Service-Architektur für verteilte semantische Auswertungen entwickelt. Diese Architektur erlaubt Fernanfragen an eine Mehrzahl der "domain reasoning services" mittels eines domäne-unabhängigen Anfrageformats.

Bei unserem Ansatz erlauben wir die manuelle Erzeugung von Aufgaben und bieten dafür Autorentools an, so dass die Aufgaben manuell, aber auch ganz oder zum Teil automatisch generiert werden. Wir haben auch ein erweiterbares Framework für tutoriellen und Präsentationsstrategien entwickelt, das es ermöglicht, die gleiche Aufgabe mit mehreren Strategien wiederzuverwenden.



## Acknowledgment

First of all, I would like to thank Erica Melis who was the supervisor of my thesis for her continuous support and valuable advices. Her ideas and guidance made this work possible. Her careful proofreading of this thesis has lead to significant improvements.

I am deeply indebted to Jörg Siekmann who has been extremely supportive to me all these years. Both his enthusiasm and critics have played a constructive and motivating role in my work. I would also like to thank him for the detailed review of this thesis and a lot of valuable feedback.

I would like to thank my colleagues and ex-colleagues at Saarland University and DFKI for their support, in particular the members of the `ACTIVEMATH` team Eric Andres, Michael Dietrich, Ahmad Salid Doost, Andrey Girenko, Alberto González Palomo, Paul Libbrecht, Bruce McLarren, Dimitra Tzovaltzi, Oliver Scheuer, Sergey Sosnovsky, Stefan Winterstein, Martin Homik, and Arndt Faulhaber, as well as members of the former `ΩMEGA` group Andreas Meier, Marvin Schiller, Marc Wagner, and Claus Peter Wirth for a very fruitful and inspiring research environment during the past several years.

Special thanks to Alberto González Palomo for his ideas and his contribution to the implementation of big parts of the `Exercise Subsystem` of `ACTIVEMATH`. Additional thanks to Marvin Schiller who helped organizing the bibliography for this thesis, thanks to Stefan Winterstein for providing the data on scalability tests, thanks to Britta Seidel for correcting my German text.

Last but not least, I am infinitely grateful to my parents who encouraged me to do this work and supported me in all possible ways; and I am thankful to my brother for he has always been there for me.





## Extended Abstract

We present a generic approach for intelligent interactive exercising within an e-learning environment. The backbone of this approach is the generic descriptive knowledge representation format for multi-step interactive exercises and the distributed semantic evaluation service architecture of ACTIVE-MATH that allows to connect to domain-aware semantic services for multiple domains.

The knowledge representation and its semantics is part of a framework for user-adaptive tutorial strategies, in which multiple strategies can be applied to the same exercise. The main novelty of the semantic domain reasoning service architecture is a generic format for queries, specific to the educational purpose of the respective exercise. A domain reasoner provides a refined diagnosis upon user actions in order to generate feedback in multiple domains.

The exercise sub-system within the ACTIVEMATH learning environment can be viewed as a multi-domain ITS system capable of fully automatic exercise generation, as well as playing authored exercises, and hybrid - partially authored and automatically enhanced exercises. The latter give more control to teachers, who can decide which parts of the exercise need to be authored manually, so that the rest can be left to be automatically generated with the help of domain reasoners.

In comparison to other ITS', for each new domain the exercise sub-system of ACTIVEMATH only needs a domain reasoning component to be connected as a service to the semantic evaluation service architecture. The other ITS components in ACTIVEMATH such as the tutorial module, and the student model can be reused, if they are populated with the domain ontology for the new domain. We have also developed a framework for the creation of new domain reasoners as modules for the YACAS system, which is already connected to ACTIVEMATH as a service. This makes the implementation of new domain reasoners easier and removes the efforts of the integration with the ACTIVEMATH system.



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introduction</b>                                | <b>3</b> |
| 1.1      | Motivation . . . . .                               | 3        |
| 1.2      | Contributions . . . . .                            | 5        |
| 1.3      | Overview . . . . .                                 | 7        |
| <b>2</b> | <b>Background and Related Work</b>                 | <b>9</b> |
| 2.1      | Interactive Learning Environments . . . . .        | 10       |
| 2.1.1    | IMS QTI Standard . . . . .                         | 10       |
| 2.1.2    | CAS-powered Exercise Systems . . . . .             | 11       |
| 2.1.3    | Intelligent ILEs with Stepwise Solutions . . . . . | 13       |
| 2.2      | Early Computer Aided Instruction Systems . . . . . | 14       |
| 2.3      | Intelligent Tutoring Systems . . . . .             | 15       |
| 2.3.1    | ITS Architecture . . . . .                         | 15       |
| 2.3.1.1  | Domain Model . . . . .                             | 15       |
| 2.3.1.2  | Student Model . . . . .                            | 18       |
| 2.3.1.3  | Teaching Model . . . . .                           | 19       |
| 2.3.1.4  | User Interface . . . . .                           | 22       |
| 2.3.2    | SCHOLAR . . . . .                                  | 23       |
| 2.3.3    | Model Tracing Tutors . . . . .                     | 24       |
| 2.3.3.1  | The PUMP Algebra Tutor (PAT) . . . . .             | 24       |
| 2.3.3.2  | ANDES Physics Tutor . . . . .                      | 25       |
| 2.3.3.3  | Ms. Lindquist . . . . .                            | 27       |
| 2.3.4    | Constraint Based Tutors . . . . .                  | 27       |
| 2.3.5    | ITS Authoring . . . . .                            | 29       |
| 2.3.5.1  | Eon Tools . . . . .                                | 29       |
| 2.3.5.2  | REDEEM Environment . . . . .                       | 30       |
| 2.3.5.3  | CTAT - Cognitive Tutors Authoring Tools . . . . .  | 30       |
| 2.4      | The ACTIVEMATH Learning Environment . . . . .      | 31       |

|          |  |           |
|----------|--|-----------|
| <b>3</b> | <b>Knowledge Representation</b>  | <b>35</b> |
| 3.1      | Requirements and Motivation . . . . .                                  | 35        |
| 3.2      | Structure of an Exercise FSM . . . . .                                 | 37        |
| 3.3      | Types of Exercise States . . . . .                                     | 40        |
| 3.3.1    | Task. . . . .  | 40        |
| 3.3.1.1  | Metadata of Tasks . . . . .  | 40        |
| 3.3.1.2  | Usage of the Task Metadata . . . . .                                   | 42        |
| 3.3.2    | Feedback . . . . .   | 43        |
| 3.3.2.1  | Usage of Feedback Metadata . . . . .                                   | 48        |
| 3.3.3    | Interaction . . . . .  | 51        |
| 3.3.3.1  | Selections . . . . .   | 52        |
| 3.3.3.2  | Multiple Choice Questions . . . . .                                    | 52        |
| 3.3.3.3  | Orderings . . . . .  | 54        |
| 3.3.3.4  | Mappings . . . . .   | 54        |
| 3.3.3.5  | Fill-in-blank . . . . .  | 56        |
| 3.4      | Transitions Between Exercise States . . . . .                          | 58        |
| 3.4.1    | Diagnosis . . . . .  | 58        |
| 3.4.2    | Condition . . . . .  | 61        |
| 3.4.2.1  | Evaluating Multiple Choice Questions . . . . .                         | 63        |
| 3.4.2.2  | Evaluating Mappings . . . . .  | 64        |
| 3.4.2.3  | Evaluating Ordering . . . . .  | 64        |
| 3.4.2.4  | Evaluating Blanks . . . . .  | 65        |
| 3.4.3    | Evaluation using a CAS . . . . .                                       | 66        |
| 3.4.4    | Context Evaluation . . . . .   | 69        |
| 3.4.5    | Generic Query Format . . . . .   | 71        |
| 3.4.6    | Interoperability between different CASs . . . . .                      | 73        |
| 3.4.7    | Diagnosis using a CAS . . . . .  | 74        |
| 3.4.8    | Further Intelligent Diagnoses . . . . .                                | 75        |
| 3.4.9    | Domain Reasoner Queries in the Exercise Transitions . . . . .          | 77        |
| 3.4.10   | Special Requests . . . . .   | 78        |
| 3.5      | Modifications and Extensions of the Exercise Representations . . . . . | 79        |
| 3.5.1    | Parameterized Exercises using Randomizer . . . . .                     | 80        |
| 3.5.2    | Static and Dynamic Language Mappings . . . . .                         | 82        |
| 3.5.3    | Hierarchical Exercise Representations . . . . .                        | 84        |
| 3.5.4    | Attaching Tutorial Strategies . . . . .                                | 86        |
| 3.5.5    | Atomic Strategies . . . . .  | 86        |
| 3.5.6    | Strategies in MATHEFÜHRERSCHEIN and LEACTIVE-MATH . . . . .            | 87        |

|          |   |            |
|----------|---|------------|
| 3.5.7    | Composite Strategies in ATuF and ALOE . . . . .         | 88         |
| 3.5.8    | Exercises with local strategies . . . . .               | 91         |
| 3.5.9    | Multi-Exercises in MAPS MOSSAIC . . . . .               | 92         |
| 3.5.10   | Feedback generation . . . . .                           | 95         |
| 3.5.11   | Local Exercise Generators in a Curve Sketching Exercise | 97         |
| 3.6      | Interoperability of the Exercise Format . . . . .       | 100        |
| 3.6.1    | Compliance to IMS QTI Standard . . . . .                | 100        |
| 3.6.2    | Relation to Other Exercise Languages . . . . .          | 101        |
| <b>4</b> | <b>Exercise System Architecture</b>                     | <b>103</b> |
| 4.1      | Exercise Controller . . . . .                           | 105        |
| 4.2      | Exercise Manager . . . . .                              | 106        |
| 4.3      | Diagnoser . . . . .                                     | 106        |
| 4.3.1    | Employing Mathematical Services . . . . .               | 108        |
| 4.3.2    | Semantic Service Broker . . . . .                       | 111        |
| 4.3.3    | Syntactic and numeric comparisons . . . . .             | 111        |
| 4.3.4    | Semantic evaluation using CAS . . . . .                 | 112        |
| 4.3.5    | Domain Reasoner Diagnosis . . . . .                     | 112        |
| 4.4      | Exercise Generator . . . . .                            | 113        |
| 4.4.1    | StaticGenerator . . . . .                               | 114        |
| 4.4.2    | Domain Reasoner Exercises . . . . .                     | 115        |
| 4.5      | Exercise Events . . . . .                               | 116        |
| 4.6      | Local Student Model . . . . .                           | 116        |
| 4.7      | Applying Tutorial Strategies . . . . .                  | 117        |
| 4.8      | Presentation Strategies . . . . .                       | 118        |
| <b>5</b> | <b>Exercise Authoring</b>                               | <b>123</b> |
| 5.1      | Visual "Authoring-by-Doing" Tool . . . . .              | 123        |
| 5.1.1    | Editing Exercise Nodes . . . . .                        | 125        |
| 5.1.2    | Editing Transitions . . . . .                           | 126        |
| 5.1.3    | Adding Metadata . . . . .                               | 127        |
| 5.1.4    | Authoring Parametrized Exercises . . . . .              | 129        |
| 5.1.5    | Comparison to Related Tools . . . . .                   | 130        |
| 5.1.6    | Further Development . . . . .                           | 130        |
| 5.2      | Exercise Generation using Domain Reasoners . . . . .    | 130        |
| 5.2.1    | Selecting Tasks for Problems . . . . .                  | 131        |
| 5.2.2    | Authoring Parameters for Exercise Generation . . . . .  | 131        |
| 5.2.3    | Prolog-based Domain Reasoners . . . . .                 | 132        |

|          |   |            |
|----------|---|------------|
| 5.2.3.1  | SLOPERT . . . . .                                     | 133        |
| 5.2.3.2  | MATHCOACH . . . . .                                   | 133        |
| 5.2.4    | YACAS Reasoners . . . . .                             | 134        |
| 5.2.4.1  | Domain Reasoner for Fraction Arithmetics . .          | 135        |
| 5.2.4.2  | Domain Reasoner for Integrals . . . . .               | 135        |
| 5.2.5    | Haskell-Based Domain Reasoners . . . . .              | 136        |
| <b>6</b> | <b>Evaluation of coverage and performance</b>         | <b>139</b> |
| 6.1      | Range of Expressive Power . . . . .                   | 139        |
| 6.2      | Performance tests . . . . .                           | 141        |
| <b>7</b> | <b>Conclusion and Future Work</b>                     | <b>147</b> |
| 7.1      | Authoring Tutorial Strategies . . . . .               | 147        |
| 7.2      | More Domain Reasoners . . . . .                       | 148        |
| 7.3      | Authoring Advanced User Interfaces . . . . .          | 148        |
| 7.4      | Exercise Log Data Analysis . . . . .                  | 149        |
|          | <b>Bibliography</b>                                   | <b>151</b> |
|          | <b>A Curve Sketching Problem</b>                      | <b>173</b> |
|          | <b>B Sample Complex Strategy</b>                      | <b>193</b> |
| B.0.1    | Self-Regulation with Solicited Worked Example . . . . | 194        |
| B.0.2    | Control Strategy . . . . .                            | 203        |

# List of Tables

- 3.1 Values and sub-values of PISA competencies . . . . . 41
- 3.2 Feedback classification in ACTIVEMATH exercises . . . . . 47
- 3.3 Generating meta-cognitive feedback from observable student  
behavior . . . . . 51
- 3.4 Forms and presentation styles of interactive elements . . . . . 52
  
- 6.1 Total numbers of exercises and exercise nodes . . . . . 140
- 6.2 Tutorial and presentation strategies implemented in ACTIVE-  
MATH . . . . . 140





# List of Figures

|      |   |    |
|------|---|----|
| 2.1  | Sample encoding of an early ACTIVE MATH exercise . . . . .        | 12 |
| 2.2  | Inner and outer loops of an ITS . . . . .                         | 19 |
| 2.3  | CBM tutor architecture . . . . .                                  | 28 |
| 2.4  | Coarse architecture of ACTIVE MATH . . . . .                      | 32 |
| 3.1  | Structure of an exercise . . . . .                                | 38 |
| 3.2  | A simple exercise graph . . . . .                                 | 39 |
| 3.3  | Metadata of the task . . . . .                                    | 42 |
| 3.4  | Pizzas representing fractions . . . . .                           | 42 |
| 3.5  | A simple exercise graph transformed by the strategy . . . . .     | 49 |
| 3.6  | A presentation of a multiple choice in ACTIVE MATH . . . . .      | 53 |
| 3.7  | Representation of multiple choice questions . . . . .             | 53 |
| 3.8  | Sample ordering . . . . .   | 54 |
| 3.9  | The presentation of an ordering exercise in ACTIVE MATH . . . . . | 55 |
| 3.10 | Example of a mapping . . . . .                                    | 55 |
| 3.11 | Presentation of a mapping exercise . . . . .                      | 56 |
| 3.12 | Representation of fill-in-blank . . . . .                         | 57 |
| 3.13 | Different presentations of blanks in ACTIVE MATH . . . . .        | 58 |
| 3.14 | Excerpt from an ontology of the fractions domain. . . . .         | 60 |
| 3.15 | Representation of the misconception M1 . . . . .                  | 60 |
| 3.16 | Exercise diagnosing the misconception M1 . . . . .                | 61 |
| 3.17 | Condition for the correct answer in a multiple choice . . . . .   | 63 |
| 3.18 | Evaluation matrix for a mapping . . . . .                         | 64 |
| 3.19 | Condition matching a mapping answer . . . . .                     | 64 |
| 3.20 | Condition for the correct answer in an ordering . . . . .         | 65 |
| 3.21 | Simple condition for multiple free inputs . . . . .               | 65 |
| 3.22 | Complex Boolean condition for multiple free inputs . . . . .      | 66 |
| 3.23 | Evaluation in different contexts in Maxima CAS . . . . .          | 68 |

|      |  |     |
|------|--|-----|
| 3.24 | Diagnosis and feedback generation with CAS . . . . .                 | 74  |
| 3.25 | Applying the query 'getResults' . . . . .                            | 77  |
| 3.26 | Buggy rules in the exercise solution space . . . . .                 | 78  |
| 3.27 | Matching correct answers with domain reasoner . . . . .              | 79  |
| 3.28 | Representation of a hint request . . . . .                           | 79  |
| 3.29 | Sample exercise generator representation . . . . .                   | 80  |
| 3.30 | Transitions in the randomized exercise . . . . .                     | 81  |
| 3.31 | Custom format for simple fill-in-blank exercises . . . . .           | 82  |
| 3.32 | Mapping of the omdoc quiz module . . . . .                           | 83  |
| 3.33 | Hierarchical exercise representation . . . . .                       | 84  |
| 3.34 | Simple format for hierarchical solutions . . . . .                   | 85  |
| 3.35 | Exercise strategy in MATHEFÜHRERSCHEIN . . . . .                     | 87  |
| 3.36 | Exercise Strategy in LEACTIVEMATH . . . . .                          | 88  |
| 3.37 | ALOE two-phase exercise strategy . . . . .                           | 89  |
| 3.38 | ALOE exercise rendering . . . . .                                    | 89  |
| 3.39 | Additional relation types in an ATuF exercise . . . . .              | 90  |
| 3.40 | ATuF exercise rendering . . . . .                                    | 91  |
| 3.41 | Sample representation of local strategy . . . . .                    | 92  |
| 3.42 | MAPS MOSSAIC strategy example . . . . .                              | 93  |
| 3.43 | Representation of subgoal templates . . . . .                        | 94  |
| 3.44 | MAPS MOSSAIC subgoal ordering . . . . .                              | 95  |
| 3.45 | Feedback generation using CAS . . . . .                              | 96  |
| 3.46 | Commonly used generated KR feedback . . . . .                        | 96  |
| 3.47 | Feedback generation using a domain reasoner . . . . .                | 97  |
| 3.48 | Representation of domain reasoner-powered sub-exercise . . . . .     | 98  |
| 3.49 | Representation of a sub-exercise using a tutorial strategy . . . . . | 99  |
| 3.50 | Calculating intersections with coordinate axes . . . . .             | 99  |
| 4.1  | Architecture of the exercise subsystem . . . . .                     | 105 |
| 4.2  | Defining a custom diagnoser . . . . .                                | 107 |
| 4.3  | Semantic Service Broker . . . . .                                    | 110 |
| 4.4  | Excerpt from the hierarchy of <b>Exercise Generators</b> . . . . .   | 113 |
| 4.5  | Explorative CAS Console in ACTIVEMATH . . . . .                      | 114 |
| 4.6  | Domain reasoner-powered feedback . . . . .                           | 115 |
| 4.7  | Applying Tutorial Strategies . . . . .                               | 118 |
| 4.8  | Custom feedback layout . . . . .                                     | 119 |
| 4.9  | Different presentations of flag feedback in ACTIVEMATH . . . . .     | 120 |
| 4.10 | Two interaction interfaces . . . . .                                 | 121 |

4.11 Different multiple interaction interfaces . . . . . 121

5.1 Exercise authoring user interface . . . . . 124

5.2 Sample exercise graph . . . . . 125

5.3 Microstructure editor window . . . . . 126

5.4 Authoring blanks and selections . . . . . 127

5.5 Defining transitions . . . . . 128

5.6 Authoring relevant metadata . . . . . 128

5.7 Authoring Randomizer variables . . . . . 129

5.8 Authoring Domain Reasoner Exercises . . . . . 132

5.9 Explorative Domain Reasoner Console . . . . . 134

5.10 Custom error feedback in fraction exercises . . . . . 136

5.11 IDEAS Domain Reasoner exercises use the same generator . . 137

6.1 Average values of the stress test results . . . . . 143

6.2 Median values of the stress test results . . . . . 144

6.3 90<sup>th</sup> percentile values of the stress test results . . . . . 145



# Chapter 1

## Introduction

### 1.1 Motivation

Interactive exercises are essential for learning in any ITS/e-learning system. As a counterpart of other (static) learning objects containing factual or exemplary information about the domain, interactive exercises represent learning objects that can train and assess the skills of the learner.

Due to the rich domain knowledge and student models Intelligent Tutoring Systems (ITS) developed a very close interaction with the student and were able to provide the student's learning progress comparable to traditional individual instruction [Corbett, 2001].

Most Intelligent Tutoring Systems integrate complex domain models for diagnosis. All the other components, such as the student interface, teaching model and student model also rely on the structure of the domain. Even though the domain knowledge is in some cases separated from the pedagogical knowledge, the authoring of content for such ITS systems involves a lot of work to adapt the teaching strategies and student model to the concrete domain. The current practice is to build a new Tutoring System for each domain rather than building one generic system and then "load" a domain into it, e.g. Cognitive Tutors [Anderson et al., 1995], [Murray, 2004]. Also, Andes [Van Lehn et al., 2005] - is built for Physics, particularly for Newton mechanics problems; Ms. Lindquist [Heffernan, 2001] - for algebra, Cognitive Tutors [Anderson et al., 1995] - each implemented specifically for the domain (Algebra Word Problem Tutor, Geometry Tutor, LISP Tutor etc.).

Building such Tutoring System is expensive. It requires domain experts,

educationalists and programmers to implement it.

Programming the domain model and adjusting the reasoning to the pedagogical needs of individual exercises is a tedious process and might take years. For example, the Andes Physics Tutoring System has been in development for about 10 years and the total amount of existing problems in the system, mentioned in [Van Lehn et al., 2005] is 356.

Another issue that has not been properly addressed so far is the composition of ITS systems to tackle cross-domain problems. Since the domain model is tightly integrated with the student model in most ITS systems and tutorial models are customized for the domain of instruction, there are no mechanisms for seamlessly merging several ITS' from different domains.

Unlike ITS, recent Web-based e-learning systems that try to define a generic representation for an interactive computer instruction are only able to provide shallow feedback on student actions, since they lack the depth of the domain knowledge. They are mostly restricted to simple multiple choice questions and do not possess a learner model. Such are, for example, systems using Question and Test Interoperability Standard of IMS Global Learning Consortium [IMS Global Learning Consortium, 2005]. Even if some free student input is possible, it is mostly analyzed using literal string comparison. Such exercises are easy to create without having any tools providing domain intelligence. However they cost plenty of time and effort, since for providing the correct diagnosis of the learner's answer many equivalent solutions have to be hand-crafted.

Other recent e-learning systems such as older versions of ACTIVE MATH [Büdenbender et al., 2002a], MathDox [MathDox, 2007], STACK and AIM computer aided assessment systems (CAA) [Sangwin, 2008, Sangwin, 2004], use Computer Algebra Systems (CASs) to verify the correctness of student's answer. Here, the system remotely queries the CAS for the result.

This approach provides some domain knowledge, where the semantic evaluation is possible, but it is restricted to the functionalities of the CAS which originally was not designed for providing intelligent diagnosis. The knowledge representation of exercises in these systems is mostly a mixture of the syntax of a particular CAS with own custom syntax. Such knowledge representation is mostly procedural and bound to the syntax of a particular CAS. For instance, AIM is using Maple [Maple, 2009], STACK uses Maxima [Maxima, 2010], and old exercises in ACTIVE MATH were using Mupad CAS [MuPAD, 2009]. The Mathdox system is not bound to the syntax of CAS but to the usage of a particular CAS. It uses the generic semantic

markup language OPENMATH for encoding mathematical expressions, but the instructions describing how to evaluate the student's answer point to a particular CAS's evaluation method. It also uses other specific script languages such as the Jelly platform and XForms which makes exercises dependent on these technologies. This makes authoring more difficult for a non-programmer.

The European research project LEACTIVEMATH has shown that Computer Algebra Systems are not able to provide enough diagnosis upon learners actions which would be sufficient for generating informative feedback within a tutorial dialogue and properly updating the learner model about learner's current mastery of domain concepts [Consortium, 2004]. This is due to the fact that CASs have no support for incremental problem solving as humans perform it and are not aware of typical errors of students. Moreover CAS systems do not provide a generic possibility to evaluate an expression in a particular mathematical context. As the mathematical *context* of the task we understand the specification of the domain concepts or rewrite rules the learner can apply to the mathematical object in the problem statement in order to proceed to the next step. Context evaluation can be achieved only after non-trivial programming that maps the mathematical contexts onto commands of a particular CAS. Even this is possible only for some CASs.

Such a tradeoff of generality versus intelligence has been among the topics of discussion of authoring technology for ITS Systems [Murray, 2004].

## 1.2 Contributions

In this thesis we present a module of the web-based learning platform ACTIVEMATH that we refer to as **Exercise Subsystem**, that has several novel aspects as compared to other approaches. This system combines features of an Intelligent Tutoring System and a generic framework for hand-crafted exercises, similarly to the ASSISTMENT Systems [Turner et al., 2005] and the Example Tracing Tutors of CTAT [Alevan et al., 2009]. However, there are some substantial differences to the above mentioned systems.

We build a system for interactive exercises, that combines hand-crafted and automatically generated exercises, and defines generic representation for exercises suitable for many domains. It is capable of quick CAS evaluation of constraints upon student's input, and uses intelligent domain reasoners for detailed diagnosis of student's input. The system can also generate various

types of feedback and give an author the opportunity to decide which parts of the exercise will be hand-crafted leaving the rest to be automatically generated. Finally, authors can employ automated tutorial strategies to enhance the exercises with the needed tutorial behavior.

We believe that this thesis contributes to knowledge representation of interactive exercises, authoring of Intelligent Tutoring Systems, Intelligent Tutoring Systems Architectures, educationally oriented semantic web-services, and provides a growing set of tools for educationalists and cognitive psychologists for research and practice strategies for math education. In summary:

**Contribution I.** A knowledge representation format for multi-step interactive exercises is developed that is suitable for (but not limited to) tasks within mathematical domains that deal with incremental problem solving. This representation allows for hand-crafted interactive exercises which can be automatically enhanced by intelligent domain reasoning tools and automated tutorial strategies. Interactive exercises in *ACTIVEMATH* can combine generative and evaluative approaches to intelligent tutoring and give freedom to human tutors to have more control over exercise at several levels.

**Contribution II.** A knowledge representation for generic queries to the domain reasoning services is defined which allows for generative and evaluative queries. We show that the expressive power of these queries is suitable for serving deep diagnosis and intelligent feedback generation for interactive exercises in several mathematical domains. Such a query format is novel and gives rise to a concept of pluggable educational domain reasoner.

**Contribution III.** The domain intelligence is provided to the system by multiple remote mathematical reasoning services accessed via a context-aware broker. Currently, the platform for remote educational domain reasoning services in *ACTIVEMATH* integrates several Computer Algebra Systems, and about 10 domain reasoning services for several mathematical domains. Guidelines for implementation of educational domain reasoners are proposed and two domain reasoners have been implemented in *ACTIVEMATH* following these guidelines. All the integrated services can be used simultaneously and the semantic context-aware broker distributes the queries to the needed system. This is the first case known to the author in which several domain reasoners are used by one ITS, doing it simultaneously for the same exercise.



**Contribution IV.** An extensible framework for tutorial and presentation strategies for interactive exercises has been developed. Strategies can be applied to authored and generated exercises and to their parts. Authors can also manually attach and configure strategies in different parts of exercises. Various tutorial strategies have been developed that satisfied the needs of educationalists and cognitive psychologists within several research and empirical projects.

## 1.3 Overview

This thesis has the following structure:

**Chapter 2** gives an overview on the background for this thesis: several types of Interactive Learning Environments and Intelligent Tutoring Systems are considered. We discuss their strengths and how we can combine them in our system.

**Chapter 3** defines a generic knowledge representation for Interactive Exercises, showing how the annotations of exercise states and transitions allow for application of intelligent strategies to the exercises. We also define a generic format for educational queries to external domain reasoning services.

**Chapter 4** describes a modular architecture of the **Exercise Subsystem** and the work flow of an exercise solving process.

**Chapter 5** describes tools for authoring and generation of interactive exercises. Here, we consider in more detail the function of the **Exercise Generator** component and describe different methods of generating exercises.

**Chapter 6** provides an evaluation of the expressive power of the defined exercise language w.r.t. usability for different mathematical courses in schools and Universities. We also provide some technical evaluation of performance of the exercise subsystem, by presenting results of the so-called stress-tests.

**Chapter 7** concludes with a short summary of main achievements of this work and defines several directions of further research, such as adding more domain reasoners, authoring tutorial strategies and advanced user interfaces for interactive exercises.

## Chapter 2

# Background and Related Work

In this chapter we look at several systems and technologies, which are related to the goals of this work.

First we consider general Interactive Learning Environments (ILE) that aim at generality and simple coaching without deep domain expertise. The most generic ILE systems represent interactive exercises as simple multiple choice questions. They are simple to build, easy to deploy, and usually have little educational impact.

Somewhat more advanced systems allow free input of the learner which is then analyzed by an integrated Computer Algebra System and the corresponding correct/incorrect feedback can be given to a learner. However no deep diagnosis is possible, and there is no informative feedback on errors.

Next we consider Intelligent Tutoring Systems (ITSs) that are designed carefully for each domain and enjoy deep domain expertise, detailed student models and automated teaching strategies. They are difficult to build, although the general architectures for such systems are well-studied and there are many implementation guidelines and techniques, as well as some authoring tools. The design of an ITS system is highly impacted by the domain and, more concretely, topics addressed in this domain. The design of domain reasoning modules is mostly driven by the structure of the domain, but sometimes it is also influenced by pedagogical decisions. For instance, the tutorial model of the ANDES2 system is aimed to use only one ideal solution of the problem and if the learner is trying to follow an alternative path, the system tries to bring the learner back to the ideal one.

Even though the main modules of an ITS are distinct and well separated, the domain model of a particular system communicates with the rest of this

system in a fixed way, so it is not possible to connect several reasoners to the same system in a general way.

There are two well-known techniques we discuss below, which ITS systems use for reasoning within a domain: Model Tracing and Constraint Based Modeling.

There are more general approaches to authoring Intelligent Tutoring Systems, such as the Eon framework and the REDEEM tool, which makes the procedure of "authoring" an ITS somewhat more comfortable, but the resulting ITSs are in general less advanced as compared to the traditional ones.

Finally, we describe ACTIVEMATH - a web-based learning environment which combines features of all three types of systems mentioned above. It is suitable for teaching various domains of knowledge, it possesses a central Student Model and Extensible Teaching Module (called Tutorial Component) and provides the infrastructure for populating the system with different domain ontologies and integrating intelligent domain reasoners.

## 2.1 Interactive Learning Environments

Interactive Learning Environment (ILE) is a general term for a computer system designed to help students to learn a domain in an interactive way. These systems usually define two kinds of interactive exercises. The first is based on simple multiple choice questions, puzzles or other kind of selections in which the whole domain expertise is hand-crafted into exercise content. They don't require domain diagnosis tools.

Exercises of the second type allow for free input of the learner. In order to check the correctness of the learner's answers, mostly syntactic comparisons are used. In cases when the free input is a mathematical formula, some semantic evaluation might be possible if a Computer Algebra System is used as a back-engine. In this case all the expressions, equivalent to a correct answer, will be accepted as a valid answers as well.

### 2.1.1 IMS QTI Standard

In a industrially oriented standardization attempt, the IMS Global Learning Consortium has produced a specification of a knowledge representation format for simple one-step interactive exercises, called IMS Question & Test Interoperability (QTI), which has been used by a variety of systems. Many

tools have been built for authoring and rendering QTI exercises. Later, a second version of the standard was issued that allowed for the representation of multi-step interactive exercises.

The QTI format offers several types of interactivity such as multiple choice questions and other kinds of selection-based interactions such as mapping or reordering. Free input is also offered, which could only be evaluated literally, using simple string comparison.

There has been an extension of QTI within the JISC-funded project "Serving Mathematics in a distributed e-learning environment"<sup>1</sup> addressing the special requirements of Mathematics within QTI. This extension offers presentation MATHML and OPENMATH formats for representing mathematical formulas. This extension, however, was never standardized by IMS Consortium.

The QTI format is a highly verbose and heterogeneous format, which represents a flow of an exercise program dressed into XML syntax. The format allows uncontrolled usage of flag-like variables possessing arbitrary implicit semantics which allow for "representing" exercises, which can then be run by systems that understand the implicit semantics of the flag variables. The specification of QTI version 2 supports templates that define common marking schemes for interoperability. For more information about the issues of the QTI format and its comparison to other exercise languages, see [Gogvadze et al., 2006].

### 2.1.2 CAS-powered Exercise Systems

There have been several Computer Assisted Assessment (CAA) systems for mathematical domains, that used Computer Algebra Systems as back engines for evaluating interactive exercises. Among the most prominent ones are AiM [Sangwin, 2004] operating with Maple, STACK [Sangwin, 2008] operating with Maxima, and the commercial MapleTA<sup>TM</sup> [MapleTA<sup>TM</sup>, 2010] using Maple. Also earlier versions of ACTIVEMATH encoded such exercises [Libbrecht et al., 2001] using Maple and Mupad CAS [MuPAD, 2009] as a back engine. The exercises in these systems are represented using a custom syntax mixed with the syntax of the corresponding CAS. They also contain small programs for the cases when the simple CAS command was not enough for evaluating the student's answer. In Figure 2.1 a sample representation of

---

<sup>1</sup>[http://maths.york.ac.uk/serving\\_maths](http://maths.york.ac.uk/serving_maths)

an exercise for permutations is shown (taken from [Libbrecht et al., 2001]). This code is not likely something school math teachers would be able to encode.

```
<code id="mapleExercises_Code1">
<data>
  <startup><command>evalSilent</command><param>
    with(group);
    P1:=[[1,3,4,2]];
    P2:=[[1,4,6,2,3,5]];
  </param></startup>
  <startup><command>eval</command><param>
    printf("Compute the inverse of the following two permutations:");
  </param></startup>
  <startup><command>eval</command><param>
    printf("1: %a 2: %a\n",P1,P2);
  </param></startup>
  <startup><command>eval</command><param>
    printf("Please use the variables Q1 and Q2 for the respective inverses!");
  </param></startup>
  <eval><command>eval</command><param>
    if assigned(Q1) and assigned(Q2)
      and evalb(mulperm(P1,Q1)=[])
      and evalb(mulperm(P2,Q2)=[]) then
      1
    elif assigned(Q1) and assigned(Q2) and evalb(Q1=I1) then
      printf("Answer Q1 is correct but not Q2, since\n
        %a * %a = %a", P2, Q2, mulperms(P2,Q2))
      <.....>
    fi;
  </param></eval>
</data>
</code>
```

Figure 2.1: Sample encoding of an early ACTIVEMATH exercise

Other systems such as WALLIS [Mavrikis and Maciocia, 2003], and MathDox<sup>2</sup> exercises, are using semantic markups OPENMATH or content MATHML for representing formulas, which makes them less dependent on a concrete CAS.

However, MathQTI inherited the other QTI problems described above, and the MathDox language mixes a custom XML format with other scripting languages bound to the concrete programming environments which makes it again quite messy and hard to understand by human encoders.

The WALLIS format is most similar to the early versions of ACTIVEMATH's exercise format. It is generic, not bound to a particular implemen-

---

<sup>2</sup><http://www.mathdox.org>

tation, nor to the concrete CAS to be used as a back-engine. For more about the interoperability of these formats, see [Gogvadze et al., 2006].

### 2.1.3 Intelligent ILEs with Stepwise Solutions

Two important examples of intelligent ILEs capable of guiding the students through the stepwise solutions of mathematical problems are MATHXPERT [Beeson, 1996] and APLUSX [Nicaud et al., 2003]. These systems implement their own mathematical reasoning engines, instead of using the existing CASs. Such decision is motivated by the fact that CASs are not designed for pedagogical purposes and are not capable of stepwise reasoning.

MATHXPERT is an ILE for algebra and calculus which provides students with an interface for solving problems in a stepwise manner. In this interface the student can choose a rewrite rule he wants to apply to an expression from the menu and the system applies the rule for him. The student can also tell the system to do the step or the rest of solution completely automatically. In this tool there is no room for free student input and hence the student is not given an opportunity to make a mistake in his calculations.

The mathematical reasoning engine of MATHXPERT has been implemented from scratch. The problem solving strategies used by the system are designed to be similar to those of humans. Granularity of a step in the interaction with a student is based on the level of detail generally appropriate for the topic. For example, if the student is performing differentiation, it is considered that the arithmetic transformations can be all done in one step.

In the previous versions of the system the granularity of steps was made adaptive to each student individually. Since MATHXPERT has a student model which keeps track of the knowledge level for each concept involved, the system could configure itself to suggest bigger steps for concepts already learned by the student. However, such adaptation lead to badly structured solutions very irregular granularity of steps and the decision was made to abandon such an adaptation.

APLUSX is an ILE for learning algebra. It provides students with an interface for stepwise problem solving, but as opposed to the menu-based interaction in MATHXPERT, it allows the students to enter arbitrary mathematical expressions using a graphical user interface for entering formulas. The system gives several types of feedback, such as correctness of the step, and some progress indicators, such as "the exercise is solved", "you still have to reduce" and so on. Additionally, APLUSX defines several commands

such as *calculate* for numerical expressions, *expand-and-reduce* and *factor* for polynomial expressions, *solve* for equations and inequalities. These commands can be executed on request of the student and the system performs the needed calculations for the student automatically.

As in the MATHXPRT, the underlying mathematical reasoning engine is implemented from scratch. In addition to the reasons the MATHXPRT implementers had for developing own reasoning module, another reason for doing so in APLUSX is the wish to provide a self-contained system not relying on an external CAS which might be expensive.

## 2.2 Early Computer Aided Instruction Systems

The first generation of computer based tutors are so-called Computer-Aided Instruction systems (CAI). Preceding the rise of Intelligent Tutoring Systems (ITS), discussed in the next section, there has been some evolution of early CAI systems, so called AFO (ad-hoc frame oriented) systems. See [Nwana, 1990] for an overview of these systems.

The first kind of CAI systems in the 1950s were 'linear programs' described in [Skinner, 1954], in which the material was presented step by step in a series of so-called frames. Each of the frames contained a simple question and the student received immediate notification of correctness. But independently of the student's results, the system proceeded to present the next frame in the sequence.

The next type of systems were 'branching programs' that used the student's response for choosing the next problem [Crowder, 1959]. But in this case the teaching material became unmanageably big.

The next generation called 'generative systems' generated teaching material and adapted automatically to the student's performance and required difficulty.

The latter systems were the closest to ITS, but lacking a structured view of fundamental aspects, such as knowledge representation for the instruction domain. The tutoring strategies were ad-hoc and not supported by any theories of learning. The student's performance and knowledge state was not systematically modeled and hence not used for making the systems dynamically adjustable to the student.



The Intelligent Tutoring Systems, that started to be developed in the 1970s, addressed these issues and the methodology developed since then has been used until now for hundreds of systems.

## 2.3 Intelligent Tutoring Systems

Intelligent tutoring systems are advanced ILEs, that aim at personalized learning similar to one-to-one tutoring. This goal has been set in order to address Bloom's two-sigma problem. A study, described in [Bloom, 1984], showed that one-to-one tutoring compared to a group instruction achieved 2 standard deviations (2 sigma) more learning effect. In particular, the study has shown that 50% of students who had been tutored in a one-to-one instruction setting, performed better than 98% of the students who had been taught in the classroom. Moreover, comparing the effect of one-on-one tutoring with the effect of a number of other variables and treatments, Bloom found that none of the others produced an effect as effective as 2 sigma.

In the following section we present the general structure and methodology of ITS followed by the brief description of some well established systems.

### 2.3.1 ITS Architecture

The main goals to be addressed by an ITS architecture are to specify *what* the system should be teaching, *whom* it is teaching, and *how*.

All traditional ITS systems contain at least four main components: a *domain model*, a *student model*, a *teaching model*, and a *student interface*.

#### 2.3.1.1 Domain Model

This component contains the material *what* to teach and represents the knowledge about the domain of instruction. This information is used by the ITS to instruct the learner about the subject, it evaluates student's answers in exercises and provides feedback.

A domain model of an ITS consists of two parts. First, there are domain concepts, represented, for example, by using semantic networks or ontologies. Second, it consists of the representation of the problem-solving ability to be taught in the form of production rules or constraints.

There have been different approaches to domain modeling in ITS. Currently, two most established general approaches are Model Tracing and Constraint-Based Modeling techniques.

These two approaches emerge from two theories of learning, while they agree upon one main principle: learning means to convert facts to procedural knowledge (problem solving skills). However, they differ in how to facilitate this process.

Model Tracing is based on the ACT-R theory of Anderson [Anderson, 1993] that considers the direct mastering of procedural knowledge as the key to learning. Declarative knowledge is only used for introduction, necessary for learning the production rules of the domain. The rest of the time the student is learning to apply the production rules of the domain to construct the incremental solutions of problems, thus training procedural knowledge.

Model Tracing (MT) tutors represent the domain by a set of production rules within a so-called domain reasoner module. For each problem, the reasoner can generate one or more stepwise solutions, constructed by incrementally applying the production rules that transform the task expression.

Constraint Based Modeling (CBM) is based on the theory of Ohlsson (see [Ohlsson, 1996]) who claims that the key to acquiring procedural knowledge is to help the student to first understand the declarative knowledge. Therefore, the problems given to the student should not necessarily concentrate on training the incremental rules, but rather on understanding the declarative knowledge.

The CBM systems represent the domain by a set of constraints. For each problem, the constraints that are relevant for this problem have to be satisfied. If the solution of the student is violating some of the relevant constraints feedback is presented to the student that explains the error and provides a conceptual hint.

Each of the approaches have pros and cons, which were extensively discussed in [Mitrovic et al., 2003a], [Kodaganallur et al., 2005], and further in [Mitrovic and Ohlsson, 2006], and in [Kodaganallur et al., 2006].

The main difference between MT and CBM tutors is the approach to evaluation of the learner's action. MT tutors use so-called *process*-centric approach, in which the process of the solution needs to be traced, whereas CBM tutors are *product*-centric in that they only evaluate the current state of learner's solution. Another fundamental difference is that if the MT tutor can not find the student's answer any of its generated solution paths, the answer is considered incorrect, whereas the CBM tutor does not impose any

particular solving strategy for the problem, so that if the student's answer does not violate any constraints relevant for the given problem, the answer is considered correct.

The main advantage of the Model Tracing tutors is the ability to generate consequent solution steps starting at any point in the set of all possible solution paths. This gives the opportunity to provide hints on how to proceed and give out correct next steps. The main limitation of the Model Tracing domain reasoners is the combinatorial explosion when building the solution space for the exercise, which has to include all possible expert and erroneous solution paths.

The main advantage of the CBM tutors is that they are easier to build and more scalable than MT tutors. The main limitation of the CBM tutors, is that they can not (easily) provide feedback on the next steps in the solution, given the student's answer.

Additional methods are used to avoid the usual combinatorial explosion, mostly custom designed for the domain of instruction. For instance, the ANDES system uses the "color-by-numbers" method in order to evaluate the correctness of an equation entered by the student. In ANDES, the solution graph of the problem contains the list of all variables in the problem together with their value when the problem has been solved. "Color-by-numbers" substitutes these values into the student's equation. If the resulting arithmetic equation numerically balances, the student's answer is correct, and incorrect otherwise. If not all variables in the problem have values, so-called 'ugly numbers' (randomly chosen numerical values) are substituted instead of these variables. If the equation balances with the ugly value it is likely that it would balance with the symbolic value as well.

Another method, which has become custom for domains of physical quantities, used by ANDES is to compare the dimensions of the physical quantity in the student's answer. If the dimensions of student's answer are not the same as in the correct answer, it is clear that the answer is wrong without further reasoning. These and other optimization techniques are described in [Shapiro, 2005].

Production rule systems of model tracing tutors, often referred to as *domain reasoners*, usually consist of two types of rules - expert rules, representing the expert model of the domain and so-called buggy rules, representing frequently occurring erroneous reasoning patterns. Systems like BUGGY [Burton and Brown, 1978], DEBUGGY [Burton, 1982], and SLOPERT [Zinn, 2006] make extensive use of buggy rules and allow for combining

them with each other and with expert rules to model complex errors. The ANDES system defines a hierarchy of so-called error handlers, as described e.g. in [Van Lehn et al., 2005]. In this hierarchy, some general typical errors are modeled and the more specific context dependent error handlers can override the general ones.

Some domain reasoners make restrictions on the depth of recursion when combining expert and erroneous reasoning rules while diagnosing complex errors. It makes sense from a pedagogical perspective and at the same time reduces the computation time. It is unlikely that the student combines more than a certain number of correct and buggy reasoning steps within a single interaction with the system. The precise number might depend on the domain and the cognitive parameters of the learner. Specific constraints can be also constructed, the violation of which diagnose a typical error and triggers corresponding feedback.

### 2.3.1.2 Student Model

The student model represents the student, *whom* the ITS system aims to instruct. The function of the student model is to enable the system to adapt to individual learners in order to present the instructional content that corresponds to the current needs of the student. The student model observes the student's performance and creates qualitative representations of his knowledge of domain concepts and skills.

Some systems model the student's expertise using skill taxonomies to represent different aspects of the mastery of domain concepts.

Other information on the student's performance, such as the interaction history, statistics on the correctness of solved problems etc. can also be stored in such a model. However, the student model is not only a (continuously updated) storage of the student's data, but also a reasoning component, transforming the observable student behavior, such as the responses of the student into the modeled unobservable data such as beliefs about the student's achieved literacy.

The simplest student models developed by early ITS are so-called *scalar* models defining one common measure of student's progress throughout the domain topics.

Much more information is collected in *overlay* models, which assigns knowledge mastery values to each domain concept.

An extension of these models are *differential* models, which differentiate

between the domain knowledge which the student has already been exposed to and which has not yet been presented to the student.

In addition to modeling student's knowledge of concepts, many ITSs model student's errors, by so-called *perturbation* or *buggy* rules, which extend the expert model with a *bug library*. The goal of the tutoring system is to grow the student's subset of the expert knowledge alongside with eliminating the buggy subset.

There are various techniques for realizing student models such as Bayesian belief networks [Millán et al., 2010], Transferable Belief Model [Smets, 1994], models based on formal logic, expert systems, plan recognition and others (see e.g. [B.P.Woolf, 2008] Chapter 3 for an overview).

### 2.3.1.3 Teaching Model

The teaching model encodes the instructional strategies used by the tutoring system. It represents the tutorial behavior of the system, specifying *how* to teach.

According to [Van Lehn, 2006] the instructional behavior of the Intelligent Tutoring System consists of two feedback loops, as depicted schematically in Figure 2.2.

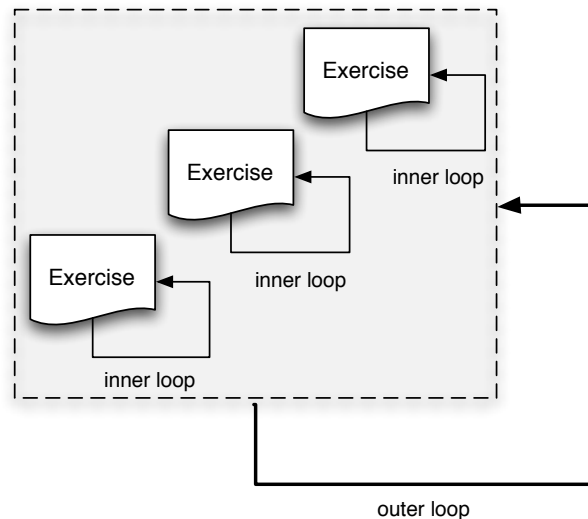


Figure 2.2: Inner and outer loops of an ITS

The inner loop refers to feedback given to the student within a single exercise as a reaction to the student's actions, whereas the outer loop selects the next problem based on the current knowledge state of the student.

The outer loop has been addressed only partially by some systems. According to [Van Lehn, 2006], four methods for selecting tasks have been used so far:

- The student selects a next task manually from the given menu
- The tasks are assigned by the system in a fixed sequence
- A *mastery learning* scenario is used, in which students keep getting exercises for training the same knowledge components until they are mastered
- A *macro adaptation* method chooses the next task based on the overlap between the knowledge components needed for the task and the student's mastered knowledge components.

The ACT Programing Tutor is an example of a system using a mastery learning scenario [Corbett and Anderson, 1995]. One of the recent achievements addressing the outer loop is the development of PAIGOS [Ullrich, 2008] - a hierarchical task network planner for adaptive course generation within the ActiveMath learning environment.

The inner loop, to which this thesis mainly contributes, concentrates on feedback in response to student's action within an exercise step. There are several aspects that play a major role in this process, such as the content of the feedback, its timing, the appropriate scaffolding strategy, etc. There are various types of feedback, as described e.g., in [Van Lehn, 2006] and [Van Lehn et al., 2005]:

- minimal feedback: just a correct/incorrect notice, possibly highlighting the error position (*flag feedback*)
- error-specific feedback on an incorrect step
- procedural hints on what to do next
- conceptual hints providing information about the involved knowledge components

- bottom-out hints or correct solutions for a problem or a problem step

Depending on the tutorial goals several other parameters can be taken into account for feedback presentation:

- feedback timing : *immediate* vs *delayed* feedback
- turn taking of a tutorial dialogue: *solicited* vs *unsolicited* feedback

Immediate feedback is presented directly after each student step. Delayed feedback is presented only after the student presented his complete solution.

Solicited feedback is only given on request of the student, whereas unsolicited feedback is given independently of the student's request.

Complex combinations of feedback types and other parameters such as feedback timing, number of trials for the step, solicited and unsolicited help, form different *tutorial strategies* for the inner loop.

Most ITS systems have a hard-coded standard tutorial strategy. For example, model tracing tutors (described in Section 2.3.3 below) have been criticized for a too simple tutorial model, where the error feedback was directly attached to the buggy production rule. It was presented to the student when the rule application was diagnosed. The same holds for Constraint Based Systems (see section 2.3.4 below), that attach error feedback directly to the constraints and present it to the student in case the constraint is relevant for the solution, but not satisfied.

For example, the ANDES system implements a so-called minimal non-invasive strategy [Lehn et al., 2002], that is hard-coded in the system and applies to all exercises. This strategy helps students to follow the expert solution path and gives feedback on request. In ANDES, the strategy implements three types of feedback: *flag feedback*, *error feedback* and *what's wrong help*. The sequence of hints of several types leads the student to follow the expert solution path. The order in which different feedback types are given is predefined. In the ATLAS-ANDES system, the situation is improved by introducing a dynamic strategy, conducting a dialogue with the student, based on a solution path of the problem. This dialogue uses a hierarchical menu of possible principles applications and it aims at eliciting the knowledge construction by the student [Rosé et al., 2001]. The ITS for algebra, called Ms. Lindquist [Heffernan, 2001], [Heffernan et al., 2008] proposes a similar dialogue-based architecture, called ATM (Adding Tutorial Model).

#### 2.3.1.4 User Interface

The user interface is a very important part of an ITS system, as it establishes the interaction between the student and the system. It carries the important function of presenting the domain of instruction to the learner and providing means of interaction of the student with the problem. The problem solving environment is highly dependent on the domain of instruction as well as on pedagogical strategies. Until now there has not been a systematic classification of user interfaces in ITSs.

User interface modules range from simple multiple choice and free text input to complex simulation environments.

Following a tutorial strategy with immediate feedback, the User Interface might allow the student to construct his solution step by step with the possibility to enter one step at a time and provide immediate feedback, before the next step. In this cases, the interface for the next user interaction might even depend on the result of the previous interaction. For example, if the student provides only a partial answer, the next interaction can present the same step with the partial answer filled in. Once the student provides the complete answer, the next task will be presented. The stepwise solution strategies can define various types of scaffolding which have an impact on the complexity of the User Interface. Natural language dialogue systems need the least complexity - the student only needs a text field for submitting his answer. Based on the student's interaction history the dialogue system adapts the content and timing of feedback, as well as the turn taking of a tutorial dialogue.

Depending on the diagnostic capabilities of the system and the tutorial strategy, some disambiguation questions or socratic sub-dialogues might be suggested by the system. The challenging part for the User Interface is not to overload the student with interface components and let him concentrate on the problem solution instead.

In case of a delayed feedback strategy the student can submit the complete solution to a problem and the system has to analyze this solution. Depending on the domain of instruction and the diagnosis possibilities it can be a challenging task to design a User Interface for delayed feedback strategy.

Some systems, capable of natural language understanding, allow the student to enter the solution in the form of an essay and then tries to analyze the text.

Other systems provide template-based input, suggesting the structure of



his solution to the student. This can, on the one hand, make it easy for the system to analyze the student's solution. On the other hand, it can be easier for the student to just fill in the suggested solution structure instead of writing long sentences of text.

Systems like ANDES or the Algebra Cognitive Tutor provide a complex User Interface consisting of several interaction areas in which the student can manipulate different representations of the problem using graphs, tables, diagrams, as well as formula input. These different components of the User Interface help to analyze the problem and enter the solution.

Another type of User Interface is a simulation environment, in which the student can virtually interact with simulations of components of a complex physical environment such as, for instance, a large air compressor [Rickel and Johnson, 1999] or a nuclear power plant [Méndez et al., 2003].

### 2.3.2 SCHOLAR

The first Intelligent Tutoring System was SCHOLAR [Carbonell, 1970]. This tutoring system was capable of a mixed initiative tutorial dialogue on geography of South America. The input of the student and the output of the program were sentences in english.

The knowledge in the expert knowledge module was represented in a semantic network whose nodes instantiated geographical objects and concepts. This network also represented the ideal student model - an overlay model, in which each node was associated with a flag to indicate whether the student knows the domain concept represented by that node. Moreover, modeling of student errors was proposed by introducing so-called perturbations. However, this proposal was never realized.

The tutorial strategies of SCHOLAR consisted mainly of logical topic selections from the agenda, provided by a teacher. If the topic was too general, SCHOLAR randomly generated a subtopic.

In order to communicate with the student, SCHOLAR possessed some language processing capabilities. The text of questions and feedbacks was generated using a set of templates, filled with information from the semantic network. The parsing of student's answer was done by matching keywords from a list that was dynamically generated from the network for each question.

Important contribution of SCHOLAR is that it introduced many central methodological principles of ITS design, such as separation of tutorial strate-

gies from domain knowledge, more explicit representation of knowledge using a semantic network, and student modeling.

### 2.3.3 Model Tracing Tutors

The Model Tracing approach to ITS is based on the ACT-R theory of Anderson [Anderson, 1993]. The knowledge is constructed using a rule based production system.

There have been a variety of Model Tracing Tutors built for the various domains of instruction, e.g. LISP programming [Anderson and Reiser, 1985], Geometry [Anderson et al., 1985], Algebra [Koedinger and Anderson, 1997]. We give a brief overview of the most important systems below.

#### 2.3.3.1 The PUMP Algebra Tutor (PAT)

The PUMP Algebra Tutor (or PAT) is a Model Tracing Tutor for teaching introductory algebra [Koedinger and Anderson, 1997]. Each problem in this system involves some algebraic problem with several quantities. Using the complex multi-part graphical interface, students have to represent quantities as spreadsheet columns, answer the questions in the spreadsheet rows, construct algebraic representation of the relationships between these quantities and graph these relationships.

PAT is a classical Model Tracing tutor, which solves each problem step-by-step with the student and provides assistance if necessary. It tries to trace the student's solution path and checks it against many possible solution paths in its domain model. Tracing the student step-by-step gives an opportunity to provide individualized instruction, concentrating on the current solution state. The system gives flag feedback on errors, and in case a typical error is diagnosed by matching the student's answer against a buggy rule, the system presents feedback that indicates what is wrong with the answer.

In addition to the error feedback, PAT also provides help on request. The system gives three types of advices, such as a reminder of the current goal, a general description of how to achieve the goal (conceptual hint), and finally a description of exactly what problem solving action to take (procedural hint). For each of these three types of feedback, several levels of detail are possible.

The PAT tutor uses two student modeling techniques: Model Tracing and Knowledge Tracing. Model Tracing is used to monitor the student's progress within one exercise and Knowledge Tracing is used to monitor student's

growing knowledge from problem to problem. As the student solves problems, the PAT tutor estimates the probability that each domain concept is in the learned state, based on the students actions. This process involves a simple Bayesian decision process described in [Corbett et al., 1997].

The PAT tutor implements a mastery learning strategy for the outer loop. The students continue solving problems until the probability that each involved domain concept is at the learned state has almost reached certainty.

### 2.3.3.2 ANDES Physics Tutor

ANDES is an intelligent problem solving environment for quantitative problem solving in introductory college physics [Van Lehn et al., 2005]. It was designed for the purpose of an intelligent homework environment. It does not have an automated outer loop, but suggests a hierarchical menu from which the student can select problems manually.

There are two major versions of the ANDES system: ANDES1 and ANDES2. Both systems share the same domain model and the user interface.

The domain model of ANDES (1,2) consists of a set of production rules that are used to automatically generate solution spaces of problems in the domain of Newton mechanics. Apart from rules, for each of the major mechanics principles ANDES has developed a problem solving method, that incorporates a multi-step hierarchical plan for generating an application of the principle. Solving a problem is realized as a form of state-space search, that iteratively chooses the problem solving methods to be applied. For more details on problem solving in Andes, see [Van Lehn et al., 2005] (pages 30-31).

ANDES also possesses a library so-called called error-handlers which can match typical errors of the student and then help to generate error-related feedback.

ANDES (1,2) possesses a complex user interface, in which the students can read the problem statement, draw coordinate axes and vectors representing Newtonian forces, define variables and input equations. The system has a mathematics package, that solves equations for students to free them from mathematical steps and let them concentrate on aspects of physics.

ANDES (1,2) implements an elaborate but fixed tutorial strategy for feedback. The system gives unsolicited feedback on slip errors and several types of feedback on student's request: *what's wrong help* pointing to the error and providing error specific feedback. It also generates a sequence of hints on

how to proceed (*next step help*) with different levels of detail. Typically the next step help consists of a *teaching hint* containing information about the concepts involved, and the *bottom out* hint giving out (parts of) the correct solution. The module for Next Step Hint generation is called Procedural Helper.

Another pedagogical decision in ANDES is to relieve students from algebraic manipulation tasks by providing them with algebraic manipulation tools such as the equation solver. This way the students can concentrate on actual physics principles rather than on algebraic calculations.

The main difference between ANDES1 and ANDES2 is the content of feedback produced by the Procedural Helper. The ANDES1 has a plan recognition module, based on the Bayesian network student model. The system attempts to recognize the plan of the student and gives the next step hint following this plan [Gertner et al., 1998].

Empirical studies have shown that although ANDES1 was successful and increased learning outcomes by 1 standard deviation in comparison with traditional instruction, the Bayesian student model was not the key to its success. It turned out that the elaborate tutorial strategy was key to its success. Also, since the ANDES system does not have an automated outer loop and the next exercises to tackle are manually assigned to students, the Bayesian student model was not used for selecting the next tasks.

Inconsistencies in the plan recognition results suggested that in most cases of erroneous reasoning the learner does not have any expert solution plan. So the Procedural Helper in the ANDES2 is giving hints directed to return the learner back to the fixed standard expert solution path, rather than trying to recognize the student's path.

Another method used in ANDES2 was the Conceptual Helper module, which was called when ANDES2 decided that the student is unfamiliar with a specific principle of physics or has a misconception. In this case the Conceptual Helper presented so-called "mini-lessons" to the learner, which are adapted to the student's problem solving context.

Years of research, continuous evaluation and deployment allowed ANDES researchers to build a powerful learning environment for introductory physics and make significant contributions to the methodology and architectures of Intelligent Tutoring Systems. It is now used widely in the US and elsewhere in more than 200 schools.

### 2.3.3.3 Ms. Lindquist

The new generation of model tracing tutors with an advanced tutorial model was announced by Ms. Lindquist - a system using multiple tutorial strategies for teaching "symbolization" in an algebra domain [Heffernan et al., 2008]. The kind of problems considered contain a modeling step in which the students have to formalize some aspect of the real world. This introduces the need for a more strategic approach to student feedback than in simple incremental problem solving.

Ms. Lindquist implements an elaborate tutorial dialogue architecture, called ATM (Adding Tutorial Model), built on top of a model tracing tutor. In addition to the traditional *outer* and *inner* loops, this system defines a so-called *knowledge-search* loop. This means, that when the student has reached an impasse in his reasoning, the system starts asking him questions, the answers of which are not always parts of the problem solution, but elicit the knowledge construction of the student that helps him to solve the problem. Such scaffolding also appears in ATLAS-ANDES within the modified Conceptual Helper in which the static "mini-stories" of ANDES2 are replaced by interactive knowledge-eliciting dialogues, realized as multiple-choice questions.

In comparison to classical model tracing tutors, that generate feedback from text templates coupled with production rules, the ATM architecture generates a dialogue plan for each error student makes. The tutorial model of Ms.Lindquist contains seventy seven rules to generate these plans.

After the student model has made a diagnosis of the student's action, there are three types of responses: give a bug message, give a hint or use the tutorial sub-strategy addressing the error. There are two kinds of strategies involved - so-called *Knowledge Remediation Dialogue* and *Knowledge Construction Dialogue* that invoke multi-step plans to deal with particular errors.

Evaluations of Ms.Lindquist have shown the effectiveness of multiple tutorial strategies applied to algebra "symbolization" problems.

### 2.3.4 Constraint Based Tutors

The Constraint Based Modeling (CBM) approach is based on the theory of learning from performance errors [Ohlsson, 1996].

The basic idea of the constraint-based approach is to define a set of constraints that describe the expert domain knowledge and model the student

behavior by diagnosing whether the constraints relevant to the current problem have been satisfied.

The structure of a domain model has the similar structure for all CBM tutors. The domain model is a collection of so-called state descriptions. Each state description is a triple consisting of a *relevance constraint*, *satisfaction constraint* and *feedback* in case of constraint violation. When evaluating a student's solution, first the relevance constraints help to identify which constraints were relevant to the current state of the problem and then the satisfaction constraints of the relevant state descriptions are checked. If a violation is detected, the student is provided with feedback to inform him of the violation. Different tutorial behaviors w.r.t. feedback presentation can be expressed in cases when several constraints are violated at once. Several levels of feedback and hints are possible, as well as some macro-adaptation.

All CBM tutors have similar architecture, as shown in the Figure 2.3.

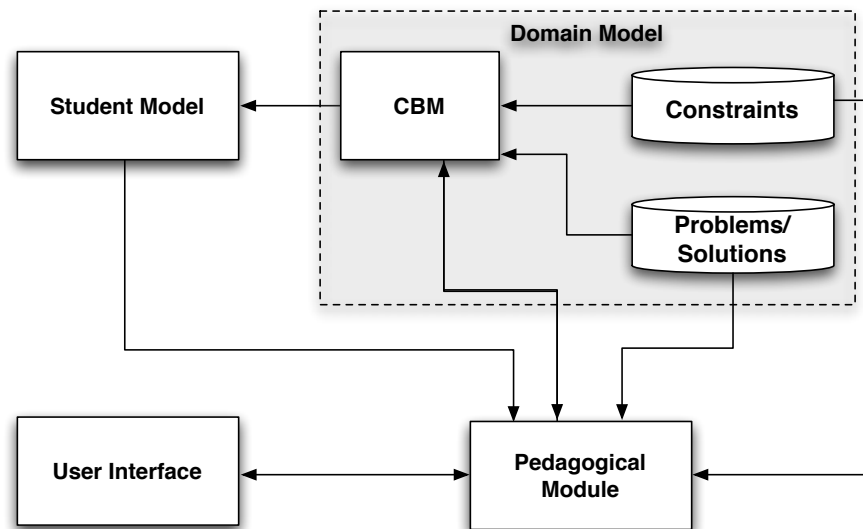


Figure 2.3: CBM tutor architecture

It refines the general ITS architecture with CBM specific components, such as a constraint database, a constraint based modeler, and database of problems and their solutions.

The most prominent Constraint Based Tutors built so far are: SQL Tutor teaching the well-known database language [Mitrovic and Ohlsson, 1999],

COLLET-UML for object oriented design [Baghaei et al., 2005], NORMIT for data normalization [Mitrovic, 2002], CAPIT - a tutoring system for capitalization and punctuation [Mayo et al., 2000]. Finally, an authoring framework ASPIRE [Mitrovic et al., 2008] has been built to support rapid development of new CBM tutors.

### 2.3.5 ITS Authoring

Since building an ITS for each new domain from scratch has significant costs, there have been approaches to domain independent tools for authoring all main modules of an ITS: the domain model, the teaching model, the student model and the user interface. Requirements and principles for building educationally oriented authoring tools are summarized in [Murray, 2003b]. We briefly present some important examples of generic authoring tools, relevant for our work on representation and authoring of exercises and tutorial strategies.

#### 2.3.5.1 Eon Tools

The first system was called KAFITS (Knowledge Acquisition Framework for ITS)[Murray, 1991]. An authoring tool for the domain ontology and an editor for creating instructional strategies was created. The strategies had a form of so-called Parametrized Action Networks.

The results of a study with human tutors using these authoring tools showed for the first time that intelligent tutors can be built with human resources comparable to building CAI.

In the follow up project further authoring tools have been developed, called Eon tools [Murray, 1998, Murray, 2003a]. Here, the domain model to be authored may consist of a domain ontology and learning objects as well as an executable (e.g. rule-based) part, representing domain inference and problem solving. Apart from expert knowledge, Eon allowed to create a database of buggy rules. It also represented misconceptions attached to the domain ontology. Eon has a layered overlay user model, in which the layers represent several decision layers, that assign mastery values to each level of domain hierarchy: Lessons, Topics, Topic Levels, Presentation Contents and events. Eon uses so-called 'flowlines' for authoring tutorial strategies. These are diagrams, graphically representing the procedure of a strategy. Each strategy can have input parameters, local variables and can return values. In order to

choose among strategies some meta-strategies are used. Each meta-strategy has application conditions and strategy parameters, where the application conditions of meta-strategies can be adaptive to student's behavior, as inferred from the student model.

### 2.3.5.2 REDEEM Environment

Similarly to Eon, the REDEEM system (Reusable Educational Design Environment and Engineering Methodology) creates an ITSs with less effort, less training and knowledge, and provides support for authors to help them create tutorial strategies. (Murray, 1999).

But, they have little domain knowledge compared to other ITS systems. According to Murray, this system is primarily concerned with the development of Tutoring Strategies (like Eon). REDEEM does not support the construction of a domain model.

It is routine for AI researchers to represent domain knowledge in production rules, but unintuitive for most teachers. Therefore, REDEEM reduces the teacher's options to modify low level instructional behavior for ease of authoring.

Instead, REDEEM focuses on the authoring of pedagogical rules. It allows tutors to set parameters of tutorial strategies and to define application conditions for tutorial actions. However, the strategies themselves are not freely definable, like in Eon, but they are all pre-defined.

### 2.3.5.3 CTAT - Cognitive Tutors Authoring Tools

Another approach that makes authoring of Intelligent Tutoring Systems easier for non-programmers is the Cognitive Tutors Authoring Tool (CTAT), developed at CMU [Koedinger et al., 2004]. Here the ease of implementation for tutors is not the only goal, but another goal is to improve the cognitive task analysis and the exploration of pedagogical content knowledge.

Apart from authoring Cognitive Tutors, the CTAT tool allows to author so-called "Pseudo Tutors" (now also called Example Tracing Tutors) that mimic the behavior of an intelligent tutor. Instead of automated AI tools, the teachers author the exercise by demonstrating and recording student's behavior and the reaction of the system. This means that instead of encoding the domain production rules as in the rule based model tracing tutors, the author simulates the student's possible solution path together with the



feedback of the system. This process is recorded by a so-called **Behavior Recorder** into an annotated exercise graph which is later used to trace the student's steps through the problem solution.

Such a simple tool removes the necessity of programming domain production rules, that makes it more accessible to teachers. But since there is no automatic pattern matching that can combine any possible next step with any possible next but one step, all the alternative correct and incorrect solutions have to be hand-crafted by the author.

Each step can be annotated with one or more hints, which are ordered sequentially at runtime. Finally, so-called knowledge labels are attached to the links in the behavior graph to refer to the underlying domain concepts.

Studies have shown that the Pseudo Tutor system finds intelligent tutorial behavior equivalent to the production based Cognitive Tutor. Authoring both kinds of systems is comparably time consuming.

The recorded behavior graphs, used for building Pseudo Tutors, can be also used to guide the development and testing of a cognitive model for use in a Cognitive Tutor, as described in [Aleven et al., 2006]. However, Pseudo Tutors do not remove the necessity to program domain production rules. The behavior graphs just help to explore the cognitive model of the domain and can be used as unit tests for the domain model.

A collection of tools for authoring Cognitive Tutors includes a number of further components, assisting building the Cognitive Tutors as well as external editor for editing Jess rules for the cognitive model.

CTAT tools have also been used for a machine learning approach to rule creation, using the **Simulated Student** - a machine learning agent [Matsuda et al., 2005]. This agent analyses the sample behavior graphs produced by the human tutor and infers production rules. The **Simulated Student** agent is integrated into the CTAT tools, so that the authors can test the generated production rules by letting **Simulated Student** solve the new problems. Advanced authors can then manually modify the generated Jess rules, if desired.

## 2.4 The ACTIVEMATH Learning Environment

The web based interactive learning environment ACTIVEMATH is an intelligent tutoring system that supports advanced outer and inner feedback loops. Figure 2.4 shows the coarse architecture of ACTIVEMATH.

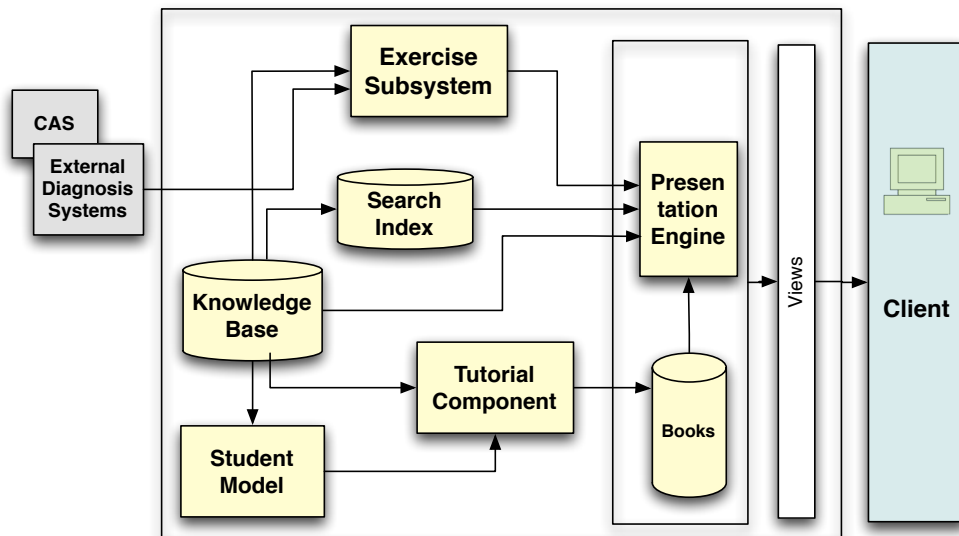


Figure 2.4: Coarse architecture of ACTIVEMATH

One of the major features of ACTIVEMATH is adaptivity. The Tutorial Component of ACTIVEMATH, responsible for the adaptation in the outer loop, is capable of automatic course generation and presentation which is adapted to the learner's goals, the given learning scenario and other individual parameters of the learner, as presented in [Ullrich, 2008] and in [Ullrich and Melis, 2009].

The Tutorial Component actively communicates with the ACTIVEMATH Student Model, gathering information about the learner's progress and making conclusions about his current knowledge mastery.

An important feature of the ACTIVEMATH's Student Model is that it dynamically extracts relations and metadata from the content ontology and makes use of their semantics. This metadata, together with the diagnosis of the student actions in interactive exercises, enables the Student Model to estimate the student's competencies. This way the Student Model of ACTIVEMATH is not bound to any concrete domain, but can work with any domain, that can be "loaded" into the system. For more information on the Student Model of ACTIVEMATH see [Faulhaber and Melis, 2008].

The Exercise Subsystem of ACTIVEMATH implements the inner loop and provides the Student Model with evidence about learning progress,

based upon the diagnosis of learning events in interactive exercises.

Early versions of ACTIVEMATH provided only hand-crafted interactive exercises, which lacked deep diagnosis of the learners' actions and provided the `Student Model` only with a flag feedback.

This thesis shows how we enriched ACTIVEMATH with intelligent reasoning tools in multiple domains and provided a framework for reusable tutorial strategies for the inner loop.

Among other advantages of the `ACTIVEMATH Exercise Subsystem`, is the possibility to create hybrid, partially authored and partially generated exercises. This allows for more control of the exercise design by the author or teacher, letting the automatic intelligent tools *assist* and not *control* the tutoring process. The author or teacher can reuse the automatic existing tutorial strategies or devise his own strategies manually.

The authored part of the ACTIVEMATH exercise is similar to the behavior graph of CTAT. This graph is, however, more compact than the graph produced by CTAT. Some of the alternative steps of the learner can be evaluated by a CAS, and hence all the semantically equivalent steps of the learner can be collapsed into one. Each step is annotated with some relations to domain concepts, which is similar to the knowledge labels in CTAT. However, in ACTIVEMATH the exercise steps are annotated with more educational metadata such as difficulty, competency and others, as described in Section 3.3. Such annotation provides a basis for application of different tutorial strategies, as we show later in this thesis.

We make use of domain reasoner systems connected to the ACTIVEMATH as Web-services, and use a generic format for educational queries and semantic Web broker for distributing these queries. Thus, we offer a web service architecture that can be connected to multiple interoperable domain reasoning services.

This gives us the unique possibility of semi-automatically creating cross-domain exercises, that use several domain reasoners simultaneously.

The framework for tutorial strategies of ACTIVEMATH seems to fulfill the requirements of educationalists and cognitive psychologists as demonstrated in several research projects, see, e.g. [Tsovaltzi et al., 2009] for the ALOE project, [Eichelmann et al., 2008] for the ATuF project, and [Polushkina, 2009] for the MAPS MOSSAIC project. Many more strategies were built for the needs of teachers in schools and universities where ACTIVEMATH exercises are used for teaching different mathematical subjects.

As opposed to other generic environments that construct multiple tutorial

strategies, such as Eon and REDEEM, ACTIVEMATH allows human tutors to customize the automated strategies and locally overwrite them or manually tune their parameters. This is decided because in many cases teachers wish to have custom strategies, which are statically attached to the tasks within exercises.

Also, generic automatic tutorial strategies that model the tutorial process in various domains, are insensitive to the needed values of parameters when applied to a concrete task. In comparison, an experienced tutor will always have some rough estimate on the average values of such parameters, e.g. the maximal number of trials or hint levels sensible for the current task. In case of an adaptive strategy, the values of strategy parameters may change dynamically, as the strategy continuously adapts to the learner.

# Chapter 3

## Knowledge Representation

### 3.1 Requirements and Motivation

One of the goals of this work is to find a balanced way to incorporate manually authored and automatically generated exercises. Hence, knowledge representation becomes an important part of this thesis. This is due to the fact that the intuitive representation for human authors might not be suitable for automatic generation and vice versa. So joining them in one system is among the novelties this work produces.

Exercises in the highly domain-sensitive Intelligent Tutoring Systems do not have a fixed body. Feedbacks to the actions of the learner and the respective next steps are generated using the diagnosis of the learner's answer to the current task and previous interactions with the learner. In this case the tutorial dialogue is fully controlled by the system and can not be directly influenced by a human tutor/author.

Interactive Learning Environments that aim at generality, define knowledge representation for exercises that do not require any domain intelligence for the evaluation of the learner's answers. In this case, exercises are represented as (possibly nested) multiple choice questions only and the domain information needed for an exercise is then encoded by the author of the exercise. He provides the content for the choices and leaves to the learner to choose among the alternatives. Exercises allowing free input are not frequently used, since there are no means to automatically analyze such input. An example of a general-purpose representation is the IMS Question Test Interoperability (QTI) standard [IMS Global Learning Consortium, 2005].

The knowledge representation of the third type of systems, such as AIM and STACK, MATHDOX, and MATHCOACH, uses semantic representation for mathematical formulas. This allows to evaluate the free input of the learner by a Computer Algebra system. However, such a knowledge representation is bound to the syntax of a particular CAS and contains program code. Such exercises can not be authored by teachers without programming skills.

Our goal is to design a generic knowledge representation that allows for the best features of these ILEs. We define a declarative exercise format, suitable for authoring and automatic generation. We allow for different types of interactivity including free input of the student.

Mathematical expressions use a semantic knowledge representation that can be interpreted with different tools, such as CASs - for quick evaluation of constraints upon the student's input, as well as powerful domain reasoners that achieve deeper domain diagnosis and generate (parts of) a correct solution. We also go beyond the existing approaches to interactive exercises by enabling hybrids of authored and generated exercises and their reusability using different tutorial strategies.

In this chapter, we define a knowledge representation format for interactive exercises, that satisfies the following criteria:

1. The format is suitable for representing exercises from different mathematical domains. It can represent problems that are solved incrementally.
2. The format is not bound to the concrete implementation (several exercise players possible). This means that the knowledge representation of an exercise provides all information needed for running it and does not define semantic under-specifications that have to be made concrete in the implementation of the exercise player.
3. Different tutorial strategies can be applied to the same exercise representation. The exercise representation can be automatically enriched by the tutorial strategy that defines general patterns for reacting to the actions and requests of the learner while solving the exercise.
4. Exercise representation does not hardcode any presentation information. Different presentation strategies can be applied to the same exercise representation, that define the placement and look of the tasks

and feedbacks, style of interactive elements, and other presentational aspects.

5. Different mathematical services can be used for providing diagnosis of the learner's actions. This is achieved via a uniform knowledge representation standard -OPENMATH for mathematical expressions and a generic query language, specifying how this expressions have to be manipulated for exercises (see Section 3.4.5).
6. The format is suitable for hand-crafted exercises as well as automatically generated exercises, and the hybrid types, such as partially generated exercises that use a hand-crafted skeleton.

**Writing Convention.** In the following, we assume that formulas in ACTIVEMATH are represented using the OPENMATH format but due to the verbosity of this format we use the OPENMATH notation in example listings only if it is necessary to show the microstructure of particular formulas. In all other cases, formulas are represented in a more human-readable way.

## 3.2 Structure of an Exercise FSM

A multi-step solution of a mathematical problem can have one or more correct solution paths, that are sequences of solution steps. 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 a process in which the learner is moving step by step within this graph and each next node he is visiting depends on the answer he provides for the task at the previous step.

Therefore, an exercise can be coded as a finite state automaton of states representing tutorial actions of the system and transitions between these states which represent reactions to the learner's answers.

We define three basic types of exercise **states**. These are: **task**, **interaction** and **feedback**.

- **task** defines the problem statement of the current task to be solved by the learner

- **interaction** contains interactive elements by which the learner provides the solution to the task or performs other actions such as asking for hint or solution, giving up the exercise etc.
- **feedback** represents feedback to the learner's action

Each state contains a set of **transitions** issued in this node. There can be two types of transitions between exercise states: **unconditional** and **conditional**. Unconditional **transitions** connect **tasks** with **interactions** and **feedbacks** with the following **tasks**. Conditional **transitions** connect **interactions** with **feedbacks** given the learner's action satisfies a particular condition. Each transition contains a pointer to the target state. The conditional **transition** also contains a condition to be satisfied by the learner's input which triggers this transition and a diagnosis of the learner's answer. The states that do not issue any transitions are terminal. As soon as the terminal state is reached the exercise is finished.

Schematically, nodes and transitions of a sample exercise are shown in Figure 3.1.

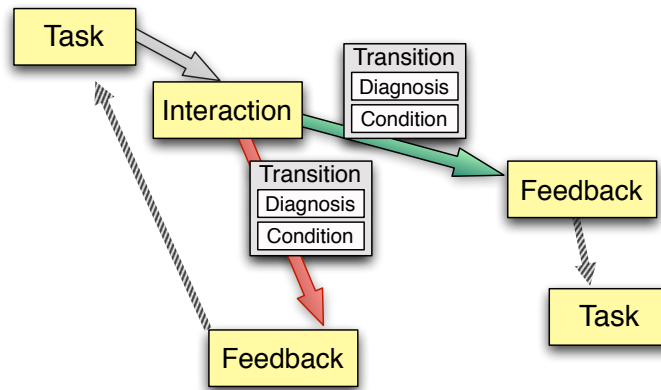


Figure 3.1: Structure of an exercise

An instance of these three types of states of an exercise represent a single case of an internal feedback loop, which we call an **exercise step**. The separation between **task**, **interaction** and **feedback** is important for the reusability of the exercise representation with different tutorial strategies, which rely on such a structure. It is not mandatory to have all three types of nodes in each **exercise step**. For example, it can happen that the exercise



provides no feedback on student's actions. An exercise can also contain loops in which there is an unconditional transition from a **feedback** back to the **task**, as shown in Figure 3.1.

Consider the exercise, in which the learner has to differentiate the function  $f(x) = 2 \cdot x$ , i.e., to calculate  $f'(x) = (2 \cdot x)'$

The simple solution space of this exercise is shown in Figure 3.2.

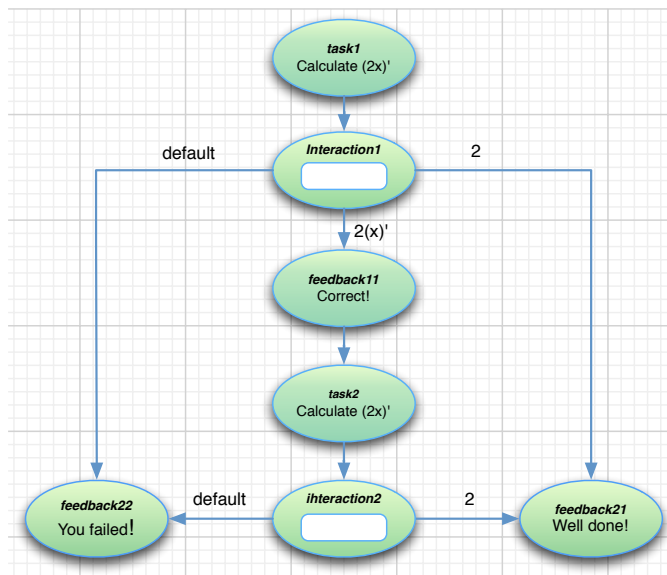


Figure 3.2: A simple exercise graph

The first state (*task1*) defines the task to calculate  $(2x)'$ . An unconditional condition leads to the interaction node (*interaction1*). If the learner's answer is equal to  $2 \cdot (x)'$  the next state will be *feedback11*. The learner receives feedback and is forwarded to the next state *task2*. If the learner enters the final correct answer - 2, he is forwarded directly to the final state *feedback21*. In all other cases, the default transition is triggered and the learner is forwarded to *feedback22*. Negative feedback is then given and the exercise is finished. Similarly, in the *interaction2*, in case of the correct answer the learner is forwarded to the final state *feedback21*, and in case of any other answer the learner is presented with the final *feedback22*.

### 3.3 Types of Exercise States

In this section we consider three types of exercise nodes : **task**, **feedback** and **interaction**.

The elements **task** and **feedback** are static and each of them has its own metadata. The element **interaction** has a more complex internal structure aimed at a complex interactivity.

#### 3.3.1 Task.

Each **task** contains a description of the task for the learner to perform. This element can contain text with formulas, images, applets, links and other multimedia and formatting elements.

Each task trains or assesses particular cognitive dimensions of some domain concept at a certain difficulty level. The domain concepts are represented as items in the knowledge base of the **ACTIVEMATH** system. Relation to these concepts have to be provided in the **task** metadata.

There are two main reasons for annotating the tasks of the exercise states with these metadata.

First, the **Student Model** needs information about the the concepts and their cognitive dimensions that need to be mastered in the task together with the diagnosis upon the learner's answer to this task in order to update the learner's mastery values. The quality of this updating defines the estimation of the learner's current literacy in the domain which, in turn may activate further tutorial actions of the system.

Secondly, as indicated e.g. in [Narciss, 2006] the design of feedback strategies is strongly influenced by the learning goals and other cognitive and meta-cognitive parameters of the task. In manually authored exercises as well as in automatically enhanced ones, these metadata annotations together with the diagnosis of the learner's answer serve as the basis for feedback authoring/generation.

##### 3.3.1.1 Metadata of Tasks

For describing cognitive dimensions of the concepts that the task is training we use a competency approach adopted from PISA, as described in [Niss, 2002] and [Klieme et al., 2004].

The set of values and sub-values of competencies are summarized in Table 3.1.

| competency value | sub-value   |
|------------------|---|
| think            | scope, formulate, generalize  |
| argue            | judge, find_choose  |
| model            | decode, encode  |
| solve            | apply_algorithms  |
| represent        | decode_understand_representations,<br>decode_understands_relations, switch_choose |
| language         | translation, interpret_manipulate, formal_rules                                   |
| communicate      | none  |
| tools            | calculator, search, concept_map, CAS, plotter                                     |

Table 3.1: Values and sub-values of PISA competencies

In our approach, the assumption is that different competencies are the building blocks of mathematical literacy: the student is mathematically literate if he has sufficient degrees for the set of competencies. These degrees are represented by the competency levels. We define the following values for competency levels : 'elementary', 'simple conceptual', 'multi step', and 'complex', as taken from PISA specification [Klieme et al., 2004] <sup>1</sup>.

Additionally to the competencies and their levels, an orthogonal difficulty level is used. The difficulty dimension is adopted from IEEE LOM standard [IMS Global Learning Consortium, 2002] and represents the technical difficulty of the task for the target audience. The target audience is represented by another LOM metadata `learningcontext`. The possible values for difficulty are 'very\_easy', 'easy', 'medium', 'difficult' and 'very\_difficult'.

A sample metadata record of the `task` element is shown in Figure 3.3. In this task the learner has to encode the graphical representation of the fraction into a formal one. The focus concept for this task is the concept of a fraction, the competency value is "model" with a sub-value "encode", and

<sup>1</sup>The complete names of the competency levels are: Level I, Computation at an elementary level; Level II, Simple conceptual solutions; Level III, Challenging multi-step-solutions; Level IV, Complex processings (modelings, argumentations)

the competency level is "simple\_conceptual".

```

<task>
  <task_metadata>
    <relation type="for">
      <ref xref="mbase://openmath-cds/rational1/fraction"/>
    </relation>
    <competency value="model" subvalue="encode"/>
    <competencylevel value="simple_conceptual"/>
    <difficulty value="easy"/>
  </task_metadata>
  <content>
    <CMP xml:lang="en">Write down a fraction shown in the
      picture in the form of the pizza.</CMP>
  </content>...
</task>

```

Figure 3.3: Metadata of the task

Depending on the concrete pizza, the difficulty can vary. In our case the pizza, displayed in Figure 3.4a has been annotated as "easy". It would be a more difficult task to deal with a pizza from Figure 3.4b. These values are estimated by the author of the exercise.

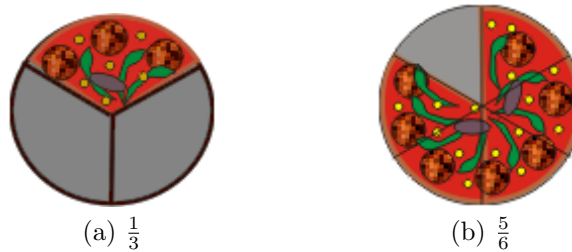


Figure 3.4: Pizzas representing fractions

### 3.3.1.2 Usage of the Task Metadata

Summarizing the above, the task of an exercise step contains the relation to the focus concept of the task, the record of competencies trained/assessed in this task, a competency level of the task, and the level of difficulty.

Given the exercise step is annotated with this metadata record, after each exercise step is performed, this metadata record together with the success rate of the student is propagated to the **Student Model** of ACTIVE MATH,

as described in Section 4.5. Based on this information, the **Student Model** updates the mastery values for the corresponding domain concept. The mastery of a concept consists of a vector of masteries w.r.t. to each competency. Updating takes the competency level and difficulty into consideration.

Another usage of the metadata of the **task** element is to generate user-adaptive feedback. An elaborate tutorial strategy might decide what kind of feedback should be presented to the learner, based upon the metadata of the task. For example, if the value of the **competencylevel** is "elementary" the strategy might decide that a simple procedural feedback is enough for the learner to be able to proceed. In the given case the value of the **competencylevel** is "simple\_conceptual" which means some conceptual modeling is present. In this case some conceptual feedback should be presented. Furthermore, if the **competency** value is "model" the strategy might decide to present a conceptual feedback which elaborates the different models of the corresponding domain concept.

Depending on the difficulty of the **task** the strategy might decide whether a simple hint is enough or a complex hint sequence is necessary. Of course, this decision is based not only upon the metadata of the task, but also upon other parameters, such as the diagnosis of the student's answer, as well as the current state of student's knowledge of the concept.

### 3.3.2 Feedback

A **feedback** element represents the feedback to the previous action of the learner. This element can contain text with formulas and multimedia and formatting elements. It can be annotated with **metadata** classifying this feedback.

There has been extensive research on feedback and its usage in ITS systems and a variety of types of feedback have been used in various ITS systems.

[Van Lehn, 2006] classifies the following categories of feedback:

- minimal feedback: correct/incorrect notice
- flag feedback, highlighting error position
- error-specific feedback on an incorrect step
- procedural hints on what to do next

- conceptual hints providing information on involved knowledge components
- bottom-out hints or correct solutions for a problem or a problem step

Some systems research the effect of meta-cognitive feedback helping the student to guide his own problem solving by including, for example, aspects such as help-seeking [Roll et al., 2006] behavior, or self-explanation [Conati and VanLehn, 1999]. Other types of feedback related to motivation and other affective parameters of the student [du Boulay et al., 2008] have been recently addressed, which are, however, out of the scope of this thesis.

In the European project LEACTIVE MATH <sup>2</sup> the following classification of feedback was used.

The feedback has been divided into four categories: *procedural*, *conceptual*, *product* and *meta-cognitive*.

- *procedural* feedback tries to help the learner to proceed towards the solution. It contains information about what to do next, suggesting the concepts to be used or rules to be applied or reducing the task by decomposing it into sub-tasks or suggesting a similar simpler task.
- *conceptual* feedback represents the factual knowledge about focus concepts in the tasks and the connections between them.
- *product* feedback gives away (parts of) the solution.
- *meta-cognitive* feedback represents feedback and suggestions for the meta-cognitive behavior of the learner, such as help seeking behavior, self-assessment and self-explanation.

In [Narciss, 2008] and [Narciss, 2006] a rich classification of feedback components within the ITF (Informative Tutorial Feedback)-framework is given. This multidimensional classification is based on findings from systems theory and integrates some results of empirical research.

The ITF framework differentiates such dimensions of feedback as *function* (instructional goals), *content* and *form* (presentation).

We employ Narciss' classification of feedback for annotating **feedback** elements with metadata. The *function* of feedback and its *form* are used

---

<sup>2</sup><http://www.leactivemath.org>

as parameters of automated *tutorial* and *presentation* strategies respectively. These strategies are the subject of consequent sections.

We operationalize the following types of feedback, which represent different aspects of instructional context (in the brackets we specify the corresponding feedback category in the Van Lehn's classification above) :

- KP - knowledge of performance
- KR - knowledge of result (correct/incorrect notice)
- KCR -knowledge of correct result (bottom out hint or correct solution)
- KTC - knowledge of task constraints
- KC - knowledge of concepts (conceptual hints)
- KM - knowledge of mistakes (error-specific feedback)
- KH - knowledge on how to proceed (procedural hints on what to do next)
- KMC - knowledge on meta cognition

Below are some examples for each category of feedback, taken from [Narciss, 2008].

**Knowledge of performance [KP] :**

- 15 of 20 task correct
- 85% of the tasks correct

**Knowledge of result/response [KR] :**

- correct / incorrect
- 3 of 4 points (e.g. in multiple choice questions 3 of 4 choices correct)

**Knowledge of the correct result [KCR] :**

- description of the correct response
- indication of the correct response (e.g. for multiple choice questions)

**Knowledge on task constraints [KTC] :**

- Hints/explanations on type of task
- Hints/explanations on task processing rules
- Hints/explanations on subtasks
- Hints/explanations on task requirements

**Knowledge about concepts [KC] :**

- Hints/explanations on technical terms
- Examples illustrating the concept
- Hints/explanations on the conceptual context
- Hints/explanations on concept attributes
- Attribute-isolation examples

**Knowledge about mistakes [KM] :**

- Number of mistakes
- Location of mistakes
- Hints/explanations on type of errors
- Hints/explanations on sources of errors

**Knowledge on how to proceed [KH] ("know how") :**

- Bug related hints for error correction
- Hints/explanations on task-specific strategies
- Hints/explanations on task processing steps
- Guiding questions
- Work-out examples

**Knowledge on meta-cognition [KMC] :**

- Hints/explanations on meta-cognitive strategies
- Meta-cognitive guiding questions

We use some of these examples to derive sub-types for each of the feedback types introduced above.

Merging the newly defined classification of feedback content into more general feedback categories defined in `LEACTIVEMATH`, we group the newly defined dimensions within the corresponding category.

A summary of this classification is shown in the Table 3.2. Each type of feedback has a category assigned to it, together with the value and a possible sub-value.

In order to enable sequences of feedback with an increasing level of detail, we introduce a `depth` metadata element that can be applied to the feedback of all types.



| category       | value | subvalue                            | definition                         |
|----------------|-------|-------------------------------------|------------------------------------|
| procedural     | KTC   |                                     | knowledge of task constraints      |
|                |       | KTT                                 | knowledge of task type             |
|                |       | KTPR                                | knowledge of task processing rules |
|                |       | KST                                 | knowledge of sub-tasks             |
|                |       | KTR                                 | knowledge of task requirements     |
|                | KH    |                                     | know how to proceed                |
|                |       | ER                                  | error remedial                     |
|                |       | TSS                                 | task specific strategies           |
|                |       | TPS                                 | task processing steps              |
|                |       | GQ                                  | guiding questions                  |
|                |       | WOE                                 | worked out examples                |
| conceptual     | KC    |                                     | knowledge about concepts           |
|                |       | CD                                  | concept definition                 |
|                |       | CI                                  | concept-illustrating example       |
|                |       | CAD                                 | concept attribute definition       |
|                |       | AIE                                 | attribute isolation example        |
|                | KCC   | knowledge about concept connections |                                    |
| product        | KM    |                                     | knowledge of mistakes              |
|                |       | KNM                                 | knowledge of number of mistakes    |
|                |       | KML                                 | knowledge of mistake location      |
|                |       | KMT                                 | knowledge of mistake type          |
|                |       | KMO                                 | knowledge of mistake origin        |
|                | KR    | knowledge of result                 |                                    |
|                | KCR   | knowledge of correct result         |                                    |
| meta-cognitive | KMC   |                                     | knowledge of meta-cognition        |
|                | KP    |                                     | knowledge of performance           |

Table 3.2: Feedback classification in ACTIVEMATH exercises

Finally, each feedback can have a `cost` metadata, representing the penalty for the learner's mastery w.r.t. the focus concepts in the task. This information is important for properly updating the `Student Model`, since the estimation of learner's mastery depends on what kind of feedback and how much of it was given to the learner before he could accomplish the current task.

### 3.3.2.1 Usage of Feedback Metadata

Each exercise is a directed graph of nodes, which represent exercise states, and edges, which represent transitions between these states. A *tutorial strategy* is a transformation that is applied to the exercise graph and produces a new exercise derived from the original graph in which the new nodes and edges might appear, and the existing ones rearranged, or even deleted.

Such a transformation does not modify the meaning of the tasks in the exercise steps, but the way the learner can navigate through the solution space. This is independent of whether he can repeat the attempts to solve the task in case of failure, whether or not he receives feedback and hints of different types and in which order.

We define a rich set of annotations for feedback having the following two usage scenarios in mind:

1. To provide the author with a set of annotations that allow him to manually define a rich exercise solution space that can be reused with various tutorial strategies, where each strategy selects a subset of the given feedback types
2. To provide a rich vocabulary of feedback types that can be generated on the fly by an elaborate tutorial strategy using intelligent domain reasoning tools

Consider the following easy example of a feedback strategy that can be applied to an exercise:

- if the learner's input in a step is incorrect, then provide him with the feedback of a type 'Knowledge of Result' (KR) specifying that the answer is incorrect
- allow the student to repeat the step again

In order to apply such a strategy to an authored exercise, the feedback of the type KR should be authored in the exercise solution space for each exercise step.

Figure 3.5 shows how the graph of the easy exercise considered in the Figure 3.2 transforms when this strategy is applied to it.

The exercise on the left hand side only accepts correct answers in order to proceed to the next steps. In case of an incorrect answer, the student is directly forwarded to the terminal state, containing the feedback "You failed!".

The transformed exercise on the right hand side has to clone the "Incorrect!" feedback node as many times as there are steps in the solution, because each time this feedback is given the system forwards the student back to the corresponding solution step, thus giving him another chance.

From this example we can see that the strategy can add nodes and edges to the exercise graph, but it can also remove other nodes and edges. In the given example the nodes "Incorrect!" are added to each step together with transition edges from and back to the corresponding interaction nodes. On the other hand, the node "You failed!" is removed together with the transition edge from both interaction nodes.

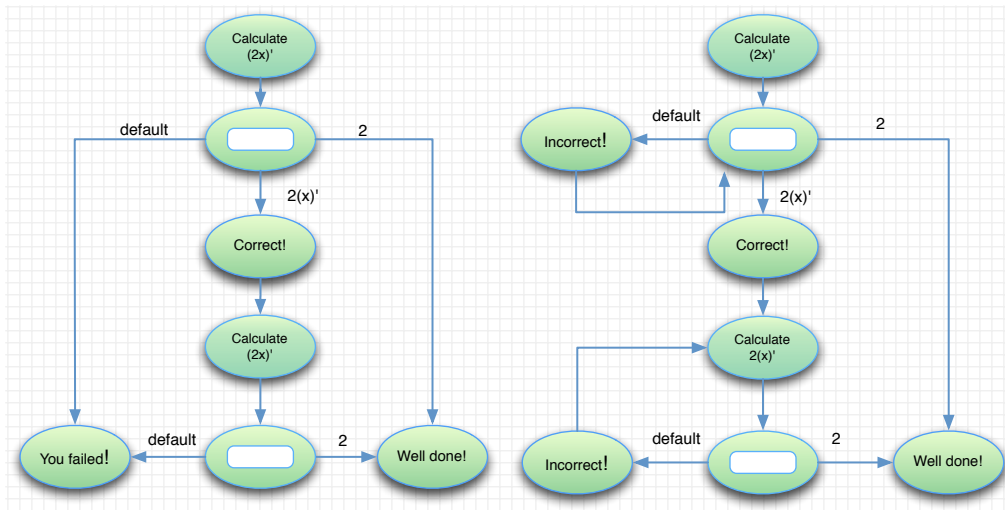


Figure 3.5: A simple exercise graph transformed by the strategy

The feedback annotations defined above are useful for elaborate tutorial strategies. Many of the feedback types can be generated automatically using

intelligent domain reasoning tools capable of diagnosis of the user actions. For example, the knowledge of result feedback (KR) can be generated knowing the task and the student's answer, the KCR (correct solution) can also be generated by the generative domain reasoner for the given task. Different kinds of feedback on mistakes (KM) can be generated: the location of mistakes in the student's answer (KML) as well as more specific error feedback if the domain reasoner encodes buggy rules.

The exercise player in *ACTIVEMATH* has a built-in mechanism for generating KR feedback, in case at least one correct answer is provided in one of the transitions originating at the given interaction. This mechanism also allows (in a restricted number of cases) to generate feedback of types KNM (knowledge of number of mistakes) and KML (knowledge of mistake location). The restriction is that in such an interaction with multiple input fields there should be only one way to enter a correct solution. This should be taken as a reference to compare the student's input.

Knowledge on how to proceed (KH) can be generated on different levels, starting from the names of concepts (rules) to be applied for the given step to actually generating the next step.

Conceptual hints can be generated by extracting definitions or explanations of participating concepts from the static content database. Examples of automatically generated feedback using domain reasoners will be given in the subsequent Sections.

Feedback about the student's performance (KP) can be generated by the system with the help of the local history of student's actions and their diagnosis, stored in the system.

Meta-cognitive feedback (KMC) can be generated on the fly using the local history of student's interactions with the system. This can be information about the usage of hints by the student, number of trials of the single steps, learning time for interactions of different types and other data about meta-cognitive behavior of the student.

Some examples of meta-cognitive behavior of the student and corresponding feedback that can be automatically generated by the system are shown in the Table 3.3

The local history of the exercise process which records the relevant information about the student's performance is stored in the **Local Student Model** component, as discussed in Section 4.6

| Observable Behaviour   | Generated Feedback   |
|--|--|
| Student always asks for hints directly, without even trying to solve the task          | Please, try to solve the problem first and ask for a hint only if you are in trouble |
| Student tries to solve the task many times without success and is not asking for hints | Perhaps asking for a hint now will give you an insight on how to proceed             |
| Student asks for many hints in a row and then submits the correct answer               | Consider trying to answer after each hint, maybe you do not need so much assistance  |
| Student thinks too long before providing the answer to the task                        | You might consider asking for a hint or trying an easier problem first               |

Table 3.3: Generating meta-cognitive feedback from observable student behavior

### 3.3.3 Interaction

An **interaction** element represents the actual interaction with the learner. It may contain some text, formulas, and placeholders for different types of interactive elements. These placeholders can be placed anywhere in the content and, in particular, inside the formulas.

A special element called **interaction.map** groups all the interactive elements and maps them to the placeholders in the content. This separation of content and interactive elements has the purpose to give easy access to interactive elements in order to change their style or content following a particular tutorial or presentation strategy. Examples of different presentation for **blank** and **selection** are shown in Figures 3.11 and 3.13.

We differentiate between two fundamental types of interactive elements - **blanks** and **selections**. A **blank** is an interactive element that represents an interaction with the learner in which the learner has to enter information into one or more input fields. As opposed to the **blank**, a **selection** represents an interaction with a learner in which the learner has to manipulate objects that are already presented to him on the screen. Here the learner selects one or more objects, groups the objects in some particular order, or forms several groups of objects. We call this process **selection** as the result of such manipulation consists of a list of (lists of) selected objects.

The set of possible types and presentation styles of these interactive ele-

ments is shown in the Table 3.4.

| Name      | Type                | Style   | Definition   |
|-----------|---------------------|---|--|
| blank     | literal             |   | literal comparison only                                    |
|           | formula             |   | allows for semantic evaluation                             |
|           | item_reference      |   | link to the concept from the knowledge base                |
| selection | mcq_single_answer   | radio_button_inline<br>radio_button_block<br>marking<br>drop_down | multiple choice questions with only one possible answer    |
|           | mcq_multiple_answer | check_box_inline<br>check_box_block<br>marking                    | multiple choice questions with one or more possible answer |
|           | mapping             | table<br>graphical  | mapping two lists of objects                               |
|           | ordering            |   | reordering a list of objects                               |

Table 3.4: Forms and presentation styles of interactive elements

### 3.3.3.1 Selections

**Selection** represents the set of **choice** elements one or more of which have to be selected by the learner. There are several types of selections: *single-answer* in which only one choice can be selected, *multiple-answer* selections in which more than one choice can be selected, as well as orderings and mappings. These interactive elements might be rendered in various ways depending on the learner profile and the tutorial strategy. Presentation ranges from simple check-boxes and radio buttons to complex graphical puzzles.

### 3.3.3.2 Multiple Choice Questions

There can be two types of multiple choice questions - those in which the user can select only one option and those in which several options can be selected.

The presentation of a simple multiple choice question block is shown in Figure 3.6. In this case, only one answer can be ticked, and a radio button is used, in case of multiple answers check boxes are used.

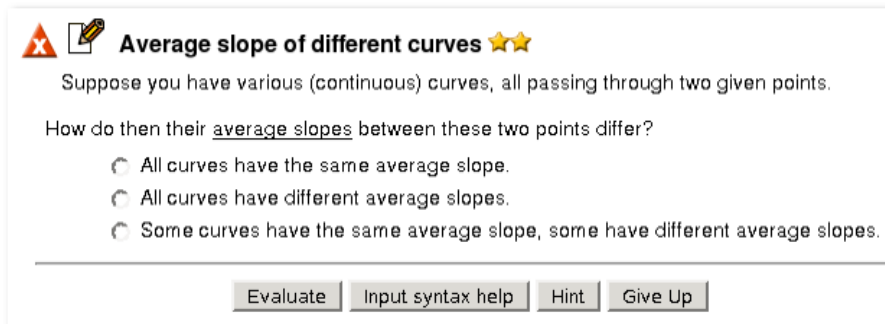


Figure 3.6: A presentation of a multiple choice in ACTIVE MATH

Figure 3.7 shows a representation of the multiple choice questions from the Figure 3.6.

```

<task id="task1">
  <content><CMP>
    Suppose you have various (continuous)
    curves, all passing through two given points.
  </CMP></content>
  <transition to="interaction1"/>
</task>
<interaction id="interaction1">
  <content><CMP>
    How do then their average slopes between these
    two points differ? <with xref="selection1"/>
  </CMP></content>
  <interaction_map>
    <selection type="mcq_single_answer" id="selection1">
      <choice>
        <CMP>All curves have the same average slope.</CMP>
      </choice>
      <choice>
        <CMP>All curves have different average slopes.</CMP>
      </choice>
      <choice>
        <CMP>Some curves have the same average slope,
        some have different average slopes.</CMP>
      </choice>
    </selection>
  </interaction_map> ...
</interaction>

```

Figure 3.7: Representation of multiple choice questions

The `interaction` node contains a placeholder element, called `with`. This

element has a reference to the `selection` element that is placed inside the `interaction_map`. The value 'mcq\_single\_answer' for the `type` attribute stands for multiple-choice questions with single answer.

### 3.3.3.3 Orderings

Consider the type of selection, where the learner has to order the elements in a list according to some criteria, as defined in the task of the exercise step. Figure 3.8 shows the representation of a sample ordering.

```

<interaction id="step1">
  <content>
    <CMP>Arrange the steps of a proof by induction
      in the right order. <with xref="ordering1"/></CMP>
  </content>
  <interaction_map>
    <selection type="ordering" id="ordering1">
      <choice>
        <CMP> Induction hypothesis: assume
          that the statement holds for some  $n > 1$  </CMP>
      </choice>
      <choice>
        <CMP> Anchor of induction: prove
          that the statement holds for  $n = 1$  </CMP>
      </choice>
      <choice>
        <CMP> Inductive step: prove that
          the statement holds for  $n + 1$  </CMP>
      </choice>
    </selection>
  </interaction_map>
</interaction>

```

Figure 3.8: Sample ordering

The corresponding rendering of this ordering task in `ACTIVEMATH` is shown in Figure 3.9. The choices are presented within boxes that can be dragged by the student for changing their order.

### 3.3.3.4 Mappings

Now, consider a more complex selection, in which the learner has to provide a mapping between two selections. For example, he is given a list of functions and the list of their derivatives and he has to link the functions from the first list to their derivatives in the second list. The representation of this mapping is shown in Figure 3.10.



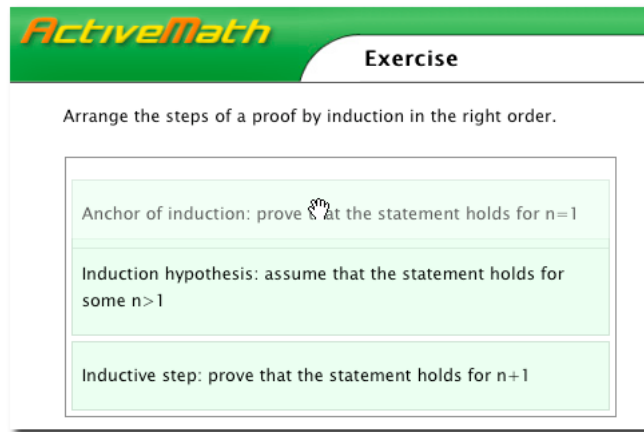


Figure 3.9: The presentation of an ordering exercise in ACTIVE MATH

```

<interaction>
  <content>
    <CMP>Pair the formulas in the left with their derivatives
      in the right, by reordering the list at the right side:
    <with xref="mapping1" /></CMP>
  </content>
  <interaction_map>
    <selection type="mapping" style="graphical" id="mapping1">
      <selection style="fixed">
        <choice><CMP>(f(x) + g(x))'</CMP></choice>
        <choice><CMP>(x^n)'</CMP></choice>
        <choice><CMP>(-cos(x) + C)'</CMP></choice>
      </selection>
      <selection>
        <choice><CMP>sin(x)</CMP></choice>
        <choice><CMP>n · x^{n-1} </CMP></choice>
        <choice><CMP>f'(x) + g'(x)</CMP></choice>
      </selection>
    </selection>
  </interaction_map>
</interaction>

```

Figure 3.10: Example of a mapping

The mapping consists of two selections that are mapped onto each other. There are two ways to present a mapping - using draggable boxes, and using a table, as shown in Figure 3.11 .

The figure shows two versions of an 'ActiveMath Exercise' interface. Both have a green header with the 'ActiveMath' logo and the word 'Exercise'.

**Left Interface (Draggable Boxes):**  
 Instruction: "Pair the formulas in the left with their derivatives in the right, by reordering the list at the right side:"  
 Left column (fixed):  $(f(x)+g(x))'$ ,  $(x^n)'$ ,  $((-\cos x)+C)'$   
 Right column (reorderable):  $\sin x$ ,  $n \cdot x^{n-1}$ ,  $f(x)'+g(x)'$   
 Buttons: Evaluate, Give Up

**Right Interface (Table):**  
 Instruction: "Pair the formulas in the left with their derivatives in the top of the table, by marking the intersections for the correct combinations:"  
 Table:  

|                  | $\sin x$                            | $n \cdot x^{n-1}$                   | $f(x)'+g(x)'$                       |
|------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| $(f(x)+g(x))'$   | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |
| $(x^n)'$         | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| $((-\cos x)+C)'$ | <input type="checkbox"/>            | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |

 Buttons: Evaluate, Give Up

Figure 3.11: Presentation of a mapping exercise

In the presentation with draggable boxes, the elements of the first list are fixed and the elements of the second list can be reordered by dragging. Such a presentation is possible only for one-to-one mappings in which the number of elements in both selections are equal. The presentation as a table can be used also for the cases where the number of choices in the first and second selection are not equal or the mapping is not one-to-one, which means that more than one element from the second list can correspond to one element from the first list, and there can be elements from the second list that do not correspond to any element from the first list.

### 3.3.3.5 Fill-in-blank

Another example of an **interaction** element containing fill-in-blank fields is shown in Figure 3.12.

The interactive element **blank** represents an input field in which the learner can enter the answer to the task in an exercise step. We differentiate three types of **blank** input. Free text that has to be literally compared to the reference value is represented by the **blank** of a type 'literal'. Formal input that can be interpreted as an OPENMATH expression is represented by the **blank** of the type 'formula' (default value). Finally, a reference to

```

<task id="task1">
  <content>
    <CMP> Use the Chain Rule to find the derivative of the
      following function:  $y = 1/(1 - 2x)^2$  with respect to  $x$ .</CMP>
  </content>
  <transition to="interaction1" />
</task>
<interaction id="interaction1">
  <content><CMP>
    <OMOBI><OMA>
      <OMS cd="relation1" name="eq" />
      <OMV name="y" />
      <OMV name="blank" xref="outer_layer" />
    </OMA></OMOBI>
    <OMOBI><OMA>
      <OMS cd="relation1" name="eq" />
      <OMV name="u" />
      <OMV name="blank" xref="inner_layer" />
    </OMA></OMOBI>
  </CMP></content>
  <interaction_map>
    <blank id="outer_layer" /><blank id="inner_layer" />
  </interaction_map> ...
</interaction>

```

Figure 3.12: Representation of fill-in-blank

the domain concept from the Knowledge Base is represented by the **blank** of type 'item\_reference'.

The representation of a **blank** element does not define the way it is presented. Depending on the strategy and the preferences of the learner, different presentations and user interfaces for the **blank** can be chosen. Figure 3.13 shows two different presentations for the same complex formula with blanks whose representation is shown in Figure 3.12. In the first case the student has to use a structured linear markup to enter formulas, in the second case a palette-based editor is invoked to assist formula editing.

The presentation style of the **blank** element can be manually assigned in the content of the exercise, as well as automatically assigned on the fly by a presentation strategy. The set of values of the *style* attribute is not fixed and can be customized.

More complex presentation strategies for **blank** elements are presented in Section 4.8.

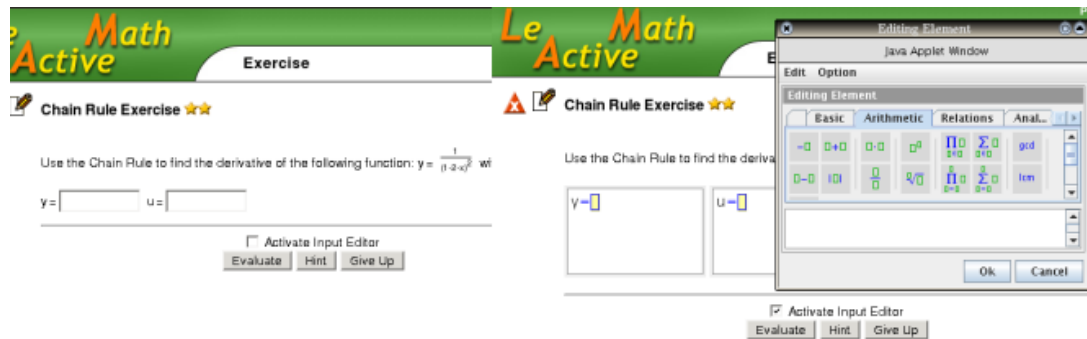


Figure 3.13: Different presentations of blanks in ACTIVE MATH

## 3.4 Transitions Between Exercise States

Each state of an exercise can contain a set of transitions pointing to further steps. These transitions can be *conditional* and *unconditional*. Each transition contains a pointer to the target exercise state.

A Conditional transition contains a **condition** to be satisfied for the learner's answer and a corresponding **diagnosis**. An unconditional transition can be originated only by a non-interactive state and contains only a pointer to the target state.

In order to avoid conflicts in case the learner's input matches conditions of several transitions, they have to be ordered by the author.

Transitions are evaluated by the system in the order of their occurrence in the transition map. Therefore, it is the responsibility of the author of an exercise to order transitions in such a way that each transition can be possibly reached by the system. If the set of conditional transitions in an interaction is not covering all possible user inputs, a special transition is authored at the end - called *default* transition. This transition does not have any condition and fires in case when the user input is not matched by any other transition.

### 3.4.1 Diagnosis

In order to have information about the learner's progress, we need to annotate each possible action of the learner with some diagnosis information. This information is placed inside a **diagnosis** element.

The success rate of the learner's answer w.r.t. the task of an exercise state

is represented by an element called **achievement** with values ranging from 0.0 to 1.0. This success rate is used for several purposes. First, the **achievement** defines a weight for updating the learner's competency values in the **Student Model**. Secondly, the **achievement** value for each step is stored in the **Local Student Model** (see Section 4.6) and used by the tutorial strategies.

Some transitions may match specific incorrect answers of the learner. These are frequent procedural and conceptual errors (CAPEs). There can be two types of CAPEs in **ACTIVEMATH** – **misconceptions** and **buggy rules**. Buggy rules mostly indicate the application of an incorrect rule or the incorrect application of an expert rule. Misconceptions refer to developmental errors that happen, for example, in cases where the new knowledge to be acquired comes in conflict with what is already known.

CAPEs are represented as items in the knowledge base and are annotated with the relationship to the corresponding concept and with the key competencies addressed by the CAPE. For more information about CAPEs in **ACTIVEMATH** see [Melis and Gogvadze, 2006].

If a CAPE can be diagnosed by an exercise, the relation from the exercise to the corresponding CAPE has to be provided in two places: in the metadata of the exercise (used by the **Tutorial Component** for choosing this exercise) and in the **diagnosis** element in a transition matching the CAPE. This information will be used by the **Student Model** to propagate the corresponding penalty to the learner's mastery computation of the related concept. In order to refer to a CAPE the **relation** element with the type 'diagnoses' is used. In fact, the relation to a CAPE in the metadata can be generated automatically, given it is provided in a **diagnosis** of some transition within the exercise. It is only needed as a top level metadata, since the **Tutorial Component** does not look inside the content of the learning objects when selecting them, but only checks their metadata.

Figure 3.14 shows an excerpt from an ontology of the domain of fractions, in which the CAPEs are linked to the corresponding concepts using a relation of a type 'of'. The exercises that diagnose these CAPES have a relation of a type 'for' to the corresponding domain concepts and a relation of a type 'diagnoses' to the CAPEs they diagnose.

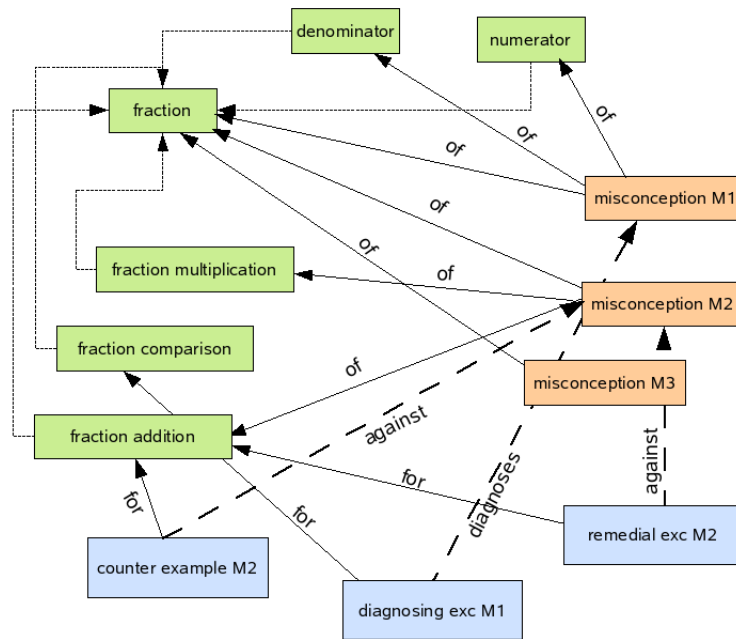


Figure 3.14: Excerpt from an ontology of the fractions domain.

The examples and exercises that can help to remediate the CAPEs are linked to the CAPEs using a relation of a type 'against'.

Figure 3.15 shows the representation of the misconception M1, in which the learner considers the numerator and denominator as independent parts of a fraction. This element refers to the corresponding concepts of the fractions domain using 'of' relations.

```
<cape id="M1" type="misconception">
  <metadata>
    <relation type="of"><ref xref="fractions/fraction"/></relation>
    <relation type="of"><ref xref="fractions/numerator"/></relation>
    <relation type="of"><ref xref="fractions/denominator"/></relation>
  </metadata>
  <CMP> Numerator and denominator are
    independent parts of a fraction. </CMP>
</cape>
```

Figure 3.15: Representation of the misconception M1

Figure 3.16 shows the source of the exercise in which the transition matching the particular wrong answer or the learner has a relation of a type 'di-

agnoses' pointing to the identifier of the corresponding misconception. The exercise also encodes the 'for' relation to the concept of fraction comparison.

```
<exercise id="diagnosing_exc_M1">
  <metadata>
    <relation type="for"><ref xref="fractions/comparison"/></relation>
    <relation type="diagnoses"><ref xref="fractions/M1"/></relation>
  </metadata>
  <interaction id="interaction1">
    ... Which comparison is correct? ...
    <interaction_map>
      <selection>
        <choice>3/5 < 3/7</choice>
        <choice>3/5 = 3/7</choice>
        <choice>3/5 > 3/7</choice>
      </selection>
    </interaction_map>
    <transition_map>
      <transition to="choice1_wrong">
        <diagnosis>
          <relation type="diagnoses"><ref xref="fractions/M1"/></relation>
        </diagnosis>
        <condition> ... </condition>
      </transition>
    </transition_map>
  </interaction>
</exercise>
```

Figure 3.16: Exercise diagnosing the misconception M1

Since it is not always possible, to match all possible answers of the learner, it is recommended to define a default transition to be triggered in case none of the conditional transitions match the learner's answer. If such a transition is not defined, and the learner's answer does not match any of the defined transitions, the finite state machine of the exercise just stays in the same state and the action of the learner is ignored.

In case of an incorrect answer or in case of a hint request, the diagnosis can explicitly encode the average penalty value for the competencies involved in the learner's answer. The penalty is represented using the `cost` element and has float values from 0.0 to 1.0.

### 3.4.2 Condition

An element `condition` represents conditions that have to be satisfied by the learner's answer. The `condition` consists of one or more `compare` elements

that represent comparisons of the learner's answer to the reference values. Comparisons can be *simple* and *complex* in structure.

**Simple comparisons.** Simple comparisons are represented using a `compare` predicate containing the reference value to be compared to the learner's answer. The `compare` predicate can be extended with a *context* attribute that defines the mathematical evaluation context, as described in more detail in Section 3.4.4. There are three basic contexts defined for all domains of exercises: *syntactic*, *numeric* and *semantic*. In case of a *syntactic* context, the learner's answer is literally compared to the reference value. A *numeric* comparison is used for comparing real numbers modulo some epsilon, represented as an attribute of the `compare` element. A *semantic* comparison is used to check whether the learner's answer and the reference value are semantically equivalent. Such an equivalence can be validated by a Computer Algebra System, employed as the semantic evaluation by ACTIVE MATH. If the *context* attribute is omitted, a *syntactic* context is used by default.

In case the learner has to submit multiple inputs to the system in one step, e.g. an answer to multiple choice questions or to fill in several blanks, a grouping element `composite` is used as a container of single comparison elements. A `composite` condition is considered satisfied if and only if all component comparisons are satisfied. In case it is not important to evaluate all components of a `composite` comparison but only some of them, the `composite` element can be annotated with a *map* attribute, containing a list of positions of single inputs that have to be evaluated. The rest of inputs will then not be evaluated by the system.

If the learner's answer consists of several elements, they are compared to the list of reference values in the order of their occurrence. If the order is not important, the value of the attribute *ordered* of the element `composite` is set to "false". The default value of this attribute is "true", which is used by the system, if the attribute is omitted in the representation.

The order of `transitions` in the `transition_map` is important when evaluating the same expression successively in different contexts. For example, if the first transition contained a `condition` comparing the learner's answer to the reference value semantically and the second transition contains a `condition` comparing the learner's answer to the reference value syntactically, then the second transition would never be reached by the system. However, assume a syntactic comparison transition is followed by a semantic compar-



ison transition. Then the second transition fires with the learner's answer only if the condition of the first transition is not satisfied. Therefore, if the second transition fires, we already know that the first transition did not fire, so the learner's answer is not syntactically equal to the correct solution, but semantically equivalent.

**Complex constraints.** In case of more complex constraints upon the learner's answer, the `composite` element can contain a single expression representing a complex constraint upon the learner's answer. It is represented as a mathematical formula (using the `OPENMATH` format as for any mathematical expression in `ACTIVEMATH`), possibly consisting of a Boolean combination of atomic predicates. An example of a complex constraint is shown in Figure 3.22.

#### 3.4.2.1 Evaluating Multiple Choice Questions

Conditions for evaluating multiple choice questions are represented by composite comparisons, containing as many elements, as there are choices in the selection. Each atomic comparison compares the learner's answer for the current choice to 1 if the choice has to be selected and to 0 otherwise. Alternatively, it is possible to represent more complex constraints consisting of Boolean combination of atomic predicates, as it is shown in Section 3.4.2.4 for blanks.

Consider the example of a multiple choice from the Figure 3.7. The condition matching the correct answer is shown in the Figure 3.17. Here, the first and the third choice are selected, and the second choice is left unchecked.

```
<condition>
  <composite>
    <compare>1</compare>
    <compare>0</compare>
    <compare>1</compare>
  </composite>
</condition>
```

Figure 3.17: Condition for the correct answer in a multiple choice

### 3.4.2.2 Evaluating Mappings

Mappings can be modeled as two-dimensional selections. In case of a mapping with dimensions -  $m \times n$ , the number of atomic comparisons will be  $m \cdot n$ .

Consider the case of a mapping exercise from Figure 3.10. Figure 3.18 shows the matrix of values matching the learner's answer in this mapping.

The screenshot shows an "Exercise" window with the following table:

|                  | $\sin x$                            | $n \cdot x^{n-1}$                   | $f(x)' + g(x)'$                     |
|------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| $(f(x)+g(x))'$   | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |
| $(x^n)'$         | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| $(-\cos x + C)'$ | <input type="checkbox"/>            | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |

An arrow points from this table to the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 3.18: Evaluation matrix for a mapping

The corresponding composite comparison is shown in Figure 3.19. The matrix is entered as a flat list of zeros and ones.

```
<condition>
<composite>
  <compare>1</compare><compare>0</compare><compare>0</compare>
  <compare>0</compare><compare>1</compare><compare>0</compare>
  <compare>0</compare><compare>0</compare><compare>1</compare>
</composite>
</condition>
```

Figure 3.19: Condition matching a mapping answer

### 3.4.2.3 Evaluating Ordering

A special case of a mapping exercise is an ordering exercise. In such exercises the list of choices is mapped to itself. The correct answer to the ordering from the Figure 3.9 is to reorder the first and the second choices.

Therefore, we need a condition representing mapping of choices from the original order 1,2,3 into 2,1,3. In our representation of mapping comparisons, this transformation is represented by a matrix:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 3.20 shows a representation of the corresponding condition.

```
<condition>
  <composite>
    <compare>0</compare><compare>1</compare><compare>0</compare>
    <compare>1</compare><compare>0</compare><compare>0</compare>
    <compare>0</compare><compare>0</compare><compare>1</compare>
  </composite>
</condition>
```

Figure 3.20: Condition for the correct answer in an ordering

#### 3.4.2.4 Evaluating Blanks

In order to evaluate blanks, the `compare` element is used as well. This element can have a `context` attribute, specifying the type of comparison, as described in Section 3.4.4. If the learner has to fill in several blanks in one step, that have to be evaluated independently from each other, the corresponding `condition` element contains a `composite` grouping element that consists of a list of comparisons, each corresponding to the respective blank. A sample condition is shown in Figure 3.21.

```
<condition>
  <composite>
    <compare context="syntactic">1</compare>
    <compare context="syntactic">2</compare>
    <compare context="semantic">1 + 2</compare>
  </composite>
</condition>
```

Figure 3.21: Simple condition for multiple free inputs

In this condition, the first input of the learner is syntactically compared to number 1, the second input is syntactically compared to the number 2, and the third input is semantically compared to  $1 + 2$ .

In order to allow for an arbitrarily complex evaluation of multiple free inputs, any constraint involving the learner's answers can be used. Figure 3.22 shows a sample condition, in which the first input of the learner can be arbitrary, but the second input must be twice the first one and the third input must be three times the first one.

```

<condition>
  <composite>
    <compare context="semantic">
       $\lambda x, y, z. (y = 2 \cdot x) \& (z = 3 \cdot x),$ 
      true
    </compare>
  </composite>
</condition>

```

Figure 3.22: Complex Boolean condition for multiple free inputs

We use a lambda expression for representing this constraints, in which the bound variables  $x, y$  and  $z$  are automatically mapped by the system to the corresponding learner's answers. The result of the expression which involves the learner's answers is compared with the reference result. In our case, the result is Boolean and it is compared to *true*.

### 3.4.3 Evaluation using a CAS

In order to compute the diagnosis of the learner's answer one has to formulate a query to an intelligent mathematical service. Computer Algebra Systems are among such services used by ACTIVEMATH [Büdenbender et al., 2002a]. As we see below, these systems are generally not sufficient to provide feedback according to human like solution steps, but they are still useful for a quick correct/incorrect evaluation.

The most frequent query is to check the correctness of the learner's answer by comparing it semantically to the task expression. In order to formulate such a query we need to have the answer of the learner and one of the following:

- a task to be performed by the learner

- an expected correct answer and the way to compare it to the given answer

This information is a way to represent the mathematical *context* of the task. In general such a context specifies which domain specific production rules have to be applied to the task.

Computer Algebra Systems use various commands in order to specify what to do with a mathematical expression. The most frequently used ones are different variants of 'evaluate', 'simplify' and 'solve'. The mathematical *context* of such a task is sometimes fixed for a particular command, or can be passed to the command as a parameter.

Surprisingly, Computer Algebra Systems develop no systematic investigation to the mathematical *context* within which the expression has to be manipulated. CAS commands sometimes use additional arguments to specify a fixed context only. The design of commands is not homogenous, arguments of CAS commands that deal with semantics of mathematical context are mixed with other parameters that need to be introduced due to technical limitations of the system. Thus it is necessary for the user to learn the syntax and programming constructs of each particular CAS.

There are historical and practical reasons for such design decisions in CAS. Computer Algebra Systems were not designed for remote use as back engines for an intelligent tutoring system. They were designed to be used as mathematical tools, directly interacting with the human user. Irregularities within the usage of different commands, notational overload and the preference for many commands with simple a structure are due to facilitate an easy and direct interaction with the human user.

In *ACTIVEMATH*, it is necessary to restrict the evaluation of the learner's answer to a particular context in order to provide a proper diagnosis of his answer. For example, we need to detect if the learner just repeats the problem statement rather than to give a correct answer, or makes a mathematically valid transformation, that does not get him closer to the solution. But a Computer Algebra System does not have any means to trace the solution process in a stepwise manner.

Consider an example: the task is to differentiate a function, and the system needs to check whether the learner has already applied all the differentiation rules, such that only some algebraic simplification is left to be applied. In order to do this, the learner's answer has to be semantically compared to the correct solution without applying differentiation rules. So the

CAS has to be blocked for the application of differentiation. In some CASs there is a possibility to program such context checks.

Figure 3.23 shows a screen shot of Maxima, where an expression with a differentiation is evaluated. In step (%i1) Maxima only simplifies common members and expands brackets, without applying differentiation. In the step (%i2) the system also performs differentiation.

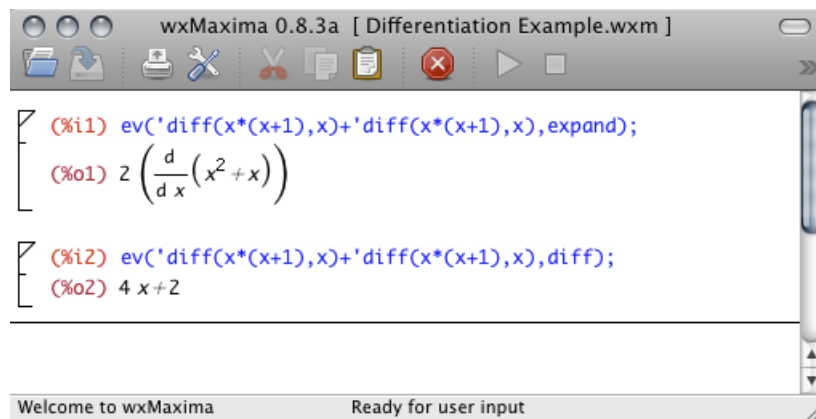


Figure 3.23: Evaluation in different contexts in Maxima CAS

In this particular case computing the differentiation or just expanding w.r.t. the arithmetics rules is triggered by the second argument of the command 'ev'. This is one of several variants of the 'evaluate' command of Maxima. The values 'expand' and 'diff' of this parameter represent the mathematical context of the task, and the 'ev' command itself represents a type of *action* to be performed.

Since CASs are external for ACTIVEMATH and the availability of them is changing with time, there is a need to be able to connect to several different CASs. In order to be able to use multiple CASs as back-engines for providing a diagnosis of the learners answer, there is the need to introduce a generic notion of a CAS query that abstracts from concrete CAS-dependent commands. The back and forth mapping between this generic format and the syntax of the concrete CAS has to be provided via a so-called phrasebook, which consists of transformations that perform such a mapping. See [Cohen et al., 2005a] for a description of phrasebooks realized within the LE-ACTIVEMATH project.

### 3.4.4 Context Evaluation

As introduced in Section 1.1 above, under a *context* of the task we understand a reference to the set of mathematical expressions that represent either rewriting rules or operations that can be applied to transform a mathematical expression. In Computer Algebra Systems, the context evaluation is usually realized by restricting the amount of operations to be applied to the expression by explicitly forbidding some of them. In the case of a CAS one can not always specify the particular derivation rules that can be allowed for a CAS to use. In general, Computer Algebra Systems are pretty unsystematic with respect to restricting the evaluation of a mathematical expression and they do not provide a generic way of doing so.

Another problem with CAS evaluation is that different CASs have different notions of simplification and provide different solutions to the same problem. Many problems are caused by differences in notation. Unless the participating systems use some form of a semantic knowledge representation for formulas, such notational confusions might have as a consequence the wrong evaluation of mathematical expressions [Davenport, 2008].

Specifying the mathematical *context*, independently of any particular CAS and making sure they are understood by all CASs is a step towards solving the interoperability problem.

**Examples.** OPENMATH content dictionaries are designed in such a way that the operations that belong to the same mathematical theory are grouped within one content dictionary. So, sometimes it is adequate to use content dictionaries as contexts.

For example, we can define a context 'arith1' that describes the context of an arithmetic transformation as the list of symbols from the 'arith1' content dictionary <sup>3</sup>:

```
'arith1' := { "lcm", "gcd", "plus", "minus", "unary_minus", "times", "divide",
"power", "abs", "sum", "product" }
```

Another content dictionary 'transc1'<sup>4</sup> that defines transcendental functions can be used as the appropriate context:

```
'transc1' := { "log", "ln", "exp", "sin", "cos", "tan", "sec", "csc", "cot",
"sinh", "cosh", "tanh", "sech", "csch", "coth", "arcsin", "arccos", "arctan", "arcsec", "arccsc", "arccot", "arccosh", "arctanh", "arcsech", "arccsch", "arccoth" }
```

<sup>3</sup>see <http://www.openmath.org/cd/arith1.oed>

<sup>4</sup>see <http://www.openmath.org/cd/transc1.oed>

Then we can build a context 'elementary\_functions' by unifying 'arith1' and 'transcl'.

'elementary\_functions' := 'arith1'  $\cup$  'transcl'

One can also specify the context by defining rewrite rules in the format of OPENMATH objects. The rewrite rules can be introduced by specifying their symbol name or by giving formal OPENMATH representation of the rule.

For example the context 'diff\_rules' can be defined as:

'diff\_rules' := {"const\_term\_rule", "const\_mult\_rule", "lin\_term\_rule", "sum\_rule", "prod\_rule", "power\_rule", "exp\_rule", "exp\_function\_rule", "quot\_rule", "chain\_rule", "sin\_rule", "cos\_rule", "tan\_rule", "arcsin\_rule", "arccos\_rule", "arctan\_rule"}.

where, for instance, the "sum\_rule" can be represented as

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMS cd="derivation_rules" name="sum_rule" />
</OMOBJ>
```

or as

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMA>
    <OMS cd="relation1" name="eq" />
    <OMA>
      <OMS cd="calculus1" name="diff" />
      <OMA>
        <OMS cd="arith1" name="plus" />
        <OMV name="f" /><OMV name="g" />
      </OMA>
    </OMA>
  </OMA>
  <OMA>
    <OMS cd="arith1" name="plus" />
    <OMA>
      <OMS cd="calculus1" name="diff" /><OMV name="f" />
    </OMA>
  </OMA>
  <OMA>
    <OMS cd="calculus1" name="diff" /><OMV name="g" />
  </OMA>
</OMOBJ>
```

A context can be referred to by a single identifier (e.g., 'arith1' as above) which is supposed to be known in the phrasebook. The names of these single functions or rules have to be mapped from OPENMATH to the syntax of the external system and back. The set of available contexts with their description should be made available for exercise authors.



### 3.4.5 Generic Query Format

Based on the research conducted within the European Project LEACTIVE-MATH [Goguadze, 2005], a format for queries to intelligent mathematical services has been developed by the author, in which a query consists of the following parts:

- **action** - an abstract representation of the type of action to be applied to the mathematical expression(s)
- a (list of) expression(s) in OPENMATH format, the input expressions of the query
- **context** (or a list of contexts) describing the mathematical operations that should be used
- an (optional) number of iterations to be used by the reasoning system (only applicable to rule-based systems)

Providing a generic classification of types of actions that an abstract intelligent reasoning service might perform and defining hierarchies of contexts is beyond the scope of this work, although this would be beneficial.

Instead, we concentrate on queries that ACTIVEMATH needs in order to provide a sensible diagnosis of the learner's answer.

Currently two main CAS queries that we use are 'evaluate' and 'compare'. The architecture of the exercise sub-system of ACTIVEMATH allows to keep this set of queries extensible. We have defined a large set of queries for the communication with external domain reasoners, that are capable of providing a more detailed diagnosis of the learner's answer which we described in Section 3.4.8.

The set of *contexts* is also extensible. Teachers can define their own contexts that consist of a formal representation of rewrite rules for mathematical expressions. The generic rule-based engine, connected to the ACTIVEMATH system is able to employ teacher-defined rules for defining the context of evaluation. This engine is a customized version of the YACAS Computer Algebra System, which will be discussed in more detail in Section 4.3.5.

In order to show the usage of the **context** parameter of a query, consider the query 'evaluate', used in the example (%i1) as shown in Figure 3.23 and the expected results of this query in two different contexts - 'arith1'

and 'diff\_rules', representing arithmetic simplification and the set of rules for calculating derivatives respectively.

Given the expression

$$f(x) = x \cdot (x + 1) \quad (3.1)$$

The query

$$query('evaluate', \frac{d}{dx}(x(x + 1)), 'arith1')$$

should result in

$$\frac{d}{dx}(x^2 + x)$$

since for this expression only the arithmetic transformation rules are applied, and the query

$$query('evaluate', \frac{d}{dx}(x(x + 1)), 'diff_rules')$$

should result in

$$1 \cdot (x + 1) + x \cdot (1 + 0)$$

since all the differentiation rules were applied, but no further arithmetic simplifications follow.

If we use the union of the two contexts 'diff\_arith'='arith1'  $\cup$  'diff\_rules', the query

$$query('evaluate', \frac{d}{dx}(x(x + 1)), 'diff_arith')$$

results in

$$2 \cdot x + 1$$

The 'evaluate' query is mostly issued while running an interactive exercise, if the system needs to verify whether the learner's input satisfies the conditions provided in the current exercise step. Using this query the system can compute a specific diagnosis of the learner's action in the exercise even if the correct answer to each step is not specified in the exercise content.

Suppose, we know the task  $T$  of the learner, the learner's input  $I$  and the context  $C$  of the task. By issuing a query 'evaluate' one obtains the correct result for this task  $R = query('evaluate', T, C)$ . Then, the  $query('evaluate', (I = R), C)$  returns the correctness status of the learner input w.r.t. the task  $T$ . The latter query is a special case of the Boolean query 'compare', that is used to check for equivalence of two expressions in a given context. This query is represented as follows:  $query('compare', (I, R), C)$ . In general however, an equivalence is not necessarily represented by an arithmetic equality, but it is specified by the *context*. For example, if the evaluation *context* is 'propositional\_logic', the equivalence means the equivalence of two formulas of propositional logic.

### 3.4.6 Interoperability between different CASs

Since the format of a query does not depend on the syntax of a particular Computer Algebra System, such queries can be reused by the system when there are different CASs available. For each new CAS that is connected to the ACTIVEMATH system, a new dedicated service can be defined, as described in Section 4.3.2. This service prepares the query before being sent to the CAS, so the service includes among others a phrasebook that translates the query to the format of the CAS and also translates the resulting expression from the CAS back into the OPENMATH format.

In the LEACTIVEMATH project, a connection to several CASs has been established [Cohen et al., 2005a, Goguadze, 2005] and we have phrasebooks for each of them [Cohen et al., 2005b].

A common issue concerning the interoperability of different CASs is that the evaluation of the same expression in different CASs might lead to different results. This is a well known problem in the CAS community. For example, the arcsin function returns different results in different CASs, since some CAS define the arcsin function to return an angle between  $-\pi$  and  $\pi$  and others between 0 and  $2\pi$ .

In ACTIVEMATH we do not bind the query actions to the particular CAS in the representation of the query. It is the task of the phrasebook routine to map the query action into the appropriate CAS command for each CAS, connected to ACTIVEMATH.

### 3.4.7 Diagnosis using a CAS

The basic diagnosis of a CAS provides a binary correct/incorrect diagnosis. But more often than not we need a more informative diagnosis.

In addition to check the correctness, authors can encode constraints for typical errors in particular steps and have the system perform a semantic comparison of the learner's answer to these typical errors. LEACTIVEMATH authors used this technique for several types of exercises.

Figure 3.24 shows several kinds of feedback, based on the CAS diagnosis. In the first trial the learner was diagnosed with a typical error, where he forgot to put the exponents in front as factors when differentiating monomials. A specific transition matching this answer and all other equivalent wrong answers was authored for this exercise step.

Compute the derivative of the polynomial  $f(x) = 3 \cdot x^2 + 2 \cdot x^3$

|   |  |
|---|--|
| $f'(x) = 3 \cdot x + 2 \cdot x^2$                 | Not quite, you forgot to put the exponents as factors in front. Try again: |
| $f'(x) = 3 \cdot 2 \cdot x + 2 \cdot 3 \cdot x^2$ | This is correct, but can be further simplified.                            |
| $f'(x) = 6 \cdot x + 6 \cdot x^2$                 | Well Done!   |

Figure 3.24: Diagnosis and feedback generation with CAS

In the second trial the learner has entered a correct, but not fully simplified fraction. This has been detected by a transition, that contains a semantic comparison and the corresponding feedback was presented. Finally, in the last trial the correct answer was given and the transition containing the syntactic comparison fired with the corresponding feedback. Note that in this exercise the part of the feedback of the type KR (knowledge of result) is automatically generated by the system. It consists of a decoration of the learner's answer with different colors that indicate the correctness of the answer. The color of the generated decoration depends on the value of the `achievement` of the learner, annotated within the `diagnosis` element of the transition.

### 3.4.8 Further Intelligent Diagnoses

In order to facilitate the generation of intelligent feedback we defined the following queries to the abstract domain reasoner (the abbreviations have the meanings:  $T$  - Task,  $LA$  - Learner's Answer,  $C$  - context):

- $query('getExpertSolutionPaths', T, C, N)$  returns the list of the first  $N$  steps of each solution path beginning at the current task  $T$  in a given context  $C$ . If  $N = 1$ , we obtain the list of possible next steps.
- $query('getStandardSolutionPath', T, C, N)$  returns the list of the first  $N$  steps of the shortest solution path.
- $query('getNextStep', T, C)$  returns the next step in the standard solution path. It is a special case of the previous query in case  $N = 1$ .
- $query('getResults', T, C, N)$  - returns the list of final nodes of the first  $N$  steps of all solution paths.
- $query('evaluate', T, C)$  - returns the final result of the computation in the given context
- $query('getUserSolutionPath', (T, LA), C, N)$  - returns the shortest path from the point  $T$  to the point  $LA$  in the solution space (this would include erroneous paths if the context contains buggy rules).
- $query('compare', (T, LA), C)$  - returns the correctness value of the learner's answer in the given context.
- $query('getRelevance', (T, LA), C, N)$  - returns the relevance measure (the step is relevant if the learner gets closer to the final solution).
- $query('getConcepts', T, C, N)$  - returns the reference to the domain concepts related to the task.
- $query('getMisconceptions', (T, LA), C, N)$  - returns the references to the misconceptions of the learner (if any have been detected).
- $query('getNumberOfStepsLeft', T, C)$  - gets the number of steps left until the solution is reached (following the shortest solution path).

- *query*('isFinished',  $(T, I), C$ ) - returns the Boolean indication of whether the learner's answer can be accepted as a final step of the solution.

These queries serve the diagnosis of the user's actions and the subsequent generation of intelligent feedback. Having detailed information on the position of the learner in the solution space of the exercise, knowing which concepts are participating in the step and which misconceptions the learner has, provides the basis for the generation of feedback of different granularity and hints of different types. Even just presenting the diagnosis to the learner will give him much more information than just the correct/incorrect statement.

We have formulated our queries for the generation of feedback dimensions in Section 3.3.2 in order to generate elaborate tutorial feedback following the research of Narciss [Narciss, 2006]

Consider again the expression 3.1 from the example in Section 3.4.5, for which we formulate some more queries from the list above.

Figure 3.25 shows the results of the query:

$$\textit{query}(\textit{'getResults'}, ((x + 1)x)', \textit{'diff\_arith'}, 2)$$

This query returns the list of nodes collected from all possible expert solution paths after two iterations of the rule engine. The corresponding nodes are highlighted in green. Such a query serves the generation of the next step to be presented to the learner as a hint.

Figure 3.26 shows the solution graph containing possible erroneous paths introduced by the buggy product rule for function differentiation. Consider results of some queries of this solution graph in different contexts.

$$\textit{query}(\textit{'compare'}, (((x + 1)x)', 1), \textit{'diff\_arith'}) = \textit{false}$$

Comparing the task with the number 1 within this context results in *false*, but

$$\textit{query}(\textit{'compare'}, (((x + 1)x)', 1), \textit{'diff\_arith\_buggy'}) = \textit{true}.$$

Here, the buggy context *'diff\_arith\_buggy'* includes the context *'diff\_arith'* and in addition the following buggy product rule:  $(f(x) \cdot g(x))' = f'(x) \cdot g'(x)$ .

$$\textit{query}(\textit{'compare'}, (((x + 1)' \cdot x)', 1), \textit{'diff\_arith'}) = \textit{true}$$

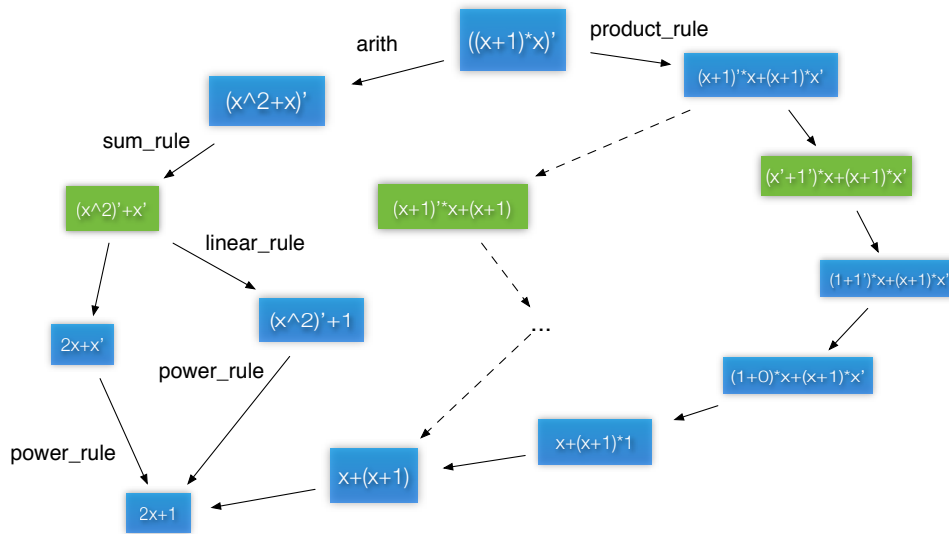


Figure 3.25: Applying the query 'getResults'

The rest of the computation, starting from the application of the buggy rule, can be still performed correctly with respect to the expert context. Using these queries we can diagnose the learner's errors and his position in the solution graph.

In the following sections we will see more examples of diagnosis and feedback generated with the help of the above queries.

Another important task is to allow for partially authored exercises, giving the teacher an opportunity to author his exercise manually and request intelligent diagnosis and some automated feedback generation from the exercise content. In subsequent sections we will see examples of such a hybrid approach to exercises.

### 3.4.9 Domain Reasoner Queries in the Exercise Transitions

Some of the evaluative queries to the intelligent reasoning components that can be used for diagnosis, such as the 'evaluate' or 'compare', can be directly used in the body of the condition element. Figure 3.27 shows an example of using domain reasoner queries inside the body of a condition. The first query asks the domain reasoner whether the learner's answer is the final step

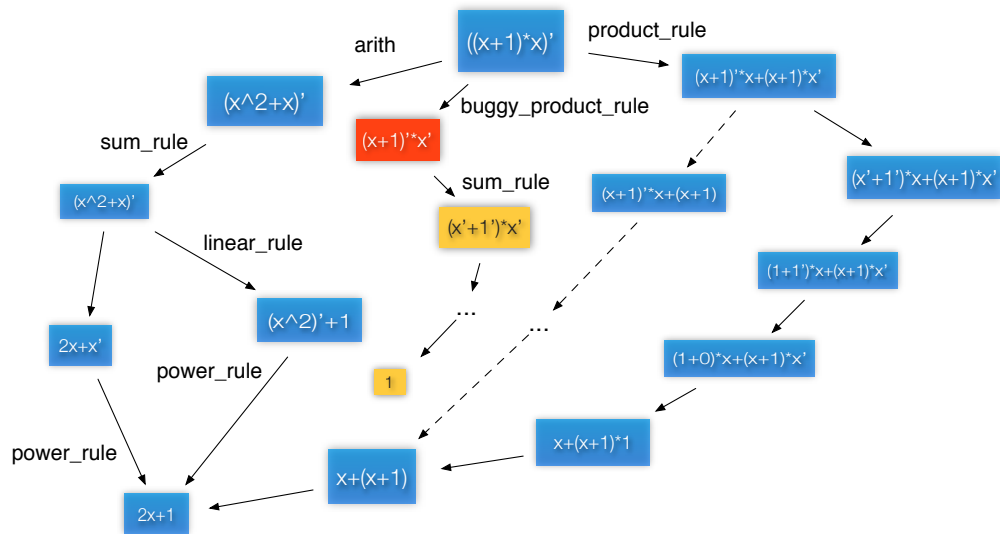


Figure 3.26: Buggy rules in the exercise solution space

of the solution in the given context.

The second query compares the learner's answer with the task equation in the given context. It will match any equation, equivalent to the task.

### 3.4.10 Special Requests

Some conditions match actions of the learner that are not just the answer to a task, e.g., different hint or solution requests, requests to finish or give up the exercise. In order to represent these requests using the same OPENMATH format as for other formal input of the learner, we have introduced an ACTIVEMATH-internal OPENMATH content dictionary, which defines the collection of user requests, named `org.activemath.exercises.user_requests`. This collection defines the following symbols, representing different learner's requests: 'hint', 'process\_hint', 'conceptual\_hint', 'product\_hint', 'next', 'giveup', or 'finish'. Note that, depending on the tutorial strategy it can happen that the learner receives a hint even if he does not request it. Therefore, it is sometimes necessary for the system to know, whether the learner did actually request a hint or not. This information is collected by the local user model, as described in Section 4.6.

A sample representation for a hint request is shown in the Figure 3.28.



### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS 77

```
<transition xref="final_correct">
  <diagnosis><achievement value="1.0" /></diagnosis>
  <condition>
    <query action="isFinished" context="math.lineq">
      3*x+5=9+x
    </query>
  </condition>
</transition>
<transition xref="correct">
  <diagnosis><achievement value="1.0" /></diagnosis>
  <condition>
    <query action="compare" context="math.lineq">
      3*x+5=9+x
    </query>
  </condition>
</transition>
```

Figure 3.27: Matching correct answers with domain reasoner

```
<condition>
  <compare>
    <OMOBJ>
      <OMS cd="org.activemath.exercises.user_requests" name="hint" />
    </OMOBJ>
  </compare>
</condition>
```

Figure 3.28: Representation of a hint request

## 3.5 Modifications and Extensions of the Exercise Representations

The exercise subsystem of `ACTIVEMATH` supports extensions and modifications of the exercise language.

There are two main reasons for extending the exercise format. On the one hand, there can be additional annotations necessary in order to enable a specific functionality of an exercise, which is not possible given the expressive power of the standard `ACTIVEMATH` exercise language.

On the other hand, there could be some specific tutorial behavior, which is representable by means of the standard exercise language, yet it requires a too verbose representation. Such high verbosity can be avoided by extracting the needed tutorial behaviour from the actual representation into an automated procedure. Such a procedure can be applied to a basic exercise in order to

automatically transform it into a more complex one. It modifies the exercise automaton in which it adds or removes transitions and feedbacks. A *tutorial strategy* that describes the behavior of the system in response to learner's actions can employ such procedures. The advantage of this approach is that more than one strategy can be applied to the same basic exercise.

The extension of the exercise representation can be realized by attaching an `exercise_generator` element to the exercise. Below we present some examples of frequently used extensions.

### 3.5.1 Parameterized Exercises using Randomizer

The `Randomizer` generator is used for authoring parameterized exercises, for which the author can define the generic problem statement using variables in places of concrete numbers or mathematical expressions, together with the solution space of this exercise.

When defining such a general exercise, the author can employ the `Randomizer` generator that substitutes random values from a given value range for the parameter variables. Figure 3.29 shows a simple example.

```
<exercise_generator name="Randomizer">
  <parameter name="constant">
    A,1,2,3,4,5,6,7,8,9,10
  </parameter>
  <parameter name="constant">
    B,1,2,3,4,5,6,7,8,9,10
  </parameter>
  <parameter name="constant">
    A_plus_B, A + B
  </parameter>
  <parameter name="evaluated_constant">
    A_plus_B_evaluated, A + B
  </parameter>
</exercise_generator>
```

Figure 3.29: Sample exercise generator representation

This generator accepts two kinds of parameters: so-called constants and evaluated constants. A parameter  $(V, S)$  consists of a variable  $V$ , and the set of admissible values  $S$ . Each instance of the exercise instantiates the parameter by a randomly chosen value from  $S$ .

For instance the variable `A_plus_B` denotes an OPENMATH expression, which after substitution represents the sum of the numbers substituted for

### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS 79

the constants  $A$  and  $B$ . The so-called evaluated constants represent expressions that are evaluated by a CAS at each exercise instance and the result of this evaluation is used in the exercise. So, the value of the variable  $A\_plus\_B\_evaluated$  is equal to the value of the sum of  $A$  and  $B$ .

Figure 3.30 shows the usage of the variables  $A\_plus\_B$  representing the sum expression and the  $A\_plus\_B\_evaluated$  representing the final answer of a calculation.

```
<transition_map>
  <transition to="final_correct">
    <diagnosis<achievement value="1.0" /></diagnosis>
    <condition>
      <compare context="syntactic">A_plus_B_evaluated</compare>
    </condition>
  </transition>
  <transition to="step_cheating">
    <diagnosis<achievement value="0.0" /></diagnosis>
    <condition>
      <compare context="syntactic">A_plus_B</compare>
    </condition>
  </transition>
  <transition to="step_correct">
    <diagnosis<achievement value="0.8" /></diagnosis>
    <condition>
      <compare context="semantic">A_plus_B</compare>
    </condition>
  </transition>
</transition_map>
```

Figure 3.30: Transitions in the randomized exercise

The first transition matches the final correct answer - the sum of the instances of the variables  $A$  and  $B$ . In this transition the learner's answer is syntactically compared to the number  $A + B$ . In the second transition the learner's answer is syntactically compared to the problem statement in order to find out whether the learner is just repeating the problem statement. The last condition checks whether the learner's answer is semantically equivalent to the task.

The possible values of the randomization variables do not have to be integers or reals but can be any mathematical expressions. In the extended version of the Randomizer, developed in a Student Project at the University of Saarland, one can randomize over integer and real intervals, and over sets of functions [Dudev and Palomo, 2007].

### 3.5.2 Static and Dynamic Language Mappings

One of the standard generators used for static transformation of exercises from other XML formats into the ACTIVEMATH format is the XSLT Generator.

This generator takes two parameters: a custom exercise representation and an XSLT stylesheet,<sup>5</sup> and transforms this custom representation into the ACTIVEMATH language.

Figure 3.31 shows an example of a custom exercise format for very simple fill-in-blank exercises, powered by the XSL generator.

```
<exercise>
<metadata>...</metadata>
<exercise_generator name="Xslt">
  <parameter name="stylesheet" href="../data/simple-fib.xml"/>
</exercise_generator>
<problem>
  Find the derivative of a function
   $f(x) = x \cdot \sin(x)$ 
</problem>
<hint> Use the product rule. </hint>
<solution>  $f'(x) = x' \cdot \sin(x) + x \cdot (\sin(x))'$ ; ... </solution>
<answer>  $\sin(x) + x \cdot \cos(x)$  </answer>
<step_correct> This step is correct. </step_correct>
<all_correct> Well done! </all_correct>
<wrong> This step is incorrect. </wrong>
<proceed> Please, proceed with your solution: </proceed>
<repeat> Please, try again: </repeat>
</exercise>
```

Figure 3.31: Custom format for simple fill-in-blank exercises

This format specifies the templates for problem statement, possible hints, solution, correct answer, and several templates for feedback in case of correct or incorrect answers. The XSL stylesheet defines rules that transform such an exercise template into an ACTIVEMATH exercise representation. Note that the XSL stylesheet can also define some default feedback phrases, which can be used if the author is happy with the default feedback. In this case authoring the exercise takes even less time.

This extension allows for creating custom XML high-level descriptive languages for different kinds of exercises alongside with the transformation into the ACTIVEMATH exercise language. Also, the XSLT generator supports

<sup>5</sup>See <http://www.w3.org/TR/xslt>

the integration of any other exercise language, that can be statically mapped into the ACTIVEMATH exercise format.

The OMDOC Quiz module is a part of the OMDOC representation format for mathematical documents [Kohlhase, 2006]. This module defines a simple representation language for one-step multiple choice questions.

Figure 3.32 shows a sample representation of an OMDOC multiple choice exercise. Each exercise can consist of several choice-answer blocks. The `choice` element contains the statement of a choice, and the `answer` element represents the corresponding feedback of the system. Each `answer` element is annotated with a *verdict* attribute defining the correctness of the current choice. Exercises can also contain `hint` and `solution` elements.

```
<exercise>
  <CMP>Problem Statement</CMP>
  <hint><CMP> Hint statement</CMP></hint>
  <solution><CMP>Solution Statement</solution>
  <mc>
    <choice><CMP>choice1</CMP></choice>
    <answer verdict="true"><CMP>Correct answer 1</CMP></answer>
  </mc>
  <mc>
    <choice><CMP>choice2</CMP></choice>
    <answer verdict="false"><CMP>Incorrect answer 2</CMP></answer>
  </mc>
  <mc>
    <choice><CMP>choice3</CMP></choice>
    <answer verdict="true"><CMP>Correct answer 3</CMP></answer>
  </mc>
</exercise>
```

Figure 3.32: Mapping of the omdoc quiz module

This representation implicitly implies that the implementation of the quiz module automatically combines the feedbacks for all the choices that are selected by the learner. In our representation each combination of the learner's answers yields a unique next state, therefore in order to handle such an exercise, we have to create  $n^n$  states of combinations of answer elements, in case of  $n$  choices.

Such a transformation can be performed by a static XSLT, but in order to avoid the combinatorial explosion we chose another way where the mapping is generated dynamically.

An `Exercise Generator` for the OMDOC Quiz module, receives the learner's input in a step and generates a corresponding feedback on the fly,

creating a new node that combines the feedbacks corresponding to each selected choice.

### 3.5.3 Hierarchical Exercise Representations

There is one implicit construct in the exercise representation that allows to include a hierarchy in the representation of exercises: If the `task` of a problem has several subtasks, then several subgoals are introduced together with unconditional transitions to all subgoals from the main `task` node. If more than one unconditional transition is authored in an exercise node, the system follows all of these transitions in the order of their occurrence. The `Exercise Subsystem` of `ACTIVEMATH` will then play each sub-problem in the sequence in which the transitions occur. It will follow the first transition, then run all the consequent steps until the final node of the subgraph is reached and finally return back to the second transition and so on.

However, such a representation would not allow for the full flexibility, that the non-hierarchical representation offers. For example, a student can not provide the final answer to the problem from inside a subgoal. Figure 3.33 shows a hierarchical exercise graph. In this case, allowing the final solution inside the subgoal would mean to define the transition, depicted by a dashed arrow.

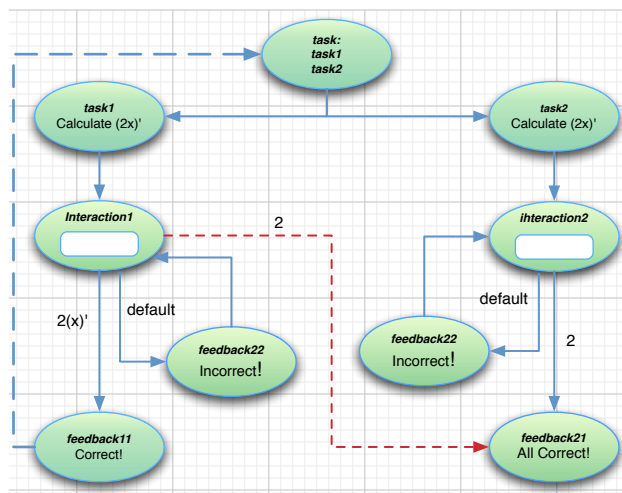


Figure 3.33: Hierarchical exercise representation

### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS83

However, there could be conceptual problems that arise with such a representation. For example, if there is a direct transition possible from inside the first subgoal to the third one, it is not clear whether one should consider the second subgoal solved in this case, or should the system go back to it after solving the third one?

In order to avoid these inconsistencies, we decided to "linearize" the hierarchical solutions by fixing the order of subgoals to tackle. Namely, all the subgoals will have to be solved in the order of their occurrence.

Furthermore, since hierarchical solutions are natural for the class of problems we address in this thesis, we suggest a simple representation format for hierarchical solutions. An excerpt from this representation is shown in Figure 3.34. In this format, a solution consists of sequence of steps, represented by `step` elements. Each step has a textual content, within the `content` element and the `next_step` element, containing the correct answer for the step and a pointer to the next step. Alternatively to the `next_step` element, a step can contain one or more `subtask` elements containing pointers to next steps.

For supporting this representation, we provide an XSLT stylesheet, which can be applied to hierarchical solutions to produce exercise representations in the `ACTIVEMATH` format.

Alternatively, hierarchical solutions can be stored directly in the database and the transformation can be applied at runtime, using the XSLT generator.

```
<solution>
  <step id="step1">
    <content>Differentiate the function:  $(2 \cdot x + 1)'$ .</content>
    <next_step xref="step2"> $(2 \cdot x)' + (1)'$ </next_step>
  </step>

  <step id="step2">
    <content>The next step is:  $(2 \cdot x)' + (1)'$ </content>
    <subtask xref="subtask21"/>
    <subtask xref="subtask22"/>
    <subtask xref="subtask23"/>
  </step>
  ...
</solution>
```

Figure 3.34: Simple format for hierarchical solutions

### 3.5.4 Attaching Tutorial Strategies

A tutorial strategy can be manually assigned to an exercise by attaching the `exercise_generator` which refers to the identifier of the strategy to be used. The value of the attribute *type* has to be set to "strategy". A tutorial strategy consists of a pedagogical strategy, represented by a program that transforms the exercise automaton to induce the needed tutorial behavior and a presentation strategy that defines graphical parameters such as placement of feedbacks, highlighting, folding and other presentational aspects that play a role in perception of the exercise content.

A tutorial strategy of an exercise can be also assigned by the Tutorial Component of ACTIVEMATH, instead of being included into the content. In such cases the `Exercise Subsystem` creates the corresponding `exercise_generator` element on the fly and inserts it into the exercise content, before playing the exercise.

### 3.5.5 Atomic Strategies

Consider now some examples of simple strategies used in ACTIVEMATH and their various combinations.

A KR strategy inserts the feedback of type 'KR' after each learner interaction. It outputs the learner's answer and provides the correct/incorrect feedback, by decorating the learner's answer, and specifying the error location in case of multiple answers.

A KCR strategy inserts the feedback of type 'KCR' if the learner's answer is incorrect and forwards the learner to the next step.

A KH (knowledge on how to) strategy inserts a sequence of procedural hints on how to proceed that can be presented on request of the learner.

A KC (knowledge on concepts) strategy inserts a sequence of conceptual hints of different types that can be presented on request of the learner.

A KM (knowledge of mistakes) strategy inserts error related feedback if the learner's answer is incorrect.

The AUC (answer until correct) strategy returns the learner to the same step until he provides the correct answer or gives up the exercise.

The `Exercise Generator` classes which implement corresponding strategies, can be reused when implementing more complex strategies, as described in Section 4.7. For example, the `Exercise Generator` for the KR strategy is reused for the KRKCR strategy, and the `Exercise Generator` for the AUC



strategy is reused for the KRAUC strategy.

### 3.5.6 Strategies in MATHEFÜHRERSCHEIN and LEACTIVE-MATH

The German project MATHEFÜHRERSCHEIN running at the Fachhochschule Dortmund in collaboration with the ACTIVEMATH group aims at preparing the school graduates for technical college or university. In order to achieve this goal, a collection of learning materials that contains a large set of interactive exercises is served by the ACTIVEMATH system. Each interactive exercise consists of a single step and uses a simple tutorial strategy, shown in Figure 3.5.6.

**Beispiel 2**

Welcher Bruchteil wird in grauer Farbe jeweils dargestellt und welche Addition damit insgesamt? Kürzen Sie alle Eingaben vollständig!

Die richtige Antwort wäre:

$\frac{1}{4} + \frac{2}{4} = \frac{3}{4}$

$\frac{1}{4} + \frac{1}{2} = \frac{3}{4}$

Zurück Lösung Zurück Eingabehinweise

Figure 3.35: Exercise strategy in MATHEFÜHRERSCHEIN

This strategy was proposed by educationalists. The learner enters his solution for the problem and clicks the button "Stimmt's?" ("Is it correct?"). If the answer is correct, the system acknowledges the correctness by highlighting the correct answer in green. If the answer is incorrect, the flag feedback is presented as well, highlighting the incorrect answers in red. Additionally, the button for requesting the correct solution is presented. However, the learner is still allowed to go back to the interaction and try answering again.

Another simple tutorial strategy, shown in Figure 3.5.6, was used within the EU-project LEACTIVE-MATH. After each incorrect answer the learner receives the correct/incorrect feedback, including specific error feedback in case a typical error is detected. Then it returns to the same step to try again.

Sequences of hints of increasing level of detail followed by the correct solution can be authored for each step of the exercise. The system will automatically count the number of hints already given in order to provide the subsequent hint at the next request. Additionally, the student can always give up the exercise, using the dedicated `giveup` button.

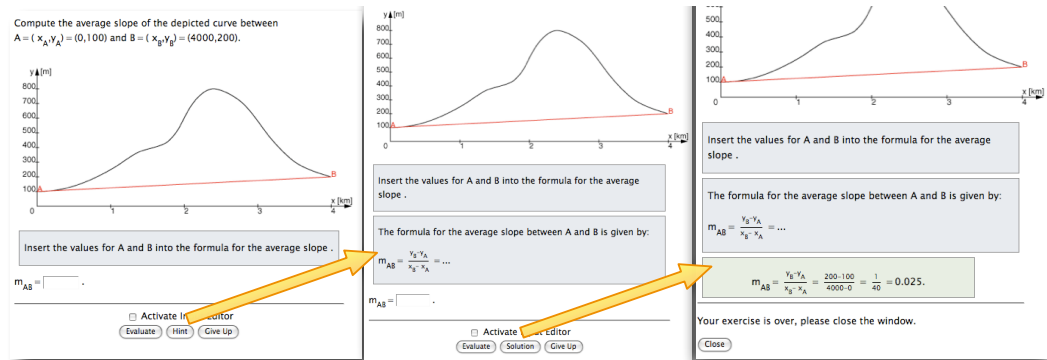


Figure 3.36: Exercise Strategy in LEACTIVE MATH

### 3.5.7 Composite Strategies in ATuF and ALOE

Some tutorial strategies need additional parameters for identifying specific steps of the exercise. For example, we have defined strategies in the projects ATuF (Adaptive Tutorial Feedback) and ALOE (Adaptive Learning with erroneous Examples) that differentiate between two phases in some exercises. Each of these phases uses a different tutorial sub-strategy.

Although such a two-phase strategy is completely independent of the exercise representation, the strategy needs to be provided with the identifier of the exercise node at which the second phase starts in order to change the tutorial procedure.

In case of the ALOE strategy, it is sufficient to specify the identifier of the step at which the second phase of the exercise begins. The strategy should not hardcode the identifiers of particular exercise steps, since it has to work for all exercises.

Figure 3.37 shows the representation of such a language extension in which the strategy generator uses the parameter `second_phase_id`.

### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS87

```
<exercise id="ewh_PaulEx">
  <metadata> ... </metadata>
  <exercise_generator name="aloe" type="strategy">
    <parameter name="second_phase_id" xref="ewh_PaulEx_phase2" />
  </exercise_generator>
  <task>...
</exercise>
```

Figure 3.37: ALOE two-phase exercise strategy

The screenshot of the exercise as presented to the student is shown in Figure 3.38, where the first phase of the exercise consists of a multiple choice question which aims at the identification of the wrong step in a given erroneous solution. In the second phase the learner has to correct the erroneous step.

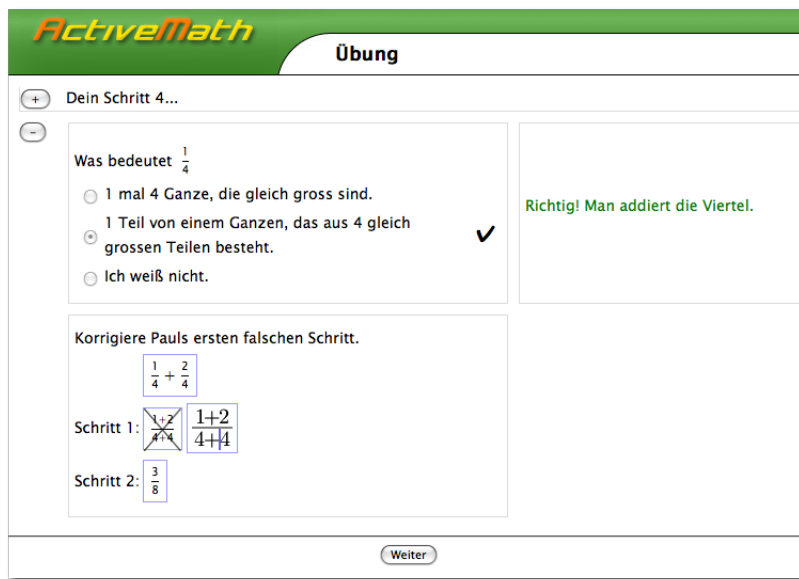


Figure 3.38: ALOE exercise rendering

A specific presentation strategy is used in ALOE exercises. The feedback is always presented immediately after the learner's answer. Irrelevant previous steps are folded, but not removed from the screen. The learner can still get back to the previous steps by unfolding them. A new hybrid interactive element is introduced in the second phase: a multiple choice question is combined with a free input. First, the learner has to click on the step he wants

to modify - at this stage all the steps are clickable and act as multiple choice questions. After the learner clicks on one step, the other steps are fixed and they are not clickable any more, the current step is crossed through and an input field appears.

In case of the ATuF strategy the second the phase is triggered in a similar way as in ALOE - a parameter for the tutorial strategy is defined pointing to the initial node of the second phase. However, in this case more annotations need to be assigned to the second phase sub-strategy, such as the references to the different kinds of feedbacks.

```

<exercise id="Represent_Peter">
  <metadata> ... </metadata>
  <exercise_generator name="atuf" type="strategy">
    <parameter name="sequence">ch, ce, kcr</parameter>
    <parameter name="second_phase" xref="Erweitern_Klaus_step2">
      <ref type="kcr" xref="Erweitern_Klaus_solution"/>
      <ref type="ch" xref="Erweitern_Klaus_conc_hint"/>
      <ref type="ph" xref="Erweitern_Klaus_proc_hint"/>
      <ref type="ce" xref="Erweitern_Klaus_conc_exp"/>
      <ref type="pe" xref="Erweitern_Klaus_proc_exp"/>
    </parameter>
  </exercise_generator>
  <task>...
</exercise>

```

Figure 3.39: Additional relation types in an ATuF exercise

Using these annotations, the strategy can define the sequence and the types of feedbacks presented in a concrete instance of the exercise, depending on the value of the `sequence` parameter. The value of this parameter can be modified by the Tutorial Component of ACTIVE MATH on the fly. Without the application of the strategy all the feedbacks would have been presented in the order of their occurrence.

Figure 3.39 shows a sample representation of such strategy parameters for the ATuF strategy.

The screenshot of the resulting exercise is shown in Figure 3.40.

ATuF employs a complex graphical presentation strategy. The first phase is a multiple choice question, the second part has a complex graphical layout. The user interaction and its results are presented on the left, and the feedback and hints of different types are presented on the right of the screen, as shown in Figure 3.40. As in the case of ALOE, the previous steps are folded, but the learner has a possibility to unfold them and look at them.

### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS89

The presentation strategy of ATuF introduces a new kind of interactive element, in order to facilitate the learner’s interaction with the system. The interactive element consists of a drop-down selection. After the learner selects an appropriate response, a structured template is presented to the learner, which contains blanks at some places.

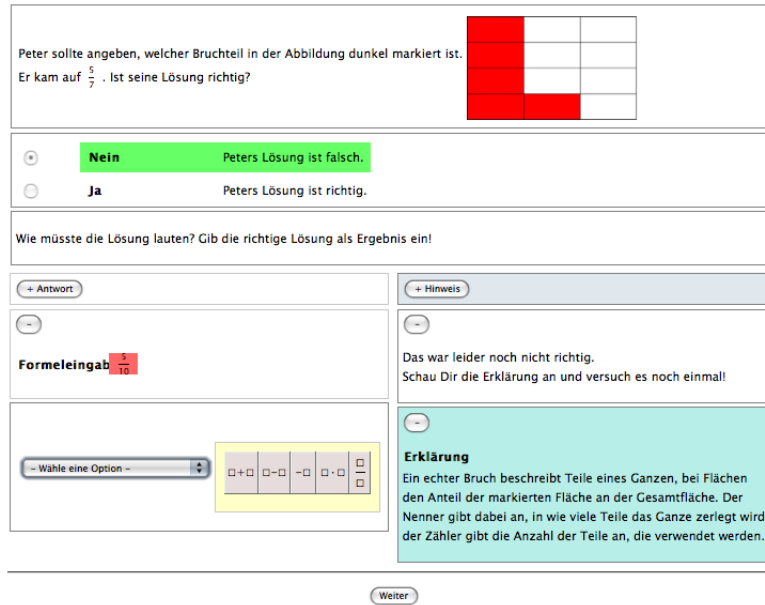


Figure 3.40: ATuF exercise rendering

This exercise can be reused with a different strategy, in which the value of the parameter `sequence` is changed. In this case the strategy will change the sequencing of different kinds of feedback in a local feedback loop.

For more information about the usage of strategies in the projects ATuF and ALOE see [Eichelmann et al., 2008] and [Tsovaltzi et al., 2009] respectively.

#### 3.5.8 Exercises with local strategies

The next natural step is to build a mechanism which allows to use different tutorial strategies for different parts of the same exercise. In order to do so, we apply each of them locally to the exercise at the places where they are needed. Hence, we allow to insert the `exercise_generator` element not only

at the top level of an exercise, but within any exercise step. The starting point of the strategy is the node at which the local strategy generator is inserted. There can be more than one final node of the local strategy, since the sub-exercise to which such local strategy is applied can have several terminal states.

Alternatively, the local strategy generators can be listed at the beginning of the exercise parametrized by the identifiers of all the initial and final nodes. In this case we need to specify only once that the same local strategy is applied to several sub-exercises, instead of putting the generator element in the body of each initial node.

Figure 3.41 shows the sample source code of a local strategy, called 'KRKCR-N'. This strategy gives the learner several trials for the task, each time presenting some KR feedback which tells the learner whether his answer is correct or not. After a number of incorrect trials, the correct solution (KCR) is given. The number  $N$  of trials is one of the parameters of the strategy. Another parameter is the identifier of the node, containing a KCR feedback. KR feedbacks are automatically generated, based on the diagnosis of the learner's answer. Since the strategy is used locally, the identifiers of initial and final nodes are among the parameters of the strategy.

```
<exercise_generator name="KRKCR-N" type="strategy">
  <parameter name="initialnode" xref="crossaxis-y_interactive" />
  <parameter name="finalnode" xref="crossaxis-y_kcr" />
  <parameter name="finalnode" xref="crossaxis-y_correct" />
  <parameter name="kcrid" xref="crossaxis-y_kcr" />
  <parameter name="N">2</parameter>
</exercise_generator>
```

Figure 3.41: Sample representation of local strategy

### 3.5.9 Multi-Exercises in MAPS MOSSAIC

The exercise language of ACTIVEMATH was employed to represent a multi-exercise which has been reused by several complex tutorial and presentation strategies within the project MAPS MOSSAIC (MAtheMatical Problem Solving and MOdelling: Scaffolding Self-Regulated Acquisition of Interdisciplinary Competencies). Under a multi-exercise we understand a collection of exercises that are integral parts of one problem. The order of execution of each of these exercises and the tutorial strategy used for each of them is

### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS91

controlled by another top-level strategy. Tutorial strategies developed in this project have varied the sequencing of sub-exercises, representation of tasks and presentation and sequencing of feedback and hints. For more information on the research goals of the project see [Polushkina, 2009].

Figure 3.42 shows a structured interaction interface in one of the phases of this complex strategy. On the left we see templates for the statements of the subgoals of the current exercise phase. The statements of the subgoals which are not yet solved have placeholders for the missing values, for the solved subgoals these placeholders are filled with the correct answers.

Frage vereinfachen (1)
Informationen sammeln (2) und Teilaufgaben bearbeiten (3)
Frage beantworten (4)

**Entfernung Haus - Tankstelle:**  
Landstraße  km  
Autobahn  km

**Durchschnittliche Fahrgeschwindigkeit:**  
Landstraße  km/h  
Autobahn  km/h

**Fahrtdauer zu den Tankstellen:**  
Gasolina  min  
Fuel  min

**Unterschied in der Fahrtdauer:**  
Zu der Tankstelle  min  
länger, als zu der Tankstelle

Fahrtdauer zu den Tankstellen:  
Gasolina  min  
Fuel  min

Leider falsch! Versuch es noch einmal.

Die **Entfernungen zu den beiden Tankstellen** und die **durchschnittlichen Fahrgeschwindigkeiten** sind schon bekannt:

|   |         |
|---|---------|
| Haus - Gasolina (über Landstraße):      | 12,5km  |
| Fahrgeschwindigkeit für die Landstraße: | 75km/h  |
| Haus - Fuel (über Autobahn):            | 10km    |
| Fahrgeschwindigkeit für die Autobahn:   | 120km/h |

Berechne jetzt die Fahrtdauer zu den Tankstellen und trage Deine Ergebnisse hier ein:

Fahrtdauer zu den Tankstellen:  
Gasolina  min  
Fuel  min

Kevin weiß schon, wie das geht, und probiert das erst einmal für die Sporthalle aus: Der Weg zur Sporthalle geht durch die Stadt und ist **7km** lang. Die durchschnittliche Fahrgeschwindigkeit in der Stadt ist **40km/h**.

| Strecke (in km) |    | Fahrtdauer (in min) |     |
|-----------------|----|---------------------|-----|
| :40             | 40 | 60                  | :40 |
| *7              | 1  | 1,5                 | *7  |
|                 | 7  | 10,5                |     |

Die Fahrt zur Sporthalle dauert also etwa 10 bis 11 Minuten.

So rechnen wir das auch für die beiden Tankstellen aus!

Weiter
Tipp

Figure 3.42: MAPS MOSSAIC strategy example

In order to enable such a subgoal presentation, additional information needs to be attached to the exercise representation, as shown in Figure 3.43. Namely, the strategy parameter with the name 'subgoals' is introduced, which consists of the reference to the main task and the list of subgoals, represented using a **subgoal** element. Each subgoal has two child elements - the **template** element which is used by the presentation strategy while the subgoal is not yet solved, and the **answer** element which replaces the

`template` as soon as the subgoal is correctly answered. If the learner fails to solve the subgoal correctly after several trials, the strategy informs the learner of the correct answer and replaces the `template` element with the `answer`.

As we can see in Figure 3.42, there are several graphical structure elements, such as a horizontal bar indicating the learner’s progress, a list of subgoals on the left, a custom placement of feedback and hints. All these graphical components are realized by a presentation strategy which uses the metadata of the feedbacks and the additional annotations of the content by the tutorial strategy.

```
<exercise_generator type="strategy" name="selfreg">
  <parameter name="subgoals">
    <task xref="self_reg_task2" />
    <subgoal xref="subgoal2_1">
      <template<with style="self_reg2">Entfernung Haus - Tankstelle:
        <table>
          <tr><td>Landstrasse <with style="blank" /> km</td></tr>
          <tr><td>Autobahn <with style="blank" /> km</td></tr>
        </table></with>
      </template>
      <answer>
        <with style="self_reg2">Entfernung Haus - Tankstelle:
          <table>
            <tr><td>Landstrasse <with style="blank">12,5</with> km</td></tr>
            <tr><td>Autobahn <with style="blank">10</with> km</td></tr>
          </table></with>
        </answer>
      </subgoal> ...
    </parameter> ...
  </exercise_generator>
```

Figure 3.43: Representation of subgoal templates

Apart from the sophisticated tutorial strategies for each of the sub-exercises, we implemented several global strategies for navigating between the sub-exercises in order to realize the meta-cognitive aspects of the MAPS MOSAIC strategies. Figure 3.44 shows a so-called self-regulation panel where the learner chooses himself the order of solving the subgoals. This is followed by an automatic reordering of the subgoal templates in the consequent steps.



### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS 93

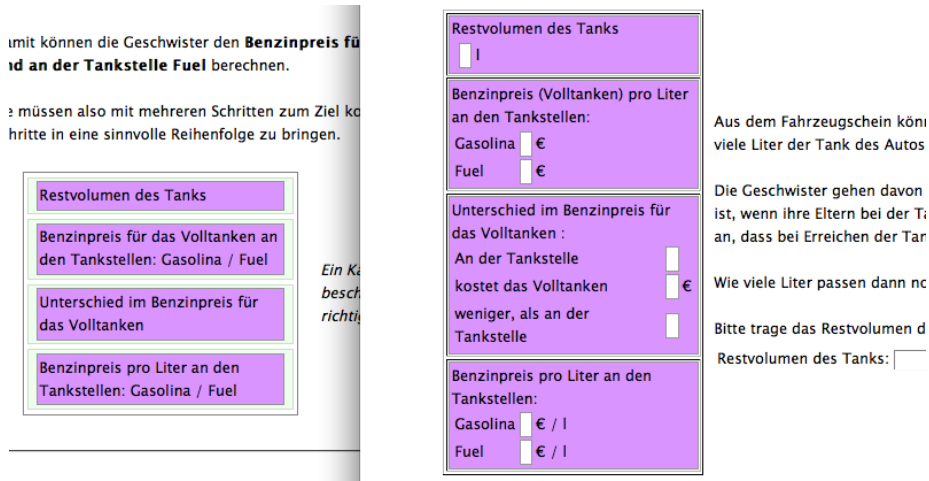


Figure 3.44: MAPS MOSSAIC subgoal ordering

Note that the additional information needed for specific tutorial behavior is represented within the parameters of the tutorial strategy. Therefore, all the exercises can also be reused using the default strategy.

See Appendix B for the more detailed description of the important aspects of the MAPS MOSSAIC strategies which were fully realized by the ACTIVEMATH system.

#### 3.5.10 Feedback generation

In this section we describe, which feedback can be automatically generated in a hybrid exercise.

First of all, some standard feedback commonly used in a tutorial dialogue can be inserted automatically. For instance, flag feedback can be generated automatically based upon the diagnosis of the learner's answer. Other feedback types that can be inserted automatically are the requests to repeat the step, motivational feedback which warns about unfinished task and other domain independent feedback.

We have extended the exercise language such that it supports insertion of requests for automatically generated feedback in any exercise step, using the `feedback_generator` element. Figure 3.45 shows an example representation of the commonly used automatic KCR feedback, placed in the body of a `feedback`. It takes the current task and the context as parameters and

generates the correct solution for the task.

```
<feedback>
  <feedback_metadata>
    <type category="product" value="KCR" />
  </feedback_metadata>
  <feedback_generator context="semantic">
    <parameter name="task">(x3 - 5x2 - 6x)'</parameter>
  </feedback_generator>
</feedback>
```

Figure 3.45: Feedback generation using CAS

A specific kind of KR feedback that is frequently used in ACTIVE MATH exercises is generated from the content of the previous interaction, in which the learner's answer is substituted into the interactive elements and decorated automatically using colors or other decoration styles depending on the correctness of the answer. An example of such generated feedback is shown, e.g., in Figure 3.24. This feedback can be represented as a special kind of `content` element within any `feedback`. This element has an empty body and an attribute `from` specifying the id of the previous interaction, as well as an attribute `input_decoration` specifying the type of decoration for the user input. This `content` element can be possibly followed by another `content` element representing additional custom feedback, as shown in Figure 3.46.

```
<feedback id="interaction1_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="interaction1" input_decoration="automatic" />
  <content><CMP>Not quite, you forgot to put the exponents as
    factors in front. Try again:</CMP>
  </content>
  <transition to="interaction1" />
</feedback>
```

Figure 3.46: Commonly used generated KR feedback

Another kind of automatically generated feedback is based upon an intelligent diagnosis of the learner's answer if a domain reasoner service is available. The feedback generation architecture can send queries to intelligent domain reasoner services in order to generate single feedbacks.

Figure 3.47 shows the representation of a next-step hint to be generated.

```

<feedback>
  <feedback_metadata>
    <type category="product" value="KH" />
  </feedback_metadata>
  <feedback_generator context="math.lineq">
    <parameter name="task">3x + 5 = 9 + x</parameter>
  </feedback_generator>
</feedback>

```

Figure 3.47: Feedback generation using a domain reasoner

Even though the domain reasoner services are remote and do not necessarily have their own internal state, each query needs some initial parameters that might depend on the state and on the runtime parameters of the exercise. For instance, the query `getNextStep`, requesting the computation of the next possible correct steps needs a current task as input parameter.

In order to use problem related procedural feedback generation within the authored exercise, the author should use the specific **Exercise Generator** called **DRGenerator**. This **Exercise Generator** takes care of keeping track of the domain reasoner related runtime parameters, such as a current task. In this case it is enough to specify the initial task as the parameter of feedback generation, and it will be updated automatically in a subsequent request for feedback generation.

### 3.5.11 Local Exercise Generators in a Curve Sketching Exercise

Here we describe another complex exercise that serves as a good illustration of the combined approach to authoring interactive exercises in **ACTIVEMATH**, and a cross-domain exercise diagnosis powered by domain reasoners. Both aspects can currently only be achieved with the **ACTIVEMATH** system.

As a part of a standard calculus course, students learn to perform the so-called curve sketching. The students have to be able to sketch the curve of a function, investigating important features of the function. They use standard calculus techniques such as finding zeros of the function, local extremum, monotonicity intervals, inflection points, and so on.

In **ACTIVEMATH** we offer students to perform this procedure within an interactive exercise. This exercise consists of several sub-exercises, in which the student is given the standard tasks used in curve sketching. Each of the sub-exercises is either authored manually or generated automatically with

the help of some domain reasoner. In this exercise several domain reasoners were used for different sub-exercises.

```
<!-- crossing axis x -->
<task id="crossaxis-x">
  <task_metadata>
    <relation type="for"><ref xref="functions/root"/></relation>
  </task_metadata>
  <exercise_generator name="IDEASGenerator">
    <parameter name="problemstatement">
      3. Now find the points at which the graph of the
      function is crossing the  $x$  axis. In order to
      do this, solve the following equation:
    </parameter>
    <parameter name="context">algebra.equations.polynomial</parameter>
    <parameter name="task">
       $x^3 - 5x^2 - 6x = 0$ 
    </parameter>
  </exercise_generator>
  <transition to="crossaxis-y"/>
</task>
```

Figure 3.48: Representation of domain reasoner-powered sub-exercise

Figure 3.48 shows the knowledge representation of a sub-exercise, generated using a Domain Reasoner. In order to represent such a sub-exercise, the `exercise_generator` element is employed. This generator has several parameters: the name of the `Exercise Generator`, a problem statement for the exercise, a reasoning *context* and a formally represented task.

Figure 3.49 shows the knowledge representation of a manually authored sub-exercise, parametrized with a tutorial strategy. The parameters of the corresponding `exercise_generator` include the name of the strategy, identifiers of initial and final nodes and a custom parameter, representing the maximal number of trials for each step of a sub-exercise, running with this strategy.

### 3.5. MODIFICATIONS AND EXTENSIONS OF THE EXERCISE REPRESENTATIONS 97

```

<!-- crossing axis y -->
<task id="crossaxis-y">
  <content keep="yes">
    <CMP xml:lang="en">4. Now find the coordinates of crossing axis
      y. In order to do this, calculate f(0):</CMP>
    </content>
    <transition to="crossaxis-y_interactive" />
  </task>
<interaction id="crossaxis-y_interactive">
  <exercise_generator name="KRKCR-N" type="strategy">
    <parameter name="initialnode" xref="crossaxis-y_interactive" />
    <parameter name="finalnode" xref="crossaxis-y_kcr" />
    <parameter name="finalnode" xref="crossaxis-y_correct" />
    <parameter name="kcrid" xref="crossaxis-y_kcr" />
    <parameter name="N">2</parameter>
  </exercise_generator>
  <content>...</content>
  <interaction_map>...</interaction_map>
  <transition_map>...</transition_map>
</interaction>

```

Figure 3.49: Representation of a sub-exercise using a tutorial strategy

The sub-exercises represented in 3.48 and 3.49 are shown in Figure 3.50. The presentation on the left shows the domain reasoner-powered exercise, in which the automatically generated next step hint and a complete generated solution are shown. The presentation on the right shows the application of the KRKCR strategy on the next sub-exercise, in which after 2 unsuccessful trials, the learner is presented with the correct answer.

3. Now find the coordinates of crossing axis x.  
In order to do this, solve the following equation  
 $x^3 - 5 \cdot x^2 - 6 \cdot x = 0$   
The next correct step is :

$(x = 0) \vee (x^2 - 5 \cdot x - 6 = 0)$

The correct solution is :

|        |  |
|--------|--|
| Step 1 | $(x = 0) \vee (x^2 - 5 \cdot x - 6 = 0)$ |
| Step 2 | $(x = 0) \vee ((x + 1) \cdot x - 6 = 0)$ |
| Step 3 | $(x = 0) \vee (x = -1) \vee (x = 6)$     |

4. Now find the coordinates of crossing axis y. In order to do this, calculate f(0) :

f(0) = 1

Something went wrong, try again :

f(0) = 2

Something went wrong again.

The correct answer is : f(0) = 0

Figure 3.50: Calculating intersections with coordinate axes

See Appendix A for a full commented listing of this hybrid multi-exercise.

## 3.6 Interoperability of the Exercise Format

In the following we describe the interoperability of the *ACTIVEMATH* exercise format with the known standards for knowledge representation of interactive exercises and the relations to other exercise languages.

### 3.6.1 Compliance to IMS QTI Standard

In order to make the *ACTIVEMATH* exercises available to a wider community than only *ACTIVEMATH* users it is wished to export the exercises of *ACTIVEMATH* to some widely used standard knowledge representation, in order to run these exercises within systems other than *ACTIVEMATH* platform.

The most popular full-featured platform for eLearning representation is IMS Global Learning Consortium Standardization Platform. It consists of IMS Content Packages, SCORM Sequencing of learning materials and Learning Object Metadata (LOM) for instructional objects. Another IMS standard is Question and Test Interoperability (QTI) which is a general purpose format for interactive exercises.

Until recently QTI was a format for representing single step multiple-choice exercises. In version 2.0, the QTI format has been extended to support multi-step exercises. Another format for interactive exercises was developed at the University of Edinburgh [Mavrikis and Maciocia, 2003] for the eLearning system WALLiS. This format has been partially derived from the first versions of *ACTIVEMATH* exercise language and shares notational similarity with QTI. Later, this format gave rise to MathQTI - the extension of the QTI standard, which enabled semantic evaluation of fill-in-blank exercises by using the semantic markup *OPENMATH* format for mathematical formulas, just as in the exercise language of *ACTIVEMATH*. For more information about the described formats and their interoperability, see [Goguadze et al., 2006].

Thus, QTI-2 with MathQTI extension is a good candidate for a general-purpose standard which the *ACTIVEMATH* exercises can be exported to. Such export leads to the partial loss of information, since QTI does not represent all the metadata classifying the tasks and feedbacks.

A transformation tool exporting an earlier version of the *ACTIVEMATH* exercise format was built by a Bachelor Project at University of Saarland under the supervision of the author.

### 3.6.2 Relation to Other Exercise Languages

Another widely used format for interactive exercises is the CTAT format of so-called example-tracing tutors (former pseudo-tutors). Using the CTAT tool one can author the exercise by demonstrating and recording student's behavior and the reaction of the system [Koedinger et al., 2004]. The tool is "recording" such an intelligent behavior into a so-called behavior graph. The nodes of this graph are the exercise steps and the edges are the transitions simulating the learner's answer. Each step can be annotated with one or more hints, which are ordered sequentially at runtime. Also, so-called knowledge labels are attached to the links in the behavior graph to refer to the underlying domain concepts.

From the description above it is easy to see that the CTAT behavior graphs can be isomorphically mapped into the exercise language of ACTIVE-MATH. The only obstacle on the way to implement such a transformation is that the specification of the CTAT format is not publicly available.

There have been recent developments that provide three mechanisms to generalize the behavior graphs, as described in [Alevin et al., 2009]. The first mechanism allows the teacher to define the arbitrary nested groups of steps that should be ordered or unordered. This can be realized in ACTIVE-MATH exercises by a tutorial strategy. The second mechanism extends the evaluation capabilities by allowing the author to specify the range of student inputs by providing the numeric range, set of values, or a regular expression. The third mechanism specifies how one step depends on others by attaching "Excel-like" formulas to the links in a behavior graph.

Similar mechanisms have been implemented in the tutorial strategies of the MAPS MOSSAIC project, in which the custom constraints between some types of the exercise states were programmed as a part of the strategy behavior. In the exercise representation we avoid the custom relationships between the exercise states, which influence the exercise behavior at runtime. Such a representation is not declarative and involves programming, which increases the authoring effort.

Another exercise format, called MATHDOX was developed at the Technical University of Eindhoven (TUE) in the Netherlands. The extensions, made to the MATHDOX language made it quite similar to the ACTIVE-MATH exercise language. In this language the exercises also consist of states and transitions with the difference that the conditions use MONET queries <sup>6</sup> for

---

<sup>6</sup><http://monet.nag.co.uk/monet/>

evaluating the learner's answer.

In order to enable the interoperability of MATHDOX and ACTIVEMATH exercises within the LEACTIVEMATH project, a dynamic **Exercise Generator** was implemented by the author, and a remote TUE service answering some dedicated queries of ACTIVEMATH has been implemented by TUE.

The mechanism for this interoperability is based on a statefull communication. The exercise system of ACTIVEMATH requests every new step of the exercise from the MATHDOX exercise repository, and sends to the MATHDOX the learner's answer represented in OPENMATH format. In order to keep the state of the exercise additional state information is passed by the MATHDOX to the ACTIVEMATH, which passes it back in the next step request. The requested new step is returned to the ACTIVEMATH, converted to the ACTIVEMATH exercise format on the fly and rendered for ACTIVEMATH.

Note that the exercise is not only rendered in ACTIVEMATH in order to use its presentation facilities, but the learner's results also update the **Student Model**. In order to achieve such a proper updating of the **Student Model**, the MATHDOX exercises have included the ACTIVEMATH metadata into their own metadata e.g., educational metadata such as PISA competencies, **difficulty** and **learningcontext** of the exercise tasks.

Thus, properly annotated MATHDOX exercises, stored in the MATHDOX repository in TUE, can be used from within the ACTIVEMATH system and inform the student model of ACTIVEMATH about the learner's progress.



# Chapter 4

## Exercise System Architecture

The **Exercise Subsystem** is responsible for interpreting the knowledge representation of interactive exercises, as described in the Chapter 3 and also for running these exercises in the **ACTIVEMATH** user interface. Running an exercise typically includes the presentation of the exercise task to the learner, evaluating the results of learner interaction and choosing or generating the feedback and the task for the next step.

The **Exercise Subsystem** has a modular architecture, initially defined in [Gogvadze et al., 2005a] and [Palomo, 2006]. Since then it has been significantly extended. We will show how the separation of components contributes to the reusability of exercises and why it makes it easy to implement different extensions to the functionality realized via tutorial and presentation strategies.

The central component of the **Exercise Subsystem** is the **Exercise Manager**. It traverses the graph of the exercise nodes and passes the evaluation and feedback generation to the **Diagnoser** and **Exercise Generator** respectively. The **Exercise Controller** is a layer between the **Exercise Manager** and the rest of **ACTIVEMATH**. It handles the basic exercise session and collects all the necessary parameters to be communicated between the **Exercise Subsystem** and the rest of **ACTIVEMATH**. It also employs the presentation system of **ACTIVEMATH** for rendering the steps in the GUI (browser).

Figure 4.1 shows the basic architecture of the **Exercise Subsystem**. The workflow can be described as follows:

- the **Exercise Controller** receives an exercise together with an ID of

the tutorial strategy and other exercise parameters

- the **Exercise Controller** initializes the exercise session and passes the exercise and the parameters to the **Exercise Manager**
- the **Exercise Manager** queries the **Exercise Generator** for the initial state and passes the received initial state to the **Exercise Controller**
- the **Exercise Controller** applies the presentation strategy and sends the ready initial state to the **User Interface**
- the initial state of the exercise is presented to the learner
- the learner submits his answer to the system via the **User Interface**
- the **Exercise Controller** receives the learner's input and passes it to the **Exercise Manager**
- the **Exercise Manager** sends the learner's input to the **Diagnoser**
- the **Diagnoser** sends queries to the evaluation services and receives the diagnosis of the learner's answer
- the **Diagnoser** sends the obtained diagnosis to the **Exercise Manager**
- the **Exercise Manager** updates the state of the **Local Student Model** which stores the runtime information needed for tutorial strategies
- the **Exercise Manager** queries the **Exercise Generator** for the next state
- based upon the diagnosis the **Exercise Generator** selects (generates) the next step of an exercise (consulting the tutorial strategy and the **Local Student Model**) and returns it to the **Exercise Manager**
- the **Exercise Manager** sends the next step to the **Exercise Controller**
- the **Exercise Controller** updates the **Student Model** of **ACTIVE-MATH** with metadata and diagnosis of the current step
- the **Exercise Controller** applies the presentation strategy and passes the ready step to the **User Interface**

Below we explain the function of each component in detail.

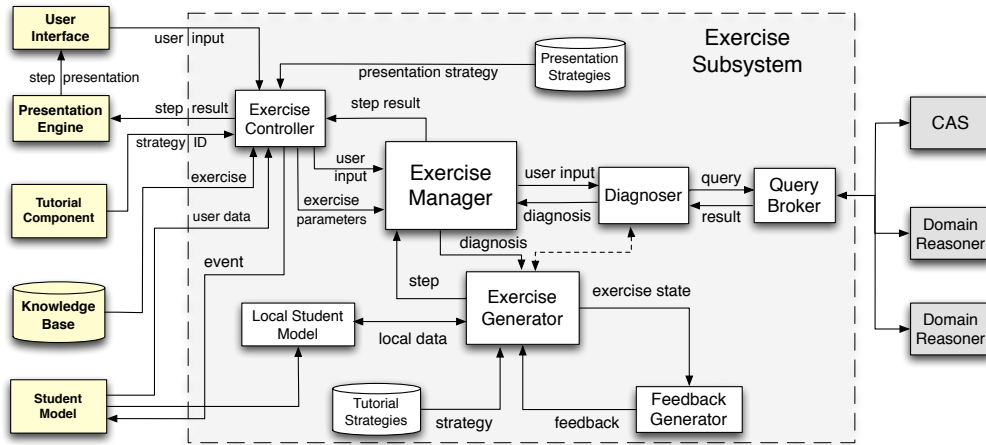


Figure 4.1: Architecture of the exercise subsystem

## 4.1 Exercise Controller

The **Exercise Controller** is a part of the Model-View-Controller architecture of **ACTIVEMATH**. This architecture separates application logic (*controller*) from application data (*model*) and the presentation of that data (*view layer*), as described in [Ullrich et al., 2004].

The **Exercise Controller** establishes the connection of the **Exercise Subsystem** to the **User Interface** and other parts of the **ACTIVEMATH** system. The task of the **Exercise Controller** is to handle the basic session management of the exercise and invoke the **Exercise Manager** with needed parameters. It also applies the presentation strategy to the exercise step and passes the result to the view layer of the presentation architecture of **ACTIVEMATH**.

An exercise can be started by the learner request from the **ACTIVEMATH** content collection, and it can be initialized by the **Tutorial Component** of **ACTIVEMATH**. The **Exercise Controller** initializes the exercise session, creates a new instance of the **Exercise Manager** and passes it the exercise representation and other initial parameters. It also produces an **Exercise Started** event. Events are special messages to the rest of the system delivering the necessary data to the **Student Model** and other interested components of **ACTIVEMATH** system (see section 4.5). After each step, the **Exercise Controller** passes the result of the step to the **Presentation System**

and issues an **Exercise Step** event, that contains the metadata of the step and the diagnosis of the learner's answer. After the last step, an **Exercise Finished** event is fired.

## 4.2 Exercise Manager

The **Exercise Manager** is the central component that manages the tutorial dialogue within an exercise session. It interprets the knowledge representation of the exercise and the tutorial strategy, and has direct access to the generative and evaluative components of the **Exercise Subsystem**.

After being initialized by the **Exercise Controller**, the **Exercise Manager** requests the **Exercise Generator** for the initial exercise step. The representation of the initial exercise step, received from the **Exercise Generator** is then sent to the **Exercise Controller**, that applies the needed presentation strategy and sends it further to the **User Interface** via the **Presentation System** of **ACTIVEMATH**.

The learner can provide the answer to the task in the exercise step, using the interactive elements in the **User Interface**, such as **blanks** and **selections**, as well as perform other actions such as request help or give up the exercise by clicking on corresponding buttons. After the interaction, the learner's answer is sent by the **User Interface** back to the **Exercise Manager** (via **Exercise Controller**), which forwards the answer to the **Diagnoser**, asking for the diagnosis.

After the diagnosis is received, the **Exercise Manager** queries the **Exercise Generator** for the next step, based upon the diagnosis of the learner's answer. Finally, the next exercise step is sent to the **User Interface** via **Exercise Controller** and the interaction/feedback loops starts again.

When the final step is reached the **Exercise Manager** terminates.

## 4.3 Diagnoser

The **Diagnoser** is responsible for providing a diagnosis upon the learner's answer to the task in the exercise step.

In case of hand-crafted exercises a number of transitions is authored for each interaction node of an exercise. The transitions contain conditions to be satisfied in order for the transition to fire, together with the corresponding

diagnosis record and the identifier of the next node of the exercise. In this case **Diagnoser** just verifies the conditions of each transition in the order of their occurrence and stops at the first transition that fires. Then it returns the diagnosis contained in this transition and the identifier of the next state.

In case of generated or partially generated exercises it can happen, that no transitions are authored in a step, and the next step has to be generated on the basis of the current one and the diagnosis of the learner's answer. In order to generate the diagnosis, the **Diagnoser** employs some semantic evaluation services. The queries it sends to the semantic services depend on the semantics of the current task and the tutorial strategy.

A custom **Exercise Generator** might need to access the **Diagnoser** directly to define custom diagnostic queries. In order to achieve this, the default **Diagnoser** class can be extended by a custom class, which is defined as an inner class of a custom **Exercise Generator**, as shown in the Figure 4.2. This gives the custom **Exercise Generator** access to the extended **Diagnoser** that can be configured to send queries to the semantic services, that are specific for the given domain or task.

For instance, if the task in the exercise is to factorize the polynomial  $a^2 + ab - ab - b^2$ , the next step in the solution is the expression  $a(a + b) - b(a + b)$ , whereas if the task is to simplify this expression, the next step in the derivation would just be  $a^2 - b^2$ . Therefore, the queries to the evaluation services would have different values for the `context` parameter in case of these two tasks.

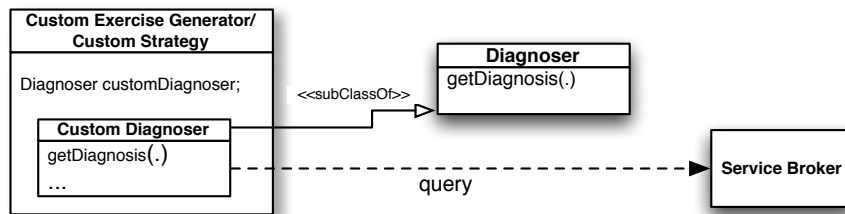


Figure 4.2: Defining a custom diagnoser

Similarly, a specific tutorial strategy can define a custom algorithm for feedback generation. A tutorial strategy can decide to learn more about the learner's error in order to generate the error feedback and in this case it will try to match the learner's answer against a buggy rule and issue a query

'getBuggyRules' or 'getUserSolutionPath', as described in Section 3.4.8. Another strategy can be satisfied with a simple correct/incorrect diagnosis in the first step, and then proceed by decomposing a task into simpler steps instead of trying to identify the error.

### 4.3.1 Employing Mathematical Services

We have designed a specific query format for the set of queries we use to access remote mathematical services, such as CAS or more sophisticated domain reasoners capable of automatic generation of different expert and erroneous solution paths for the problem. We use a set of domain-independent queries, as described in 3.4.5.

The semantic OPENMATH markup for mathematical formulas [Buswell et al., 2004] used by ACTIVEMATH and the generic format for queries to the semantic services support the interoperability of different CASs and reasoners when serving complex domains.

Currently, ACTIVEMATH integrates and communicates with the following CASs: YACAS, <sup>1</sup> Maxima, <sup>2</sup> and WIRIS; <sup>3</sup> phrasebooks for Maple <sup>4</sup> and Mathematica <sup>5</sup> are available too. Apart from these CAS, several domain reasoners are also connected to ACTIVEMATH and their services are available for ACTIVEMATH exercises.

In the architectures of the ITS systems, the domain reasoning engine is incorporated into the system, tightly communicating with the learner model and the teaching module. ACTIVEMATH instead uses domain reasoners as external remote services. In our case the `Diagnoser` component itself has no knowledge of the domain, but it uses educationally oriented queries to extract the needed diagnosis and other domain knowledge from the remotely connected reasoners. This gives us an opportunity to reuse the same custom `Exercise Generators` for multiple domains. As shown in Figure 4.4, there is one generic `Exercise Generator`, called `DRGenerator` for all exercises generated using the set of queries to a domain reasoner. `DRGenerator` automatically generates all the states and transitions of the exercise, using the queries to the domain reasoner for obtaining the diagnosis of the

---

<sup>1</sup>see <http://yacas.sourceforge.net>

<sup>2</sup>see <http://maxima.sourceforge.net>

<sup>3</sup>see <http://www.wiris.com>

<sup>4</sup>see <http://www.maplesoft.com>

<sup>5</sup>see <http://www.wolfram.com>

learner's answer and generating the feedback. Therefore, the most of the code for exercise generation can be reused for exercises in several domains. The `DRGenerator` has no knowledge of a concrete domain, but specific generators, extending the `DRGenerator` are defined for every new task. They are used for configuring the task generation and other parameters specific to the concrete topic of instruction.

The following domain reasoners have been connected to `ACTIVEMATH`: `SLOPERT` and `MATHCOACH` reasoners, implemented in `PROLOG` for the domain of symbolic differentiation, two domain reasoners for the domains of fraction arithmetics and integrals implemented as `YACAS` modules, several `IDEAS` domain reasoners for the domains of linear equations, propositional logic, higher order equations, symbolic differentiation, arithmetics, implemented in Haskell [Gerdes et al., 2008].

Figure 4.3 shows the different ways of connecting to external services that we realized within our framework. Such a variety of interfaces is due to the different technologies used by the external services.

A general utility class `OpenMathService` is defined for handling `ACTIVEMATH` queries. Furthermore, for each of the connected external systems, a specific class is created, that extends the `OpenMathService` by defining the client objects for the specific external service.

A `WIRIS CAS` server is connected to `ACTIVEMATH` via `XML-RPC` and contains an internal `OPENMATH` phrasebook. The `Maxima` server communicates via `WDSL` and the queries are piped through an external phrasebook. Connection to `WIRIS CAS` and `Maxima`, as well as phrasebooks for `Maple` and `Mathematica` was implemented within the EU-project `LEACTIVEMATH`<sup>6</sup>.

The `YACAS` server is a multi-threading web service built for the Computer Algebra System `YACAS`. It has native support for `OPENMATH` and can communicate with `ACTIVEMATH` directly via an internal `OPENMATH` protocol, as well as using an `HTTP` protocol. The `YACAS` server can be configured to be used for a general semantic evaluation, as well as loading additional domain reasoning modules for the domains of fractions and integrals, specifically developed to serve more detailed diagnosis and incremental problem solving in these domains.

Domain Reasoners `SLOPERT` and `MATHCOACH` are implemented as

---

<sup>6</sup>see <http://www.leactivemath.org>

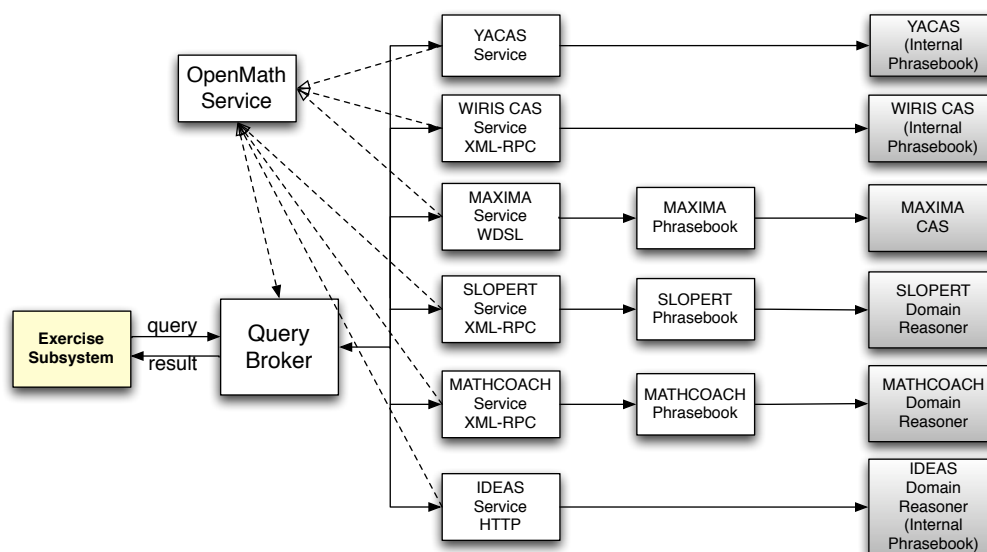


Figure 4.3: Semantic Service Broker

PROLOG programs running with SWI PROLOG<sup>7</sup> embedded into wrapper servers written in java to which ACTIVEMATH can connect via an XML-RPC protocol. The MATHCOACH server is a good example of a service that is running entirely independently from ACTIVEMATH and even receiving queries from clients other than ACTIVEMATH<sup>8</sup>.

The IDEAS reasoners are available as web services and most of them understand our OPENMATH format. They are connected to ACTIVEMATH via an HTTP protocol.

Some of the domain reasoners, such as YACAS domain reasoners for fractions and integrals, SLOPERT, and most of the IDEAS reasoners are capable of typical error diagnosis using buggy rules. Naturally, the buggy rules diagnosed by the domain reasoners have to be represented as nodes in the domain ontology of ACTIVEMATH and they must have some relation to the corresponding concepts. This is necessary in order to properly use the diagnosis information and to remediate the misconceptions of the learner.

The proper mapping of concepts and buggy rules between ACTIVEMATH and an external service is provided by a phrasebook - a routine attached to

<sup>7</sup>see <http://www.swi-prolog.org>

<sup>8</sup>see <http://mathcoach.htw-saarland.de>



the service translating the OPENMATH representation of the query input to the internal representation of the service and then translating the output of the service back to OPENMATH.

### 4.3.2 Semantic Service Broker

Queries to semantic services, issued by the **Exercise Subsystem** are received by a **Query Broker**, that distributes them to the corresponding semantic service. In order to select the appropriate semantic service, the **Query Broker** is using the *context* of a query, as defined in 3.4.5.

Each semantic service is configured in ACTIVE MATH such that it can answer within one or more *contexts*. Therefore, in order to select an appropriate service, the **Query Broker** is matching the given context of the query against the contexts of the services, configured in ACTIVE MATH and selects the one within the given context. If no service is found to answer the query, the service returns an empty result.

The queries to semantic services of ACTIVE MATH can have a composite context, that needs to involve several services. In order to ensure the interoperability of semantic services, the following conditions have to be satisfied by each semantic service, configured in ACTIVE MATH:

1. *If a service can answer the queries in the context  $C$  and an expression to evaluate has a form  $f \circ f_C$ , where  $f_C$  can be evaluated in the context  $C$ , then  $Ev_C(f \circ f_C) = f \circ (Ev_C(f_C))$ , where  $Ev_C$  is evaluation in the context  $C$ .*
2. *If no evaluation is possible, the service returns the input expression unchanged.*

The first condition means that the service knows how to recurse inside the unknown functions and tries to evaluate the arguments. Given that these conditions are matched, the composition of services can be achieved in the following way: if an expression has to be evaluated in the contexts  $C_1$  and  $C_2$ , it is first evaluated in the context  $C_1$ , then in  $C_2$  and then recurses until both services return unchanged expression.

### 4.3.3 Syntactic and numeric comparisons

Syntactic and numeric comparisons are realized as core functions within the `OpenMathService` class, so they do not require any remote mathematical

services.

Since mathematical formulas are represented in the XML format, the XML fragments are being compared in this case, which includes certain equivalence cases related to which objects are considered to be identical in XML. For example, the order of occurrence of attributes in an XML element makes no difference when comparing two elements. However, the order of occurrence of child elements does play a role, since the child elements could represent, e.g., the arguments of a non-commutative function.

Numeric equivalence is used for comparing numeric values and accepts an additional parameter `epsilon` with float values. This parameter defines the approximation tolerance threshold of the numeric comparison.

#### 4.3.4 Semantic evaluation using CAS

Global semantic evaluation is used to compare the learner's answer with the given reference value, as well as evaluating more complex constraints upon the learner's answer. In this case, the value of the *context* parameter is set to 'global' and the `Query Broker` automatically forwards the query to the CAS service. Figure 4.3 shows several CAS services that can be made available in `ACTIVEMATH`. Some CASs can be configured to be answering queries only in particular contexts, but there is only one CAS at a time configured for the global evaluation. The choice of a particular CAS for an `ACTIVEMATH` installation is a matter of tradeoff between the trust in quality of computations and the license price.

#### 4.3.5 Domain Reasoner Diagnosis

In order to obtain a detailed diagnosis of the learner's answer, the domain reasoners are used rather than just CAS. As described in 3.4.8, several queries can be used to obtain a detailed diagnosis. So, the query 'getUserSolutionPath' returns the shortest (possibly erroneous) path that the learner might have followed in order to obtain his answer from the given task, the query 'getBuggyRules' returns the list of buggy rules that are used for the generation of the erroneous solution path of the learner.

For each available domain reasoner service, a configuration entry is defined in `ACTIVEMATH` that specifies the *context(s)* supported by this domain reasoner. This information is used by the `Query Broker` when choosing the appropriate service for a given query.

## 4.4 Exercise Generator

Figure 4.4 shows an excerpt from the current hierarchy of **Exercise Generators** in **ACTIVEMATH**. An **Exercise Generator** is a component responsible for selecting or generating the next steps of an exercise based upon the diagnosis of the learner’s answer and the information available in the exercise graph.

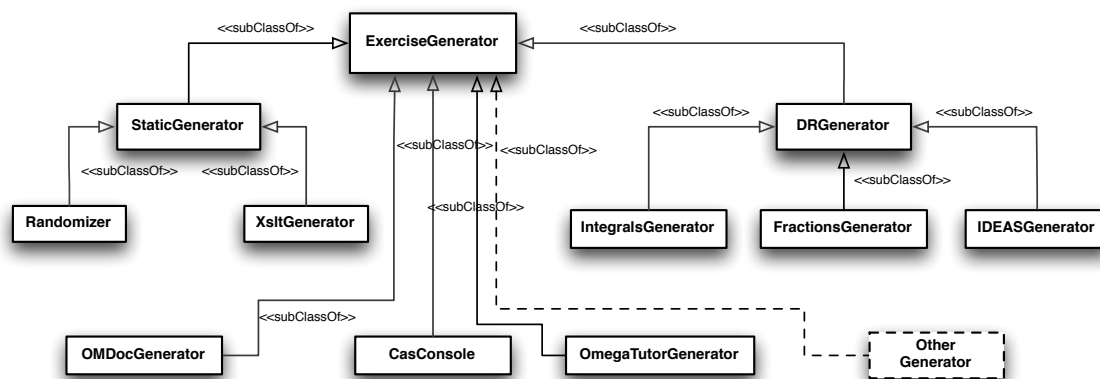


Figure 4.4: Excerpt from the hierarchy of **Exercise Generators**

In case of a fully authored exercises the **Exercise Generator** just selects the needed nodes from the given graph of the exercise. In case of a generated exercise, the states of the exercise are generated on the fly.

The **OMDocGenerator**, interpreting exercises represented in the **OMDOC Quiz** module format are not static, because the feedback nodes are composed on the fly, depending on the learner’s answer.

The **CasConsole**, shown in Figure 4.5, is a generator that provides a simple explorative tool for the user. The learner can enter mathematical expressions using a GUI and receives the feedback that is generated on the fly and represents the result of the evaluation of the entered mathematical expression in the Computer Algebra System.

The **OmegaTutorGenerator** is an **Exercise Generator** using the external proof assistant **OMEGA** in order to generate interactive exercises training the learner in assertion level proofs for the domain of basic set theory. For more information on this interactive tutor, see [Schiller, 2010].

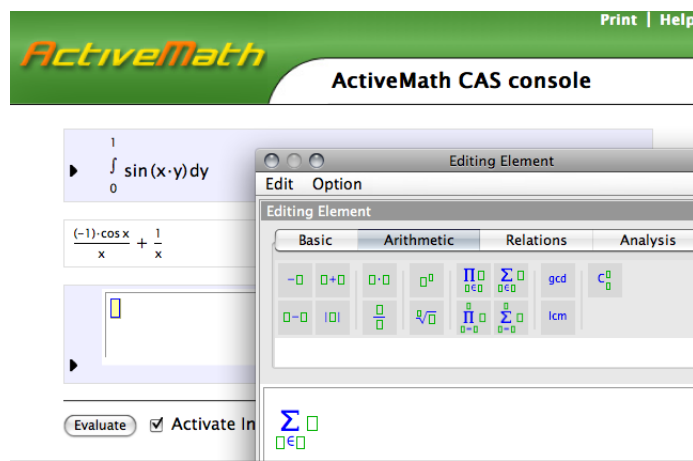


Figure 4.5: Explorative CAS Console in ACTIVEMATH

#### 4.4.1 StaticGenerator

The **StaticGenerator** is responsible for interpreting fully authored exercises and has extensions that handle parametrized exercises. The attribute 'Static' reflects the fact that the knowledge representation of the exercise is fixed and is not changing in the process of running the exercise.

The fully authored exercise contains all the information necessary for running it without generating any additional states or transitions. In this case the identifier of the next step is extracted from the transition that fires in the **Diagnoser**. The **Exercise Generator** receives the identifier of this next step, extracts it from the representation of the exercise and returns it to the **Exercise Manager**.

There are several extensions of the **StaticGenerator** that are commonly used in ACTIVEMATH. One of the extensions is the **Randomizer Exercise Generator**, considered in Section 3.5.1 that allows to define parametrized exercise representations to indicate the range of possible parameter values.

For more information on the functionality of the **Randomizer** and its extensions, see [Dudev and Palomo, 2007].

Another extension of the **StaticGenerator** is an XSLT generator, considered in Section 3.5.2. In the initialization phase it transforms the custom exercise format into the ACTIVEMATH format using an XSLT stylesheet and calls the parent **Static generator** to run the resulting exercise.

### 4.4.2 Domain Reasoner Exercises

Another commonly used `Exercise Generator` is a `DRGenerator`, which provides an additional infrastructure to be used by the `Exercise Generators`.

The `DRGenerator` is a generic class facilitating domain reasoner-powered generation of interactive exercises in several domains, such as fraction arithmetics, linear and non-linear equations, derivatives and integrals, etc. This `Exercise Generator`, naturally, covers only a restricted amount of tasks in the listed domains. It implements a general structure of a simple multi-step exercise, that involves calculation. This `Exercise Generator` has several subclasses, specializing on concrete tasks in given domains.

The `DRGenerator` generates exercise building blocks and defines a general problem solving strategy. This problem solving strategy can be overwritten by a more specific one, if necessary.

Figure 4.6 presents a sample feedback, that can be generated using a domain reasoner.

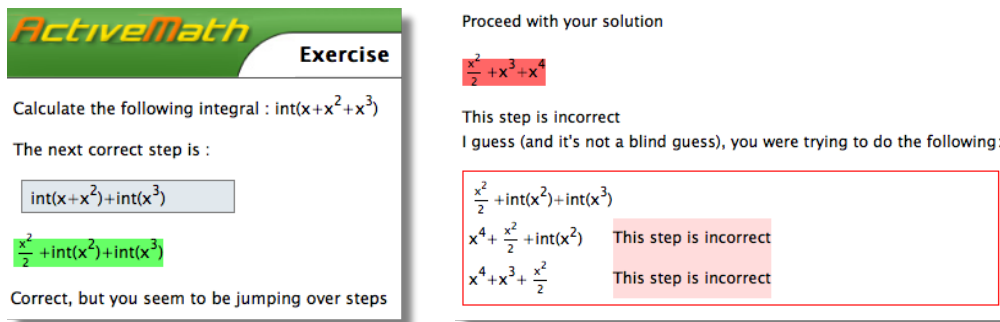


Figure 4.6: Domain reasoner-powered feedback

In the screenshot on the left the reasoner can detect a "jumping step", if more than one rule application has been applied in a step. The threshold of how many steps the learner is allowed to do at once can be controlled by a tutorial strategy. As defined in 3.4.5, each query can have a parameter that contains the number of iterations to be made by a reasoner. By varying this number we can define the granularity of steps. In the screenshot on the right the minimal error feedback is presented, in which the domain reasoner is trying to reconstruct the erroneous path the learner is following, and indicates which steps are incorrect in this path.

## 4.5 Exercise Events

Exercise events are part of the general `ACTIVEMATH` event architecture, that views different `ACTIVEMATH` components as Web-services and implements the communication between them using a specific messaging architecture. In this approach, all components report on their activities to one common pool of events. A component interested in receiving events from another component is registering its interest and the messaging system is propagating the needed events to the interested component. For more information on this approach, see the section "Communication of Components and Services" in [Melis et al., 2006].

The `Exercise Subsystem` produces three types of events. At the initialization time of the exercise, the `ExerciseStartEvent` is fired, signaling that the exercise has started. The exercise ID and other initial parameters are included into the event. Each time the learner performs an exercise step, the `ExerciseStepEvent` is sent. This event includes metadata of the task, such as the relationship to the domain concept, competency and difficulty of the task, as well as diagnosis of the learner's performance and possibly identifiers of diagnosed misconceptions. After the exercise is finished, the closing `ExerciseFinishedEvent` is published.

Each event includes a time-stamp, which allows to calculate durations of steps and of the whole exercise. Just as all the other events generated by other components, the exercise events are published at a central location. Other components, interested in receiving particular events, have to "subscribe" for these events.

The `Student Model` of `ACTIVEMATH` receives the events issued by the `Exercise Subsystem`. After each `ExerciseStepEvent` the `Student Model` receives new evidence for updating the learner's mastery of domain concepts.

## 4.6 Local Student Model

In order to react to the student's progress and actions a `Local Student Model` is used. It keeps track of visited exercise nodes, the number of times they were visited, the record of visited hints for each interaction, and other information locally available within the exercise session.

The tutorial strategies use the `Local Student Model` by dealing with repeated trials of steps, giving sequenced hints and changing behavior de-

pending on the history of the learner's actions.

It is important to note that by default not all the variables are active in the `Local Student Model`. Each variable defined there is activated by the strategy that needs to use this variable. This way we prevent the `Exercise Subsystem` from storing and constantly updating unused information.

Consider a strategy, which gives the student a certain number of trials for solving each exercise step. In case of a wrong answer, the strategy presents flag feedback to the learner and goes back to the same step to let the student try again. After the fixed number of trials, the correct solution is presented. This strategy needs to keep track of the number of trials of the same exercise step, which is represented by a variable in the `Local Student Model`. This variable can be used by all tutorial strategies that need to count the number of trials of a step.

The `Local Student Model` is used to generate the learner's score for the exercise, since it has all the runtime information about learner's progress in the exercise. The score provides an average success rate of the learner for this exercise.

The default formula for computing the average score is defined as follows:

If  $N$  is the amount of steps,  $n(k)$  is the amount of trials of the  $k$ -th step (for  $k = 1..N$ ), and  $Success(i, n(k))$  is the success rate of the  $i$ -th trial of the  $n(k)$ -th step, then the total average success rate  $S$  is equal to

$$S = \frac{1}{N} \cdot \sum_{k=1}^N \left( \frac{1}{n(k)} \sum_{i=1}^{n(k)} Success(i, n(k)) \right)$$

The algorithm for computing the score can be customized by the assessment strategy or any other tutorial strategy that uses the `Local Student Model`.

## 4.7 Applying Tutorial Strategies

Tutorial strategies are special `Exercise Generators` that modify the exercise graphs according to the tutorial behavior of this strategy.

In order to apply a tutorial strategy to an exercise, the strategy generator is composed with the `Exercise Generator` of the given exercise and overwrites its functions. As shown in Figure 4.7, some strategies are only applicable to static exercises, so they just extend the `StaticGenerator`. Other

strategies that are applicable to all kinds of exercises extend the base class `Exercise Generator`.

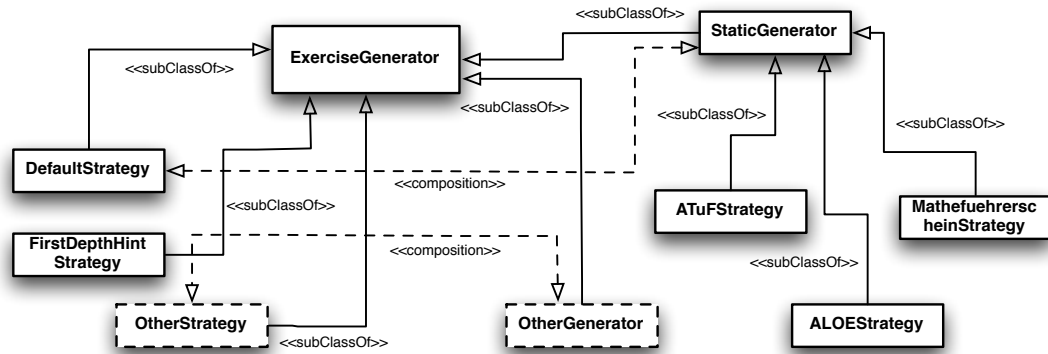


Figure 4.7: Applying Tutorial Strategies

The `Exercise Subsystem` includes a growing set of tutorial strategies. These strategies can be further specialized and composed with each other in order to form more complex ones.

## 4.8 Presentation Strategies

Depending on the educational approach and the domain of instruction, it is often useful to have a certain graphical presentation for the exercise. The presentation can influence all parts of the exercise: the learner interaction interface, the feedback form and layout, the coloring schemes for instruction elements and others.

In order to allow for different presentation, we have introduced an additional transformation by a presentation strategy, which is applied to the exercise steps, before sending them to the `User Interface`.

The presentation strategy is applied to the exercise nodes before they are processed by the `Presentation System`. These exercise nodes contain semantic annotations such as the separation into the interactions and feedback. The presentation strategy can, e.g., choose to present all the feedbacks in the immediate vicinity of the learner's answer.

In Figure 4.8, the screenshot on the left assumes the existence of hints and solutions and reserves the bottom right corner of the screen for it. The



presentation strategy on the right does not divide the screen into regions. It just provides all the feedback in a vertical sequence. So, all the feedbacks, hints and solutions, are handled in the same way.

Figure 4.8: Custom feedback layout

In the strategy on the right, the answers of the learner can be decorated using red, yellow and green colors in the cases of wrong, almost correct, and entirely correct answers, respectively. Those parts of the exercise that contain the learner interaction results are highlighted by thin borders. All the feedback of the system and the learner interaction stays on the screen even after the interaction and it is presented sequentially.

The more complex strategy on the left shows the exercise divided into two parts: the first part is multiple choice interaction presented by the default presentation strategy. The second phase is a sub-exercise with free input, where the learner interaction appears in the left part of the screen, and the feedback of the system, hints of different types and the correct solution are presented in the right part of the screen. The second phase consists of several steps but as soon as the learner makes one step and moves to the next one, only the hints and feedback, as relevant to the current step stay on the screen, the previous steps are folded away. At the same time, the first phase of the exercise always stays visible for each step in the second phase, since it is a part of the presentation strategy.

Flag feedback can be presented by coloring correct parts of the learner's answer green and incorrect ones red. Alternatively or in addition to colors, different smiley icons can be used. Another possibility of marking without colors is by checks and crosses, or by textual feedback. In Figure 4.9, the above mentioned decoration styles are shown.

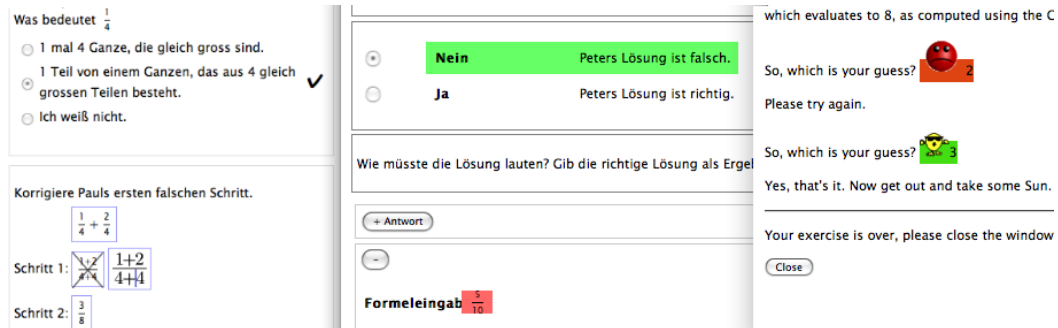


Figure 4.9: Different presentations of flag feedback in ACTIVE MATH

The user interface to the exercise can look very different for different types of interactions and presentation strategies. If the learner is supposed to receive immediate feedback in each step, it is enough to provide him with the interactive elements that are needed in the current step. Interaction interfaces for the case of immediate feedback are shown in Figure 4.10. In both cases the learner has to fill in the fixed number of blanks.

However, if the learner is supposed to input the complete solution, a more complex interaction interface is needed. Figure 4.11 shows a simple and more complex (multiple interaction) interface, that was used for the ALOE and ATuF research projects. On the right is a simple interface, which allows the learner to enter as many intermediate steps as necessary for his solution. More steps can be added using the '+' button and the unnecessary steps can be deleted using the '-' button. There is one reserved field for a final answer. On the left of Figure 4.11 a more complex interface is shown. Here, the learner can not only add steps, but he can also decide what type of step he wants to perform, which domain concept to use or which rule to apply. The list of alternatives is suggested to the learner in a drop-down menu. After the learner chooses one of the items from the list, a structured template for the interaction appears, where the learner has to fill in the blanks. As in the case on the right, there is one reserved field for a final answer. An additional GUI element of this presentation strategy is a so-called "confidence slider".

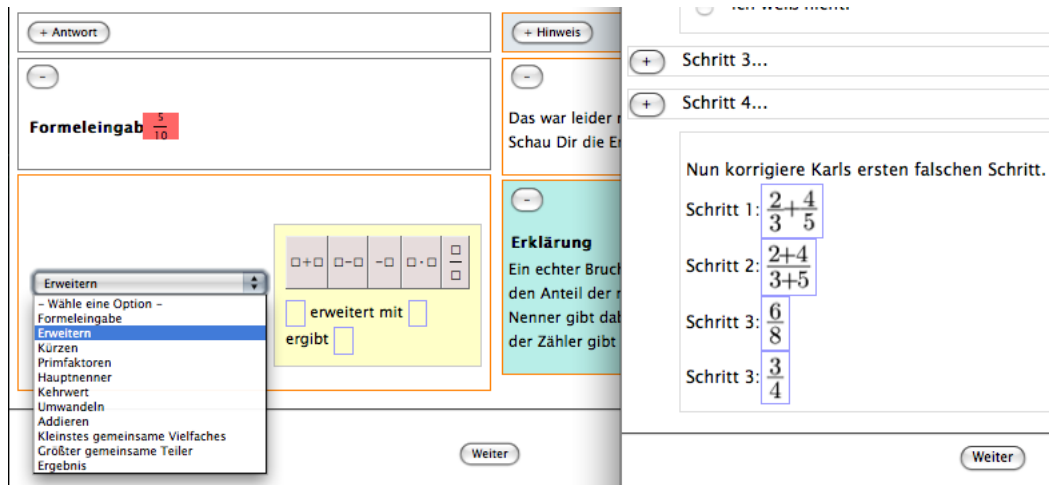


Figure 4.10: Two interaction interfaces

It is a horizontal slider bar, by moving which the learner can specify how confident he is about his answer.

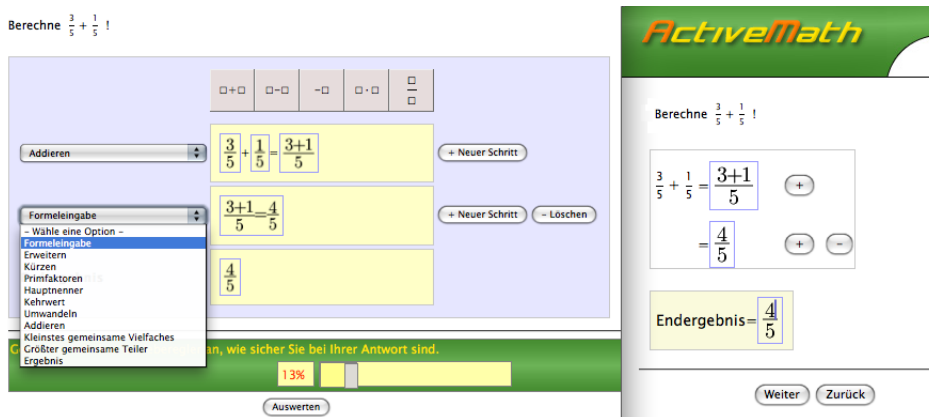


Figure 4.11: Different multiple interaction interfaces



# Chapter 5

## Exercise Authoring

In this chapter we describe our tools for the authoring of interactive exercises for ACTIVE MATH. They either support authoring hand-crafted exercise graphs or support the automatic generation exercises or parts thereof with the help of powerful domain reasoning tools.

### 5.1 Visual "Authoring-by-Doing" Tool

A visual "authoring-by-doing" tool EXAMAT for exercises in ACTIVE MATH has been implemented in the ACTIVE MATH group, under the supervision of the author. This tool provides a visual user interface for authoring exercises step by step and it plays the completed parts of the exercise. We call such exercise creation procedure "authoring-by-doing" because it is similar to actually solving the exercise following all possible solution paths. The author can create an exercise step, then input several possible answers of the learner to this step, which will trigger the creation of the consequent steps, following the corresponding answers, and so on. This way the exercise solution space is created by simulating the behavior of potential learners. This procedure is similar to authoring behavior graphs in CTAT Cognitive Tutors Authoring Tools [Koedinger et al., 2004].

EXAMAT offers WYSIWYG authoring of the representation of multi-step exercise and its visualization as a graph. The author can visually manipulate the graph of an exercise, edit the nodes and edges of an exercise graph and run the edited exercise at each step of its creation. EXAMAT combines features of systems such as Maple TA <sup>TM</sup> [MapleTA <sup>TM</sup>, 2010] for

creating tests and assignments and CTAT Cognitive Tutors Authoring Tools, cited above.

The user interface for editing steps of an exercise with interactive elements enables a semantic evaluation of the learner's answer using a Computer Algebra System (CAS). It can create parametrized exercises using randomized variables in the exercise solution space.

The EXAMAT tool is browser-based and as it is embedded into the ACTIVEMATH learning environment, it gives, among others, easy remote access to an online authoring environment.

The main window of EXAMAT tool consists of 3 main frames, shown in Figure 5.1. On the right is the visualization of the exercise structure in form of a graph, the vertexes of which represent the nodes of an exercise and the edges represent the transitions between these nodes. The upper frame on the left is for editing the nodes of the exercise and the lower frame is for editing the transitions.

| Type        | User input          | Transition to          | Edit   | Delete | Move      |
|-------------|---------------------|------------------------|--------|--------|-----------|
| Conditional | [4·1]               | fdk_umf_quad_r         | [Icon] | [X]    | [Up/Down] |
| Conditional | [4·1 <sup>2</sup> ] | fdk_umf_quad_f_flaeche | [Icon] | [X]    | [Up/Down] |
| Conditional | [1 <sup>3</sup> ]   | fdk_umf_f_volumen      | [Icon] | [X]    | [Up/Down] |
| Default     | ----                | fdk_umf_quad_default   | [Icon] | [X]    | [Up/Down] |

Figure 5.1: Exercise authoring user interface

A sample graph of the exercise nodes and transitions between them is shown in Figure 5.2. By right-clicking on the node of the graph the context

menu is called, in which the author can select one of the options - edit the node, running the exercise, or deleting this node.

When deleting an exercise node all the transitions to and from this node are deleted as well. When selecting the edit option, the node is loaded into the node editor window and the transitions issued at this node are also loaded into the transition editor window.

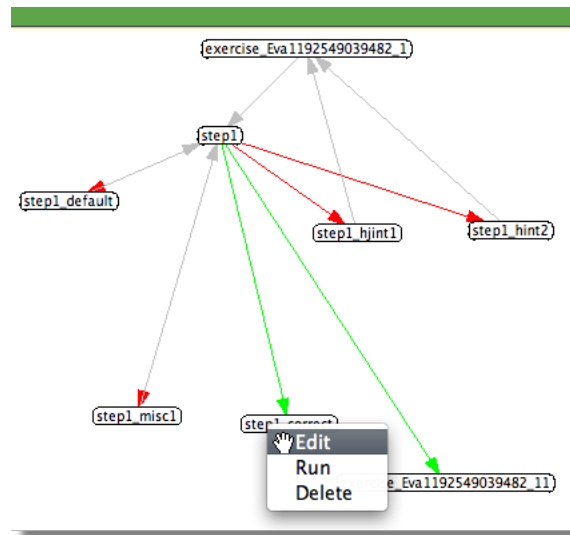


Figure 5.2: Sample exercise graph

### 5.1.1 Editing Exercise Nodes

The exercise nodes are authored within the node editor, that is shown on the top left in Figure 5.1. The author can enter text with formulas, images, tables and other formatting elements. He calls a popup microstructure editor window by clicking on the 'text' button in the sidebar. The microstructure editor window is shown in Figure 5.3.

The author can enter text in several languages by switching to the corresponding tab in the microstructure editor. The formulas are edited in a separate input field on the top left, using the embedded formula editor. In order to insert formulas, created in the formula editor field, the author has to press the  $\Sigma$  button in the toolbar. The WIRIS<sup>1</sup> formula editor, used

<sup>1</sup>See <http://www.wiris.com/>

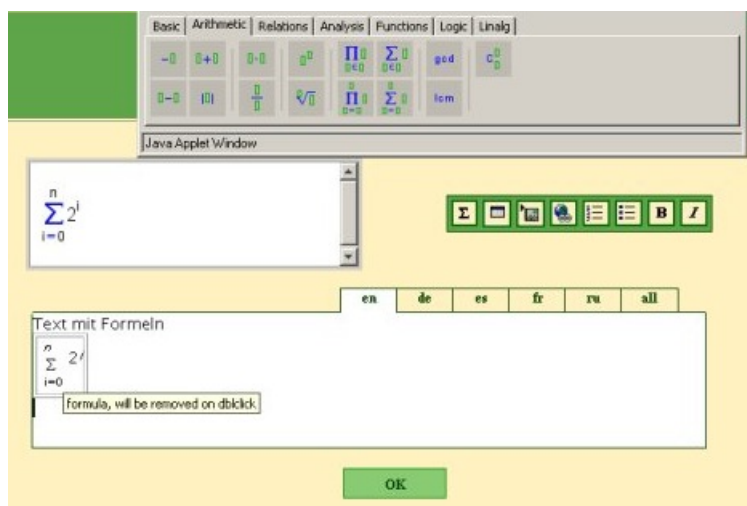


Figure 5.3: Microstructure editor window

in **ACTIVEMATH**, produces a semantic representation of mathematical expressions that make it possible to evaluate these expressions with external mathematical tools.

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 used too: for authoring blanks, the formula editor is used with the special blank symbol in the palette. The formula editor is necessary in cases of dealing with blanks inside complex formulas. For authoring selections, the microstructure editor is used as well.

### 5.1.2 Editing Transitions

In order to author a transition, an author has to use the bottom left panel of the **EXAMAT** tool. There he/she has the choice to define transitions of different type that lead to a new or to an existing step. Conditions can be conditional, default, hint request, or unconditional.

The body of a transition is authored using the **transition editor**, that pops up in a new window, and the existing transitions can be viewed in the main **EXAMAT** window in the bottom left panel body, as shown in





Figure 5.4: Authoring blanks and selections

Figure 5.1.

Figure 5.5 shows the GUI of the transition editor for authoring of a transition for a fill-in-blank interaction. This transition contains a link to a target step, diagnosis information such as correctness of the learner's answer (element *achievement*), and a reference to misconceptions, if any. The author can insert additional feedback into the target exercise node of the transition.

Finally, the condition is authored, for the comparison of the learner's answer to the control expression entered by the author. A comparison can be performed within a mathematical context, as described in Section 3.4.3.

On the right of Figure 5.5 a default transition is presented. This transition does not define any condition, it fires if none of the conditional transitions apply. This condition leads to an existing step, whose identifier is selected from a dynamically generated drop-down menu containing identifiers of all existing steps.

### 5.1.3 Adding Metadata

Popup interfaces for authoring metadata of the relevant parts of the exercises are available. Figure 5.6 shows a part of the interface for authoring exercise metadata. Some metadata, such as *competency* or *difficulty*, can be entered by selecting the values from drop-down menus. Other metadata, such as, e.g., a relation to the focus concept has to be entered in the input field.

The figure shows two dialog boxes for defining transitions. The left dialog is titled "Conditional transition" and the right is "Default transition".

**Conditional transition:**

- Target Step:** (new step will be created) step1\_correct
- Diagnosis:**
  - Misconceptions id: [ ]
  - Achievement (a number between 0 and 1): 1.0
- Additional feedback:**
  - Keep user input:
  - Decorate answer:
- Condition:**

| Compare User input to : | Comparison type   |
|-------------------------|---|
| 3<br>5                  | <input type="radio"/> Syntactic <input checked="" type="radio"/> Semantic <input type="radio"/> Numeric |

**Default transition:**

- Target Step:** exercise\_Eval1197883642126\_1 (dropdown menu)
- Diagnosis:**
  - Misconceptions id: [ ]
  - Achievement (a number between 0 and 1): 0.0

Figure 5.5: Defining transitions

The figure shows a dialog box for authoring relevant metadata. It contains several sections, each with a dropdown menu and a small icon (a square with a plus sign and a minus sign):

- Competency:** solve | apply\_algorithms
- Competencylevel:** simple\_conceptual
- Typical learningtime:** [ ]
- Learningcontext:** primary\_education
- Difficulty:** very\_easy
- Abstractness:** concrete
- Relation:** mbase://openmath-cds/arith1/divide

At the bottom of the dialog is an "OK" button.

Figure 5.6: Authoring relevant metadata

### 5.1.4 Authoring Parametrized Exercises

EXAMAT provides the possibility to employ a Randomizer generator (see Section 3.5.1) for parametrizing the exercise task and the solution space.

Using the pop-up interface shown in Figure 5.7 the author can define custom variables together with sets of possible values and use these variables in the authored exercise. As soon as the variables are defined they appear as special symbols in the palette of the formula editor and they can then be used within formulas in all relevant parts of the exercise tasks, feedbacks, and conditions.

| Variables   | Values   |
|---|--|
| <input type="checkbox"/> Evaluated<br><input type="text" value="A"/>            | <input type="checkbox"/> +<br><input type="checkbox"/> - 1<br><input type="checkbox"/> - 2 |
| <input type="checkbox"/> Evaluated<br><input type="text" value="B"/>            | <input type="checkbox"/> +<br><input type="checkbox"/> - 3<br><input type="checkbox"/> - 4 |
| <input checked="" type="checkbox"/> Evaluated<br><input type="text" value="C"/> | <input type="checkbox"/> +<br><input type="checkbox"/> - A+B                               |

**Add variable**  +

Figure 5.7: Authoring Randomizer variables

Since not only numbers are possible values of variables but any mathematical expression, variables can be composed of other variables, as shown in Figure 5.7. If such a variable is set to be evaluated, using the corresponding checkbox in the interface, the value of the variable will be evaluated in the CAS before the instantiation.

For example, if the exercises  $A$ ,  $B$  and  $C$  from Figure 5.7 are used in an exercise, then if after instantiation the values of  $A$  and  $B$  are 2 and 3

respectively, the value of  $C$  is 5. If the variable  $C$  was not marked to be 'evaluated', the value at instantiation would be a numeric expression  $2 + 3$ .

### 5.1.5 Comparison to Related Tools

The authoring-by-doing procedure of EXAMAT is similar to the behavior recorder of the CTAT tool. One of the major differences to CTAT is that when authoring so-called Pseudo-Tutors, the author has to enter all possible correct answers, creating an edge for the behavior graph for each possible answer. In EXAMAT one can author one edge for a whole class of equivalent answers.

The interface of the microstructure editor is similar to the authoring tool in Maple TA <sup>TM</sup>. The difference is, however, the underlying semantic representation of mathematical formulas, which can be evaluated with several different computer algebra and other mathematical reasoning systems, provided the mapping from and to the OPENMATH syntax (phrasebook) is implemented.

### 5.1.6 Further Development

The exercise authoring tool does not yet fully support all the expressive power of the exercise language. For example, an interface is still to be implemented for authoring of complex constraints. This is needed, e.g., for multiple answers in one interaction that cannot be evaluated component-wise but only as a whole.

The exercise editor has to be extended to support an extended version of the Randomizer, described in [Dudev and Palomo, 2007]. It is able to randomize over continuous or discrete intervals and the set of elementary functions and their compositions.

## 5.2 Exercise Generation using Domain Reasoners

As we have seen in Section 4.4.2, the exercise system of ACTIVEMATH allows for exercises, in which each step is generated on the fly by an DRGenerator. Subsequent steps are generated based on the results of the learner's interaction with the system in the previous step.

In the following sections we describe examples of different mathematical domains, for which we provide support because we already have domain reasoners for those domains integrated with `ACTIVEMATH`.

### 5.2.1 Selecting Tasks for Problems

Selecting tasks for problems is a delicate issue, where human tutors still need to be heavily involved. Although domain reasoners are programmed to solve classes of problems in a given domain and although they are able to randomly generate problems, it does not yet mean that these problems would be pedagogically suitable for the current learner.

Human tutors can manually select pedagogically valuable tasks from existing sources, such as textbooks or lecture scripts. These manually selected tasks can be annotated with metadata and attached to the `Exercise Generator` powered by a domain reasoner.

Moreover, the task selection can be a semi-automatic process, in which the possible tasks are generated automatically and reviewed by the tutor, and then saved in the database after approval of the tutor. An example of such a mixed initiative authoring process is shown in Figure 5.8. This figure shows a small tool for authoring the parameters for exercises to be generated automatically using domain reasoners. This tool is developed in `ACTIVEMATH` and is described in Section 5.2.2 below.

So far we have used the following methods for authoring tasks:

1. Generating tasks with the help of a domain reasoner (parametrized by difficulty and other metadata)
2. Teachers create lists of pedagogically valuable tasks and annotate them with metadata in order to enable metadata-based selection
3. Teachers define parametrized tasks, in which parameters are instantiated by random parameters.
4. Students can define their own task (explorative scenario)

### 5.2.2 Authoring Parameters for Exercise Generation

All `ACTIVEMATH` exercises, generated with the help of Domain Reasoners share the common architecture of the `DRGenerator`, while addressing different tasks in different mathematical domains. In order to facilitate exercise

generation and in order to give more control of the generation process to the teacher, we have designed a small form-based authoring tool within ACTIVE MATH, as shown in Figure 5.8. This tool helps to provide the necessary parameters to the DRGenerator, such as a domain of instruction and a task, encoded with the `context` parameter, as well as some other metadata. Moreover, an author can generate an exercise task in a given context automatically, using a remotely connected Domain Reasoner. If the author does not like the generated task, the he can keep generating more tasks, until the suitable one is selected. Finally, after selecting the task and authoring other parameters, the author can let the system generate an exercise, prepared to run within ACTIVE MATH and save it in the database.

|   |   |
|---|---|
| Enter the exercise invitation text  | <input type="text" value="try this exercise"/>                    |
| Enter the exercise problem statement  | <input type="text" value="solve this:"/>                          |
| Enter the ID of the focus concept<br>(e.g. mbase://openmath-cds/calculus1/diff) | <input type="text" value="mbase://openmath-cds/calculus1/diff"/>  |
| Enter the wished exercise difficulty  | <input type="text" value="easy"/>                                 |
| <input type="button" value="Generate a task"/>                                  | $\frac{d}{dx} x^2$  |
| Select the math context   | <input type="text" value="calculate derivative"/>                 |
| Select target collection  | <input type="text" value="The Content Collection LeAM_calculus"/> |
| <input type="button" value="Generate Exercise from form values"/>               |   |
| Result Exercise:  | <input type="text" value="..."/>                                  |

Figure 5.8: Authoring Domain Reasoner Exercises

### 5.2.3 Prolog-based Domain Reasoners

The first domain reasoners connected to ACTIVE MATH were implemented in PROLOG - a symbolic programming language <sup>2</sup>.

<sup>2</sup>See, e.g. <http://www.swi-prolog.org/>

### 5.2.3.1 SLOPERT

SLOPERT is a powerful domain reasoner to serve interactive exercises in the domain of symbolic differentiation. This was implemented within the LEACTIVE MATH consortium for the needs of LEACTIVE MATH project, aiming at tutorial dialogues in the domain of symbolic differentiation [Zinn, 2006]. This reasoner is rule-based and uses a number of expert and buggy rules to generate a hierarchical solution space of an exercise.

The integration into ACTIVE MATH was successful, but the exercises, produced with the help of this reasoner had one limitation - the reasoner accepted only answers, free of a differentiation operator. It could accept answers to the whole task or any of the subtasks, but not an intermediate answer, containing differentiation in a subexpression. For example, when differentiating the function  $f(x) = 2 \cdot x + x^2$ , the answers  $2$ ,  $2 \cdot x$  and  $2 + 2 \cdot$  would be accepted as correct steps, but the expression  $(2 \cdot x)' + (x^2)'$  was not recognized.

Another difficulty in using this and other PROLOG reasoners is that the condition necessary for interoperability with other reasoners, as specified in the section 4.3.2 is not satisfied. As soon as the PROLOG engine sees an unknown expression, it returns this expression unchanged, not trying to recurse inside.

### 5.2.3.2 MATHCOACH

MATHCOACH is a family of domain reasoners developed at the Hochschule für Technik und Wirtschaft (HTW) in Saarbrücken [Grabowski et al., 2005]. It was implemented taking SLOPERT as a model. As opposed to SLOPERT, the MATHCOACH reasoner is stateless. In each query we have to set the new task for the reasoner. MATHCOACH does not encode buggy rules and can not track the student's solution.

MATHCOACH implements reasoners in several domains, but currently we have integrated only the reasoner for symbolic differentiation.

An explorative dialogue strategy has been implemented by the author for MATHCOACH-powered exercises, called *Explorative Differentiation Console*, similar to the explorative CAS console, described in Section 4.4. An example of such an exercise dialogue is shown in Figure 5.9.

The student chooses a function that he wishes to differentiate, and the system assists in solving this problem by evaluating the correctness of every step the student makes and provides three levels of hints for each step. First,

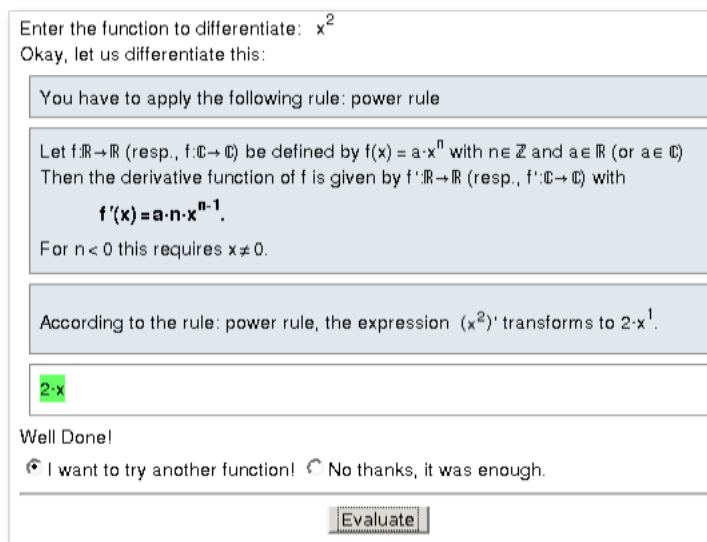


Figure 5.9: Explorative Domain Reasoner Console

a general procedural hint is generated, telling which differentiation rule has to be applied. It is followed by a conceptual hint, giving the definition of the needed rule to be applied. Finally, the bottom-out hint presents the result of the application of the needed rule to the expression in the current step. If the final step of the solution is reached, the system asks the student whether he would like to stop his exercise or differentiate another function.

### 5.2.4 YACAS Reasoners

The computer algebra system YACAS has been adapted by A. González Palomo, a member of the ACTIVEMATH team, to serve as the generic semantic evaluation engine for ACTIVEMATH. YACAS was chosen because of its modular architecture, that allows to restrict the reasoning engine to particular contexts. Other Computer Algebra Systems such as Maple, Maxima or Mathematica do not have a standardized way of restricting the evaluation context to a particular set of concepts, as discussed in Section 3.4.3. Another reason to use YACAS was, that it satisfies interoperability conditions, as specified in Section 4.3.2.

The integration of YACAS into ACTIVEMATH consisted of implementing native support for the OPENMATH format, and implementing a YACAS web-



server, that accepts semantic queries.

The service can be parametrized by the mathematical context, which restricts the evaluation engine to only tackle problems in this context.

Currently there are three main contexts within which YACAS is serving ACTIVEMATH. First of all, YACAS is used as a main semantic evaluation engine for CAS exercises. Furthermore, there are two Domain Reasoners, currently developed as YACAS modules (described below). The author supervised the implementation of those reasoners.

#### 5.2.4.1 Domain Reasoner for Fraction Arithmetics

A domain reasoner for fraction arithmetics has been developed as a Master's project at the Saarland University under the supervision of the author. One of the main requirements for the reasoner was to answer the set of queries, described in Section 3.4.8.

The domain reasoner consists of a package of scripts for the YACAS system, that has been installed as a module on the YACAS server, answering queries in the context of fractions and defining several sub-contexts, such as fraction reduction, or fraction addition contexts.

This domain reasoner is rule-based and encodes a number of expert and buggy rules, adopted mainly from [Henneke, 1999]. For a full presentation of this work, see [Masood, 2009].

Figure 5.10 shows a typical exercise automatically generated by this domain reasoner. Here, the informative error-related feedback is generated using a template attached to the corresponding buggy rule.

#### 5.2.4.2 Domain Reasoner for Integrals

Another domain reasoner for the domain of integrals has been developed as a YACAS module. This work was done by Anatoly Belchusov. Our goal was again to ensure that the set of queries, defined in Section 3.4.8 is answered by the service.

A rule-based reasoner for the domain of function integration has been implemented, including a large number of expert and some buggy rules. The reasoner has been tested with a set of commonly practiced integrals, taken from a schoolbook. As a side product of this implementation, some errors in the main YACAS module which handles integrals were found and fixed. Also, more expert rules for table integrals were introduced.

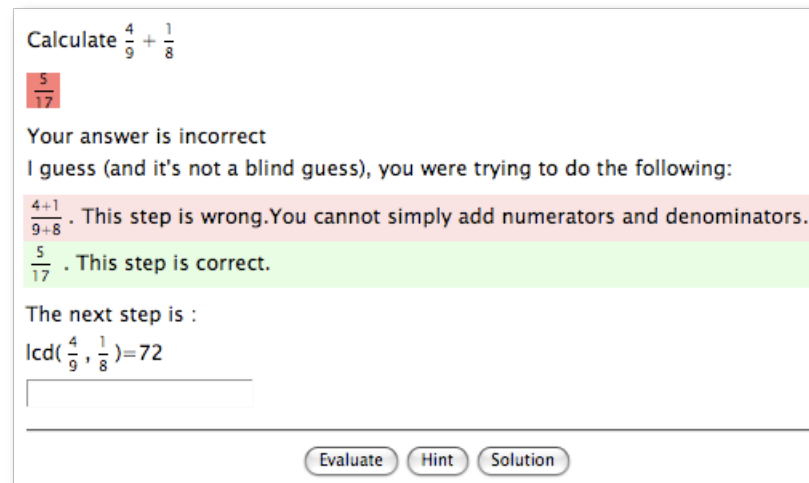


Figure 5.10: Custom error feedback in fraction exercises

Some examples of feedback, generated with the help of this reasoner are shown in Figure 4.6 in section 4.4.2. The reasoner is able to variate the granularity of steps, detect equivalent suboptimal solution paths, as well as attempt to reconstruct student's erroneous solution path.

### 5.2.5 Haskell-Based Domain Reasoners

A number of domain reasoners for various mathematical domains have been developed within the IDEAS project at the Open University of the Netherlands (OUN), in the group of Prof. Johan Jeuring<sup>3</sup>, who collaborates with us in the EU-project MATHBRIDGE.

These domain reasoners have been developed in the programming language Haskell and can answer a set of queries defined in ACTIVEMATH.

This made it easy to deploy these reasoners as semantic web-services serving interactive exercises in our system as well. The growing set of IDEAS Domain Reasoners cover such topics as linear, quadratic and higher order equations, systems of linear equations, different problems in the domain of arithmetics, propositional logic, derivatives, and others.

All the IDEA Reasoners are integrated with ACTIVEMATH using the same Exercise Generator program, although the tasks can be as differ-

<sup>3</sup>See project page under <http://ideas.cs.uu.nl/>

ent as solving equation, differentiating a function, simplifying expression, or bringing a logical formula into its disjunctive normal form. The only difference between the instances of the program, generating the three exercises shown in Figure 5.11 is the value of the `context` parameter, pointing to the concrete mathematical context of the task in a corresponding domain.

Solve the following equation :  $2 \cdot 5 - 3 \cdot x = 6 - x$   
The next correct step is :

$10 - 6 \cdot x = 6 - x$

$-5 \cdot x = 6 - 10$

This step is correct.

Proceed with your solution

Evaluate
Hint
Solution

Solve the following equation :  $x^3 - 5 \cdot x^2 - 6 \cdot x = 0$

$(x = 0) \vee (x^2 - 5 \cdot x - 6 = 0)$

This step is correct.

Proceed with your solution

The correct solution is :

Step 1  $(x = 0) \vee ((x + 1) \cdot x - 6 = 0)$

Step 2  $(x = 0) \vee (x = -1) \vee (x = 6)$

Bring to the dnf form:  $\neg(r \wedge q) \rightarrow \neg(p \rightarrow q)$

$\neg\neg(r \wedge q) \vee \neg(p \rightarrow q)$

This step is correct.

Proceed with your solution

The correct solution is :

Step 1  $\neg\neg(r \wedge q) \vee \neg(\neg p \vee q)$

Step 2  $\neg\neg(r \wedge q) \vee (\neg\neg p \wedge \neg q)$

Step 3  $(r \wedge q) \vee (\neg\neg p \wedge \neg q)$

Step 4  $(r \wedge q) \vee (p \wedge \neg q)$

Figure 5.11: IDEAS Domain Reasoner exercises use the same generator



# Chapter 6

## Evaluation of coverage and performance

In this chapter we briefly describe a coverage of the **Exercise Subsystem** of **ACTIVEMATH** and provide some results of technical performance evaluation.

### 6.1 Range of Expressive Power

We have investigated the expressive power of the **ACTIVEMATH** exercises with respect to the coverage of mathematical domains and pedagogical approaches. This expressivity was driven by projects and pedagogues.

We describe the mathematical domains, for which the exercises have been authored and tested, targeted audiences of learners, different tutorial and presentation strategies for exercises that were realized within the projects.

For each content collection, Table 6.1 shows the total numbers of interactive exercises and the corresponding total number of exercise states. The total number of exercise states is important since each state of the exercise informs the **Student Model** of **ACTIVEMATH** and contributes to the training and the assessment of the learner's mastery of the domain concepts.

Table 6.1 presents the summary of most of the strategies that were implemented by the author and used within the above mentioned projects. Note that the strategies were implemented in order to define specific tutorial behavior of the exercises. But the exercises can be reused without this specific behavior and run with the default strategy.

| Content Package       | Number of exercises | Total number of exercise nodes |
|-----------------------|---------------------|--------------------------------|
| LeActiveMath calculus | 473                 | 6243                           |
| AMOR                  | 552                 | 4439                           |
| School Mathematics    | 885                 | 2889                           |
| Matheführerschein     | 315                 | 1040                           |
| ATuF                  | 90                  | 637                            |
| ALOE                  | 76                  | 644                            |
| ALOE Collaborators    | 56                  | 432                            |
| MAPS MOSSAIC          | 34                  | 1275                           |

Table 6.1: Total numbers of exercises and exercise nodes

| Project           | Tutorial Strategy   | Presentation Strategy/<br>Formula Editor |
|-------------------|---|--|
| LeActiveMath      | Default Strategy, Assesment Strategy  | Default, WIRIS formula editor            |
| AMOR              | Default Strategy  | Custom, WIRIS formula editor             |
| Matheführerschein | Matheführerschein Strategy  | Custom, linear formula input             |
| ATuF              | Pre/Post Test Strategy, Treatment Strategy, Familiarization Strategy                        | Custom, MathDox formula editor           |
| ALOE              | Pre/Post Test Strategy, Treatment Strategy, Familiarization Strategy, Default ALOE Strategy | Custom, MathDox formula editor           |
| MAPS MOSSAIC      | Control Strategy, Treatment Strategies 1-7  | Custom, linear formula input             |

Table 6.2: Tutorial and presentation strategies implemented in ACTIVE-MATH

Using the **Exercise Subsystem** of ACTIVEMATH for the projects, mentioned above, we have reached the following objectives:

**Usage for learning.** The exercise representation format, the diagnostic capabilities of the system and the tutorial and presentation strategies suited the needs of teachers at schools and universities. The course contents within the ACTIVEMATH system including hundreds of interactive exercises are used for learning in various schools and Universities in several countries.

**Value as research tool.** The rich expressive power of the exercise language and easily extensible framework for tutorial and presentation strategies allowed to design and perform various research and empiric studies in the field of computer-based education.

## 6.2 Performance tests

In ACTIVEMATH we have performed automated tests measuring the performance of the exercises, as part of general performance testing for the ACTIVEMATH system. These tests have been developed using the JMeter framework.<sup>1</sup>

A part of this automated performance tests were tests with artificial users. "Stress-tests" have been performed with up to 200 artificial users using the system simultaneously. Note that the typical learning time of the exercises was not taken into account, since not every exercise had such metadata. So, a fix delay of 3 seconds was used between the actions of artificial users. This, however, is in most of the cases too little time for performing an exercise step. This means that in the case of real human users the delay between exercise steps will be longer, so the system will have to answer less requests per time unit, which would increase it's efficiency.

The results of the stress tests are shown in Figures 6.1, 6.2 and 6.3. We present three views of these result, that are most commonly used in performance testing, showing average, median and 90% line values respectively.

The average view shows us the sum of all time measurements divided by the number of observations. The median (or 50% line) is a number which divides the samples into two equal halves - the first half is smaller then the

---

<sup>1</sup>See <http://jakarta.apache.org/jmeter/>

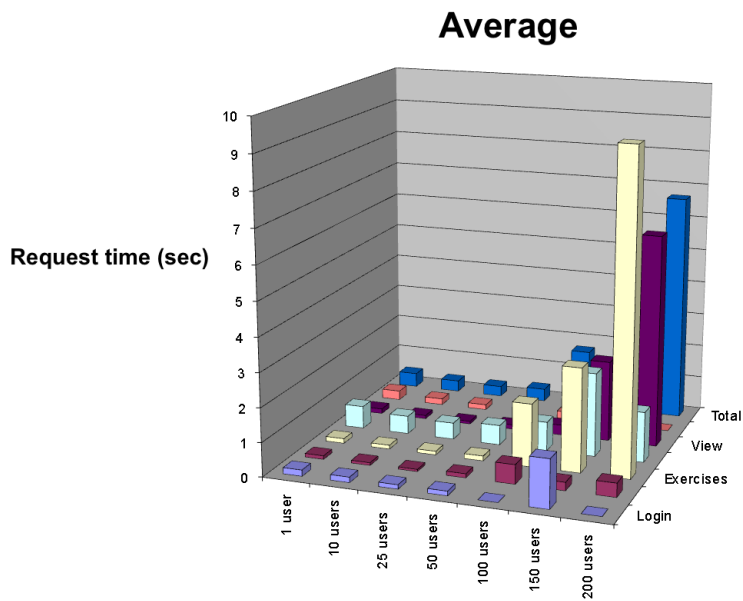
median and the other half is greater. The median value of the (multi)set of numbers can be obtained by ordering all the values and taking the one positioned in the middle of the list. The median might be a better indication of central tendency than the arithmetic mean, and the measure it gives is more robust than the mean in the presence of outlier values (i.e., values that are numerically distant from the rest of the data). The 90% line is the value below which 90% of the samples fall. This value is robust in the presence of only a few outlier values.

From the tests we can see that the exercise system scales well for up to 100 simultaneous users, which corresponds to serving up to three school classes simultaneously using the same instance of the `ACTIVEMATH` server. In this case the average request time for exercises is below 2 seconds and the median and 90% line are below 1 second. For 150 users the average value is about 3 seconds, which is still tolerable, the median value is below 1 second, and the 90% line value is pretty high (9 sec). For 200 students the average value reaches 9.3 seconds, which is practically unusable.

Therefore, currently the `ACTIVEMATH` exercises scale well for 100 simultaneous users per instance of the `ACTIVEMATH` system, which is feasible for practical usage in the classroom.

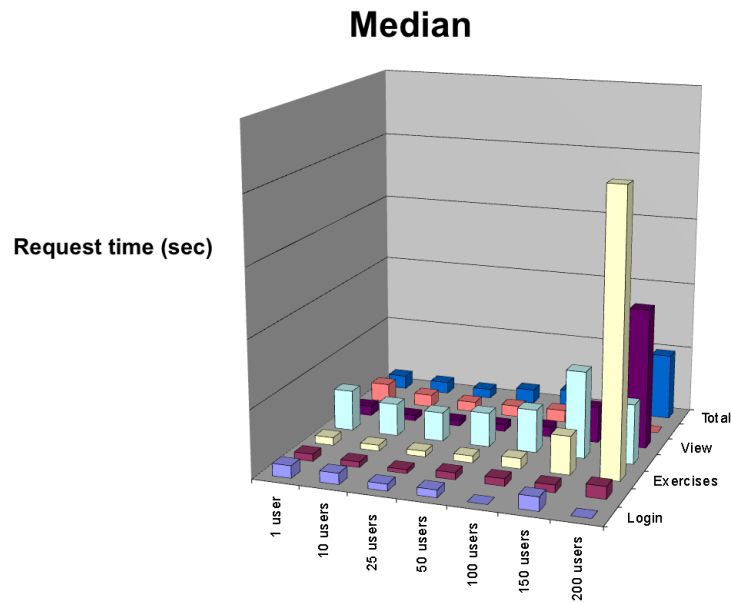
One of the main reasons for using the same instance of the system for the whole class is to ease the collection of the performance data of students in order to generate group performance reports. Of course, there are other practical reasons, such as the need for a separate server to host each running instance of the `ACTIVEMATH` system.





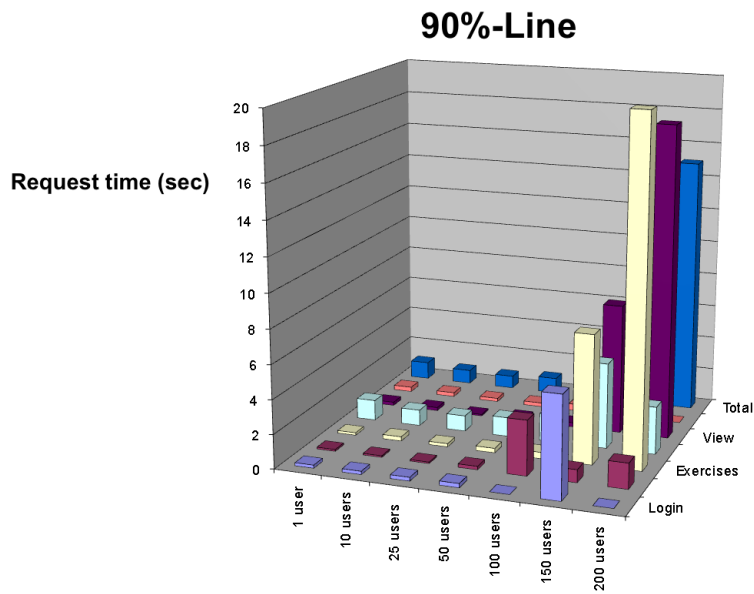
|           | 1 user | 10 users | 25 users | 50 users | 100 users | 150 users | 200 users |
|-----------|--------|----------|----------|----------|-----------|-----------|-----------|
| Login     | 0.183  | 0.161    | 0.128    | 0.135    | 0         | 1.414     | 0         |
| Menu      | 0.107  | 0.081    | 0.066    | 0.121    | 0.568     | 0.246     | 0.41      |
| Exercises | 0.141  | 0.116    | 0.109    | 0.146    | 1.846     | 3.043     | 9.3       |
| Toc       | 0.668  | 0.532    | 0.489    | 0.584    | 0.852     | 2.457     | 1.477     |
| View      | 0.143  | 0.1      | 0.084    | 0.131    | 0.3       | 2.375     | 6.206     |
| Logout    | 0.277  | 0.183    | 0.155    | 0.172    | 0.22      | 0.237     | 0         |
| Total     | 0.423  | 0.335    | 0.306    | 0.375    | 1.715     | 2.46      | 6.693     |

Figure 6.1: Average values of the stress test results



|           | 1 user | 10 users | 25 users | 50 users | 100 users | 150 users | 200 users |
|-----------|--------|----------|----------|----------|-----------|-----------|-----------|
| Login     | 0.183  | 0.165    | 0.102    | 0.113    | 0         | 0.204     | 0         |
| Menu      | 0.107  | 0.081    | 0.057    | 0.104    | 0.11      | 0.112     | 0.185     |
| Exercises | 0.116  | 0.088    | 0.085    | 0.103    | 0.155     | 0.567     | 4.144     |
| Toc       | 0.603  | 0.481    | 0.431    | 0.507    | 0.637     | 1.281     | 0.879     |
| View      | 0.143  | 0.083    | 0.079    | 0.102    | 0.15      | 0.53      | 2.065     |
| Logout    | 0.277  | 0.183    | 0.148    | 0.164    | 0.179     | 0.207     | 0         |
| Total     | 0.203  | 0.174    | 0.138    | 0.208    | 0.259     | 0.949     | 0.979     |

Figure 6.2: Median values of the stress test results



|           | 1 user | 10 users | 25 users | 50 users | 100 users | 150 users | 200 users |
|-----------|--------|----------|----------|----------|-----------|-----------|-----------|
| Login     | 0.183  | 0.197    | 0.222    | 0.244    | 0         | 5.919     | 0         |
| Menu      | 0.107  | 0.095    | 0.086    | 0.205    | 3.221     | 0.752     | 1.495     |
| Exercises | 0.15   | 0.254    | 0.191    | 0.265    | 0.374     | 7.524     | 19.97     |
| Toc       | 1.21   | 0.942    | 0.927    | 1.124    | 1.623     | 5.007     | 2.787     |
| View      | 0.203  | 0.19     | 0.092    | 0.229    | 0.251     | 7.614     | 18.255    |
| Logout    | 0.277  | 0.24     | 0.2      | 0.231    | 0.394     | 0.37      | 0         |
| Total     | 1.033  | 0.826    | 0.758    | 0.921    | 1.444     | 6.653     | 15.018    |

Figure 6.3: 90<sup>th</sup> percentile values of the stress test results



# Chapter 7

## Conclusion and Future Work

In this work we have defined a knowledge representation for interactive exercises and developed the **Exercise Subsystem** within the **ACTIVEMATH** learning platform. This system has several novel aspects as compared to other existing systems that serve interactive exercises. Firstly, thanks to our framework of distributed semantic services and generic query format, we can connect to several external systems simultaneously in order to obtain diagnosis of student's answers. A quick diagnosis can be computed by a CAS, a more detailed diagnosis can be provided by a domain reasoner.

Secondly, we support a combination of manually authored and automatically generated exercises. Moreover, the exercises can be reused with different automated tutorial strategies.

A variety of presentation strategies allows for custom layout of feedback and via combining various user interface components we can implement different approaches to user interaction.

There are several directions of further research we plan to conduct, which we briefly describe in the following sections.

### 7.1 Authoring Tutorial Strategies

Currently, the tutorial strategies for exercises are realized as programs. We have also developed a framework in which these strategies can be combined with each other to form more complex strategies.

One of the directions of further research is to devise a knowledge representation for tutorial strategies and implement a module of the **Exercise**

**Subsystem** that interprets such a representation and applies the corresponding tutorial behavior to the given exercise. Further step would be to develop an authoring tool for such strategies. As opposed to the low level procedure representation such as the so-called flow-lines in the EON strategy editor [Murray, 2003a], we aim at more high-level declarative representation that will allow the teachers to express their wishes for the tutorial behavior of the strategy without programming.

## 7.2 More Domain Reasoners

In *ACTIVEMATH*, we have defined guidelines for implementing domain reasoners suitable for interactive exercises, as described in Section 4.3.2. Two domain reasoners have been already implemented following these guidelines. These reasoners are implemented as modules for the *YACAS* system, in which the mathematical contexts are defined by sets of rewrite rules. The next step is to develop a graphical user interface, allowing authors to create new mathematical contexts by specifying sets of rewrite rules for the given mathematical domain. These new contexts can be saved as executable modules of the *YACAS* system. This procedure would facilitate "authoring" new domain reasoners without programming.

Another research question could be comparing the educational value of different domain reasoners for the same domain. In order to perform such a comparison, we would connect several domain reasoners for the same domain to the *ACTIVEMATH* system and compare the learning gains of the interactive exercises generated using these reasoners.

## 7.3 Authoring Advanced User Interfaces

In order to facilitate the learner interaction we need to introduce more structure into the user interface for the exercise steps. However, more structure does not necessarily mean more buttons and palettes, but finding the right balance between prescribing the interaction format and giving enough freedom to the student for entering the solution. Currently, several formula editors are used in *ACTIVEMATH* and some extensions of the user interface were made in projects *ATuF* and *ALOE* in order to extend the learner's working area by providing the student with structured input templates. Cur-

rently, the set of these templates is hardcoded for each domain. Important further work would be to provide a general tool for authoring such templates for various contexts, or generating them using domain reasoners.

## 7.4 Exercise Log Data Analysis

The knowledge representation of exercises allows for further analysis of the log data after the learner's interaction.

For example, typical errors can be machine-learned from such log data, in which the incorrect answers of different users are compared to each other using semantic comparisons in order to identify the classes of most common errors. Such work has been tried within a Bachelor thesis project at the Saarland University [Bellem, 2008].

An interesting further development would be an automated procedure that collects the most frequent student errors, generates rewrite rules that match these errors and stores them within the domain reasoner module assigned to the given context.





## References to Own Work

In the following we list publications, containing parts of this work that have been published at national and international conferences and workshops, leading journals in the fields of Educational Technology, and the scientific reports on research conducted within several national and European research projects.

[Tsovaltzi et al., 2009], [Tsovaltzi et al., 2010a], [Tsovaltzi et al., 2010b],  
[Gogvadze, 2009b], [Gogvadze, 2009a], [Gogvadze and Melis, 2009],  
[Gogvadze and Melis, 2008a], [Gogvadze and Melis, 2008b],  
[Gogvadze and Tsigler, 2007], [Melis et al., 2007], [Melis and Gogvadze, 2006],  
[Melis et al., 2006], [Gogvadze et al., 2006], [Gogvadze, 2005],  
[Gogvadze et al., 2005b], [Gogvadze et al., 2005a], [Gogvadze et al., 2004b],  
[Gogvadze et al., 2004a], [Melis and Gogvadze, 2004a],  
[Gogvadze and Palomo, 2003], [Gogvadze et al., 2003b], [Melis et al., 2003c],  
[Melis et al., 2003a], [Melis et al., 2003b], [Gogvadze et al., 2003a],  
[Büdenbender et al., 2002b], [Gogvadze, 2002b], [Gogvadze, 2002a],  
[Melis et al., 2002], [Gogvadze et al., 2001], [Melis et al., 2001a],  
[Melis et al., 2001b], [Siekmann et al., 2000]



# Bibliography

- [Ainsworth et al., 2003] Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., J. Underwood, B. W., and Wood, D. (2003). Redeem: Simple intelligent tutoring systems from usable tools. In Murray, T., Blessing, S., and Ainsworth, S., editors, *Authoring Tools for Advanced Technology Learning Environments*, chapter 8, pages 205–232. Kluwer Academic Publishers. Printed in the Netherlands.
- [Aleven et al., 2009] Aleven, V., McLaren, B., Sewall, J., and Koedinger, K. (2009). Example-tracing tutors: A new paradigm for intelligent tutoring systems. *International Journal of Artificial Intelligence in Education (IJAIED)*. *Special Issue on "Authoring Systems for Intelligent Tutoring Systems"*, 19:105–154.
- [Aleven et al., 2006] Aleven, V., McLaren, B. M., Sewall, J., and Koedinger, K. R. (2006). The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In [Ikeda et al., 2006], pages 61–70.
- [Anderson, 1993] Anderson, J. (1993). *Rules of the Mind*. Lawrence Erlbaum Associates, Inc. ISBN: 0-8058-1199-0.
- [Anderson et al., 1985] Anderson, J., Boyle, C., and Yost, G. (1985). The geometry tutor. In *Proceedings of the 9th international joint conference on Artificial Intelligence*, pages 1–7, Los Angeles, California. Springer-Verlag.
- [Anderson et al., 1995] Anderson, J., Corbett, A., Koedinger, K., and R. Pelletier (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207.
- [Anderson and Reiser, 1985] Anderson, J. and Reiser, B. (1985). The lisp tutor. *Byte*, 10:159–175.

- [Anderson et al., 2001] Anderson, L., Krathwohl, D., Airasian, P., Cruikshank, K., Mayer, R., Pintrich, P., and Wittrock, M. (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York.
- [Asperti et al., 2003] Asperti, A., Buchberger, B., and Davenport, J. H., editors (2003). *Mathematical Knowledge Management, Second International Conference, MKM 2003, Bertinoro, Italy, February 16-18, 2003, Proceedings*, volume 2594 of *Lecture Notes in Computer Science*. Springer.
- [Baghaei et al., 2005] Baghaei, N., Mitrovic, A., and Irwin, W. (2005). A constraint-based tutor for learning object-oriented analysis and design using uml. In *Proceeding of the 2005 conference on Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation*, volume 133 of *Frontiers in Artificial Intelligence and Applications*, pages 11–18.
- [Beeson, 1996] Beeson, M. (1996). Design principles of mathpert: Software to support education in algebra and calculus. In N.Kajler, editor, *Human Interfaces to Symbolic Computation*, volume 19, pages 105–154. Springer-Verlag.
- [Bellem, 2008] Bellem, M. (2008). Creation and analysis of a problem space based on no-feedback exercises. Bachelor thesis, Saarland University.
- [Bloom, 1956] Bloom, B., editor (1956). *Taxonomy of Educational Objectives, Handbook I: Cognitive Domain*. David McKay, New York.
- [Bloom, 1984] Bloom, B. (1984). The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16.
- [B.P.Woolf, 2008] B.P.Woolf (2008). *Building Intelligent Interactive Tutors, Student-Centered Strategies for Revolutionizing E-Learning*. Elsevier & Morgan Kaufmann.
- [Brusilovsky et al., 2003] Brusilovsky, P., Corbett, A. T., and de Rosis, F., editors (2003). *User Modeling 2003, 9th International Conference, UM 2003, Johnstown, PA, USA, June 22-26, 2003, Proceedings*, volume 2702 of *Lecture Notes in Computer Science*. Springer.

- [Büdenbender et al., 2002a] Büdenbender, J., Frischauf, A., Goguadze, G., Melis, E., Libbrecht, P., and Ullrich, C. (2002a). Using computer algebra systems as cognitive tools. In [Cerri et al., 2002], pages 802–810.
- [Büdenbender et al., 2002b] Büdenbender, J., Frischauf, A., Goguadze, G., Melis, E., Libbrecht, P., and Ullrich, C. (2002b). Using computer algebra systems as cognitive tools. In [Cerri et al., 2002], pages 802–810.
- [Burton, 1982] Burton, R. (1982). Diagnosing bugs in a simple procedural skill. In Brown, I. D. S. . L., editor, *Intelligent Tutoring Systems*. Academic Press, London.
- [Burton and Brown, 1978] Burton, R. and Brown, J. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2:155–191.
- [Buswell et al., 2004] Buswell, S., Caprotti, O., Carlisle, D. P., Dewar, M. C., Gaëtano, M., and Kohlhase, M., editors (2004). *The OpenMath Standard version 2.0*. The OpenMath Society.
- [Carbonell, 1970] Carbonell, J. (1970). Al in cai: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11:190–202.
- [Carette et al., 2009] Carette, J., Dixon, L., Coen, C. S., and Watt, S. M., editors (2009). *Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009, Held as Part of CICM 2009, Grand Bend, Canada, July 6-12, 2009. Proceedings*, volume 5625 of *LNCS*. Springer.
- [Cerri et al., 2002] Cerri, S. A., Gouardères, G., and Paraguaçu, F., editors (2002). *Intelligent Tutoring Systems, 6th International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain, June 2-7, 2002, Proceedings*, volume 2363 of *Lecture Notes in Computer Science*. Springer.
- [Cohen et al., 2005a] Cohen, A., Cuypers, H., Jibetean, D., Spanbroek, M., Caprotti, O., and D. Marques, R. E., and Pau, A. (2005a). LEACTIVE-MATH integrated cas with OPENMATH, LEACTIVE-MATH deliverable d12. Technical report, LEACTIVE-MATH Consortium.

- [Cohen et al., 2005b] Cohen, A., Cuypers, H., Jibeteau, D., Spanbroek, M., Caprotti, O., Eixarch, R., Marques, D., , and Pau, A. (2005b). LEACTIVE-MATH phrasebooks. LEACTIVEMATH Deliverable D14, LEACTIVEMATH Consortium.
- [Collins et al., 1989] Collins, A., Brown, J., and Newman, S. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In Resnick, L., editor, *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, pages 453–494. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.
- [Conati and VanLehn, 1999] Conati, C. and VanLehn, K. (1999). Teaching meta-cognitive skills: implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. In Lajoie, S. P. and Vivet, M., editors, *Artificial Intelligence in Education*, pages 297–304, Amsterdam. IOS Press.
- [Conati and VanLehn, 2000] Conati, C. and VanLehn, K. (2000). Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. *International Journal of Artificial Intelligence in Education*, 11:398–415.
- [Consortium, 2004] Consortium, LEACTIVEMATH. (2004). Requirement analysis, deliverable d5. Technical report, LeActiveMath Consortium.
- [Corbett, 2001] Corbett, A. (2001). Cognitive computer tutors: Solving the two-sigma problem. In Bauer, M., Gmytrasiewicz, P., and Vassileva, J., editors, *UM 2001*, volume 2109, pages 137–147. Springer-Verlag Berlin Heidelberg.
- [Corbett and Anderson, 1995] Corbett, A. T. and Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278. 10.1007/BF01099821.
- [Corbett et al., 1997] Corbett, A. T., Koedinger, K. R., and Anderson, J. R. (1997). Intelligent tutoring systems. In [Helander et al., 1997], chapter 37, pages 849–874.

- [Crowder, 1959] Crowder, N. (1959). Automatic tutoring by means of intrinsic programming. In *Automatic Teaching: The State of the Art*, pages 109–116. Wiley, New York.
- [Davenport, 2008] Davenport, J. H. (2008). Aisc meets natural typography. In et al., S. A., editor, *Proceedings of AISC/Calcuemus/MKM 2008*, number 5144 in LNAI, pages 53–60. Springer-Verlag.
- [Dimitrova et al., 2009] Dimitrova, V., Mizoguchi, R., du Boulay, B., and Graesser, A. C., editors (2009). *Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling, Proceedings of the 14th International Conference on Artificial Intelligence in Education, AIED 2009, July 6-10, 2009, Brighton, UK*, volume 200 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.
- [du Boulay et al., 2008] du Boulay, B., Mendez, G., Luckin, R., Miron, E. M., and Harris, A. (2008). Motivationally intelligent systems: Three questions. In *Second International Conference on Innovations in Learning for the Future, Future e-Learning 2008, Istanbul*. Istanbul University Rectorate Publication No: 4793.
- [Dudev and Palomo, 2007] Dudev, M. and Palomo, A. G. (2007). Generating parametrized exercises. Technical report, Student Project at the University of Saarland, March 2007. [http://www.matracas.org/escritos/edtech\\_report.pdf](http://www.matracas.org/escritos/edtech_report.pdf).
- [Eichelmann et al., 2008] Eichelmann, A., Narciss, S., Faulhaber, A., and Melis, E. (2008). Analyzing computer-based fraction tasks on the basis of a two-dimensional view of mathematics competences. In Zumbach, J., Schwartz, N., Seufert, T., and Lester, L., editors, *Beyond Knowledge: The Legacy of Competence. Meaningfull Coumputer-Based Learning Environments. EARLI SIG 7 Learning and Instruction with Computers in cooperation with SIG 6 Instructional Design (EARLI SIG-6 & 7), September 2-4, 2008, Salzburg, Australia*, pages 125–134. Springer.
- [Faulhaber and Melis, 2008] Faulhaber, A. and Melis, E. (2008). An efficient student model based on student performance and metadata. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., and Avouris, N. M., editors, *Proceedings of 18th European Conference on Artificial Intelligence (ECAI-08)*,

*Patras, Greece, July 21-25, 2008*, volume 178 of *Frontiers in Artificial Intelligence and Applications (FAIA)*, pages 276–280. IOS Press.

- [Gerdes et al., 2008] Gerdes, A., Heeren, B., Jeurings, J., and Stuurman, S. (2008). Feedback services for exercise assistants. In Remenyi, D., editor, *Proceedings of the 7th European Conference on e-Learning (ECEL 2008)*, Lecture Notes in Artificial Intelligence (LNAI), pages 402–410, Cyprus. Academic Publishing Limited Reading.
- [Gertner et al., 1998] Gertner, A., Conati, C., and Lehn, K. V. (1998). Procedural help in andes: Generating hints using a bayesian network student model. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence AAAI-98*, pages 106–111, Cambridge, MA. The MIT Press.
- [Gogvadze, 2002a] Gogvadze, G. (2002a). Metadata for mathematical libraries. Technical report, Report n. D3.a, Project IST-2001-33562 MOWGLI.
- [Gogvadze, 2002b] Gogvadze, G. (2002b). Metadata model. Technical report, Report n. D3.b, Project IST-2001-33562 MOWGLI.
- [Gogvadze, 2005] Gogvadze, G. (2005). Generic cas interaction. LEACTIVE-MATH Deliverable D25, LEACTIVEMATH Consortium.
- [Gogvadze, 2009a] Gogvadze, G. (2009a). Representation for interactive exercises. In Carette, J., Dixon, L., Coen, C. S., and Watt, S. M., editors, *Intelligent Computer Mathematics, 16th Symposium Calculemus 2009, 8th International Conference MKM 2009, Grand Bend, Canada, July 6-12, 2009, Proceedings*, volume 5625 of *LNCS*, pages 294–309. Springer.
- [Gogvadze, 2009b] Gogvadze, G. (2009b). Semantic evaluation services for web-based exercises. In Spaniol, M., Li, Q., Klamma, R., and Lau, R. W. H., editors, *ICWL*, volume 5686 of *LNCS*, pages 172–181. Springer.
- [Gogvadze et al., 2005a] Gogvadze, G., González Palomo, A., and Melis, E. (2005a). Interactivity of exercises in activemath. *Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences Sharing. Good Practices of Research, Experimentation and Innovation, Frontiers in Artificial Intelligence and Applications*.



- [Goguadze et al., 2005b] Goguadze, G., González Palomo, A., and Melis, E. (2005b). Interactivity of exercises in activemath. In *Proceedings of International Conference on Computers in Education (ICCE05), December 2005, Singapore*. also published as Journal Article (see below).
- [Goguadze et al., 2004a] Goguadze, G., González Palomo, A., Tsigler, I., and Winterstein, S. (2004a). LEACTIVE MATH presentation architecture, LEACTIVE MATH deliverable d9. Technical report, LEACTIVE MATH Consortium.
- [Goguadze et al., 2006] Goguadze, G., Mavrikis, M., and Palomo, A. (2006). Interoperability issues between markup formats for mathematical exercise. In Seppälä, M., Xambo, S., and Caprotti, O., editors, *Proceedings of WebAlt 2006, First WebALT Conference and Exhibition, January 5-6, 2006, Technical University of Eindhoven, Netherlands*, pages 69–80. <http://webalt.math.helsinki.fi/webalt2006/content/e31/e176/webalt2006.pdf>.
- [Goguadze and Melis, 2008a] Goguadze, G. and Melis, E. (2008a). Feedback in ActiveMath exercises. In *11th International Conference on Mathematical Education ICME 11, July 6-13, Monterrey, Mexico*. <http://tsg.icme11.org/document/get/232>.
- [Goguadze and Melis, 2008b] Goguadze, G. and Melis, E. (2008b). One exercise - various tutorial strategies. In Woolf, B. P., Aïmeur, E., Nkambou, R., and Lajoie, S. P., editors, *Intelligent Tutoring Systems, 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008, Proceedings*, volume 5091 of *LNCS*, pages 755–757. Springer.
- [Goguadze and Melis, 2009] Goguadze, G. and Melis, E. (2009). Combining evaluative and generative diagnosis in activemath. In Dimitrova, V., Mizoguchi, R., du Boulay, B., and Graesser, A. C., editors, *Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling, Proceedings of the 14th International Conference on Artificial Intelligence in Education, AIED 2009, July*, volume 200 of *Frontiers in Artificial Intelligence and Applications*, pages 668–670. IOS Press.

- [Gogvadze et al., 2001] Gogvadze, G., Melis, E., and Asperti, A. (2001). Structure and meta-structure of mathematical documents. Technical report, Report n. D1.b, Project IST-2001-33562 MOWGLI.
- [Gogvadze et al., 2003a] Gogvadze, G., Melis, E., Izhutkin, V., and Isulanov, Y. (2003a). Interactively learning operations research methods with activemath. In H.G.Bock, Domschke, W., Fahrion, R., Juenger, M., Kogelschatz, H., Liesegang, G., Reinelt, G., Rendl, F., and Waescher, G., editors, *Operations Research 2003, Annual Conference of the German Operations Research Society. Heidelberg*, page 159, Heidelberg.
- [Gogvadze et al., 2003b] Gogvadze, G., Melis, E., Ullrich, C., and Cairns, P. A. (2003b). Problems and solutions for markup for mathematical examples and exercises. In Asperti, A., Buchberger, B., and Davenport, J. H., editors, *Mathematical Knowledge Management, Second International Conference, MKM 2003, Bertinoro, Italy, February 16-18, 2003, Proceedings*, volume 2594 of *Lecture Notes in Computer Science*, pages 80–92. Springer.
- [Gogvadze and Palomo, 2003] Gogvadze, G. and Palomo, A. G. (2003). Adapting mainstream editors for semantic authoring of mathematics. In *Mathematical Knowledge Management Symposium, 25-29 November 2003, Heriot-Watt University Edinburgh, Scotland*.
- [Gogvadze and Tsigler, 2007] Gogvadze, G. and Tsigler, I. (2007). Authoring interactive exercises in ActiveMath. In *Proceedings of the MathUI Workshop at the Sixth Mathematical Knowledge Management Conference, Schloss Hagenberg, Linz, June 27, 2007*. <http://www.activemath.org/workshops/MathUI/07/proceedings/Gogvadze-Tsigler-ExAMAT-MathUI07.pdf>.
- [Gogvadze et al., 2004b] Gogvadze, G., Ullrich, C., Melis, E., Siekmann, J., Groß, C., and Morales, R. (2004b). LeActiveMath structure and metadata model, deliverable d6. Technical report, LeActiveMath Consortium, Saarland University.
- [Grabowski et al., 2005] Grabowski, B., Gäng, S., Herter, J., and Köppen, T. (2005). Mathcoach and laplasescript: Advanced exercise programming for mathematics with dynamic help generation. In *Proceedings of the ICL2005 Workshop, at International Conference on Interactive Computer Aided Learning ICL, Villach, Austria*.

- [Heffernan, 2001] Heffernan, N. (2001). *Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of Human Tutors*. Doctoral dissertation, School Of Computer Science Carnegie Mellon University, CMU-CS-01-127.
- [Heffernan et al., 2008] Heffernan, N., Koedinger, K. R., and Razzaq, L. (2008). Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. *The International Journal of Artificial Intelligence in Education*, 18(2):153–178.
- [Helander et al., 1997] Helander, M., Landauer, T. K., and Prabhu, P. V., editors (1997). *Handbook of Human-Computer Interaction*. Elsevier Science B.V., Amsterdam, second, completely revised edition edition.
- [Henneke, 1999] Henneke, M. (1999). *Online Diagnose in intelligenten mathematischen Lehr-Lern-Systemen*. PhD thesis, Universität Hildesheim.
- [Ikeda et al., 2006] Ikeda, M., Ashley, K. D., and Chan, T.-W., editors (2006). *Intelligent Tutoring Systems, 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006, Proceedings*, volume 4053 of *Lecture Notes in Computer Science*. Springer.
- [IMS Global Learning Consortium, 2002] IMS Global Learning Consortium (2002). IEEE learning object metadata standard. <http://www.imsglobal.org/lom>.
- [IMS Global Learning Consortium, 2005] IMS Global Learning Consortium (2005). IMS question and test interoperability overview version, 2.0 final specification, 24 january 2005. [http://www.imsglobal.org/question/qti\\_v2p0/imsqti\\_oviewv2p0.html](http://www.imsglobal.org/question/qti_v2p0/imsqti_oviewv2p0.html).
- [Izhutkin et al., 2004] Izhutkin, V., Melis, E., Toktarova, V., and Goguadze, G. (2004). Interactive education methods for the solution of extremal tasks with the help of the learning system activemath. In *Telematika 2004*, Moskow.
- [Kinshuk et al., 2005] Kinshuk, Sampson, D. G., and Isaías, P. T., editors (2005). *Cognition and Exploratory Learning in Digital Age, CELDA 2005, 14-16 December 2005, Porto, Portugal, Proceedings*. IADIS.

- [Klieme et al., 2004] Klieme, E., Avenarius, H., Blum, W., Döbrich, P., Gruber, H., Prenzel, M., Reiss, K., Riquarts, K., Rost, J., Tenorth, H., , and Vollmer, H. J. (2004). The development of national educational standards - an expertise. Technical report, Bundesministerium für Bildung und Forschung / German Federal Ministry of Education and Research.
- [Kodaganallur et al., 2005] Kodaganallur, V., R.Weitz, and Rosenthal, D. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *IJAIED*, 15(2):117–144.
- [Kodaganallur et al., 2006] Kodaganallur, V., R.Weitz, and Rosenthal, D. (2006). An assessment of constraint-based tutors: A response to mitrovic and ohlsson’s critique of ”a comparison of model-tracing and constraint-based intelligent tutoring paradigms”. *IJAIED*, 16:291–321.
- [Koedinger et al., 2004] Koedinger, K., Alevan, V., Heffernan, N., McLaren, B., and Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behaviour by demonstration. In [Lester et al., 2004], pages 162–174.
- [Koedinger and Anderson, 1997] Koedinger, K. and Anderson, J. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43.
- [Kohlhase, 2000] Kohlhase, M. (2000). OMDoc: Towards an OPENMATH representation of mathematical documents. Seki Report, FR Informatik, Universität des Saarlandes.
- [Kohlhase, 2005] Kohlhase, M. (2005). OMDoc: An open markup format for mathematical documents (version 1.1). Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, Pa 15213, USA.
- [Kohlhase, 2006] Kohlhase, M. (2006). OMDoc: *An Open Markup Format for Mathematical Documents (Version 1.2), (the technical specification, Primer, and discussion of the upcoming version 1.2 of OMDoc)*. Number 4180 in Lecture Notes in Artificial Intelligence (LNAI). Springer Verlag. Errata, LNAI 4810 with Corrections.
- [LeActiveMath, 2004] LeActiveMath (2004). Leactivemath project, funded under the 6th framework programme of the european commission, 2004-2006. <http://www.leactivemath.org>.

- [Lehn et al., 2002] Lehn, K. V., Lynch, C., Taylor, L., Winstein, A., Shelby, R., Schulze, K., Treacy, D., and M. Wintersgill (2002). Minimally invasive tutoring of complex physics problem solving. In S.A. Cerri, G. G. and Paraguaçu, F., editors, *Proceedings of ITS 2002*, number 2363 in LNCS, pages 367–376. Springer-Verlag Berlin Heidelberg.
- [Lester et al., 2004] Lester, J. C., Vicari, R. M., and Paraguaçu, F., editors (2004). *Intelligent Tutoring Systems, 7th International Conference, ITS 2004, Maceió, Alagoas, Brazil, August 30 - September 3, 2004, Proceedings*, volume 3220 of *Lecture Notes in Computer Science*. Springer.
- [Libbrecht et al., 2001] Libbrecht, P., Frischauf, A., Melis, E., Pollet, M., and Ullrich, C. (2001). Integration of mathematical systems into the activemath learning environment. In Wang, P., Kajler, N., and Diaz, A., editors, *ISSAC-2001 Workshop on Internet Accessible Mathematical Computation*.
- [Maple, 2009] Maple (2009). Maple<sup>TM</sup>, ©Maplesoft, a division of Waterloo Maple Inc. 2009. <http://www.maplesoft.com>.
- [MapleTA<sup>TM</sup>, 2010] MapleTA<sup>TM</sup> (2010). MapleTA<sup>TM</sup> testing, evaluation and grading software ©maplesoft. <http://www.maplesoft.com/products/mapleta>.
- [Masood, 2009] Masood, S. (2009). A domain reasoner for fraction arithmetics, serving intelligent exercises. Master’s thesis, Saarland University.
- [MathDox, 2007] MathDox (2007). Mathdox tool for interactive mathematical documents. <http://www.mathdox.org>.
- [Matsuda et al., 2005] Matsuda, N., Cohen, W. W., and Koedinger, K. R. (2005). Applying programming by demonstration in an intelligent authoring tool for cognitive tutors. In *AAAI Workshop on Human Comprehensible Machine Learning. Technical Report WS-05-04*, pages 1–8, Menlo Park, CA. AAAI association.
- [Mavrikis and Maciocia, 2003] Mavrikis, M. and Maciocia, A. (2003). Wallis: a web-based ile for science and engineering students studying mathematics. In *Workshop of Advanced Technologies for Mathematics Education in 11th International Conference on Artificial Intelligence in Education, Sydney Australia*.

- [Mavrikis and Palomo, 2004] Mavrikis, M. and Palomo, A. G. (2004). Mathematical, interactive exercise generation from static documents. *Electronic Notes on Theoretical Computer Science*, 93:183–201.
- [Maxima, 2010] Maxima (2000-2010). Maxima, a computer algebra system. <http://maxima.sourceforge.net/>.
- [Mayo et al., 2000] Mayo, M., Mitrovic, A., and McKenzie, J. (2000). Capit: an intelligent tutoring system for capitalization and punctuation. In *International Workshop for Advanced Learning Technologies IWALT2000, December 4-6, Palmerston North*, pages 151–154.
- [Melis, 2005] Melis, E. (2005). Choice of feedback strategies. In [Kinshuk et al., 2005], pages 183–189.
- [Melis et al., 2001a] Melis, E., Andrès, E., Büdenbender, J., Frischauf, A., Gogvadze, G., Libbrecht, P., Pollet, M., and Ullrich, C. (2001a). Activemath: A generic and adaptive web-based learning environment. *Journal of Artificial Intelligence in Education*, 12(4):385–407.
- [Melis et al., 2001b] Melis, E., Andres, E., Gogvadze, G., Libbrecht, P., Pollet, M., and Ullrich, C. (2001b). Activemath system description. In Moore, J., Redfield, C., and Johnson, W., editors, *Proceedings of AIED 2001, 10th World Conference of Artificial Intelligence and Education, May 19-23, 2001, S. Antonio, Texas*. IOS Press.
- [Melis et al., 2002] Melis, E., Büdenbender, J., Gogvadze, G., Libbrecht, P., and Ullrich, C. (2002). Semantics for web-based mathematical education systems. In *Semantic Web Workshop at WWW 2002, Hawaii, USA*.
- [Melis et al., 2003a] Melis, E., Büdenbender, J., Gogvadze, G., Libbrecht, P., and Ullrich, C. (2003a). Knowledge representation and management in activemath. *Ann. Math. Artif. Intell.*, 38(1-3):47–64.
- [Melis et al., 2008] Melis, E., Faulhaber, A., Eichelmann, A., and Narciss, S. (2008). Interoperable competencies characterizing learning objects in mathematics. In Woolf, B. P., Aïmeur, E., Nkambou, R., and Lajoie, S. P., editors, *Intelligent Tutoring Systems (ITS 2008), 9th International Conference, Canada, June 23-27, 2008, Proceedings*, volume 5091 of *LNCS*, pages 416–425. Springer.

- [Melis and Gogvadze, 2004a] Melis, E. and Gogvadze, G. (2004a). Towards adaptive generation of faded examples. In Lester, J. C., Vicari, R. M., and Paraguaçu, F., editors, *Intelligent Tutoring Systems, 7th International Conference, ITS 2004, Maceiò, Alagoas, Brazil, August 30 - September 3, 2004, Proceedings*, volume 3220 of *Lecture Notes in Computer Science*, pages 762–771. Springer.
- [Melis and Gogvadze, 2004b] Melis, E. and Gogvadze, G. (2004b). Towards adaptive generation of faded examples. In [Lester et al., 2004], pages 762–771.
- [Melis and Gogvadze, 2006] Melis, E. and Gogvadze, G. (2006). Representation of misconceptions. In Spector, J., Isaias Kinshuk, P., and Sampson, D., editors, *Cognition and Learning in the Digital Age (CELDA-2006)*, pages 208–214. IADIS.
- [Melis et al., 2006] Melis, E., Gogvadze, G., Homik, M., Libbrecht, P., Ullrich, C., and Winterstein, S. (2006). Semantic-aware components and services of activemath. *British Journal of Educational Technology*, 37(3).
- [Melis et al., 2003b] Melis, E., Gogvadze, G., Libbrecht, P., and Ullrich, C. (2003b). Wissensmodellierung und -nutzung in activemath. *KI*, 17(1):12–.
- [Melis et al., 2003c] Melis, E., Gogvadze, G., P. Libbrecht, I. N., Ullrich, C., and Winterstein, S. (2003c). Education-relevant features in actinmath. In *ONLINE EDUCA BERLIN 2003 - 9th International Conference on Technology Supported Learning and Training. December 3 - 5, 2003, Berlin*.
- [Melis et al., 2007] Melis, E., Moormann, M., Ullrich, C., Gogvadze, G., and Libbrecht, P. (2007). How ActiveMath supports moderate constructivist mathematics teaching. In Milková, E. and Pražák, P., editors, *Proceedings of the 8th International Conference on Technology in Mathematics Teaching (ICTMT 2007), Hradec Králové, Czech Republic, 1-4 July 2007*. <http://www-ags.dfki.uni-sb.de/~melis/Pub/MelisSupportingModerateConstructivism-ICTMT8.pdf>.
- [Méndez et al., 2003] Méndez, G., Rickel, J., and de Antonio, A. (2003). Steve meets jack: the integration of an intelligent tutor and a virtual environment with planning capabilities. In *4th International Working Conference on Intelligent Virtual Agents (IVA03). Irsee, Germany. 15-17 Sept.*,

2003. *Intelligent Virtual Agents*, number 2792 in LNAI, pages 325–332. Springer-Verlag.
- [Millán et al., 2010] Millán, E., T.Loboda, and de-la Cruz, J. L. P. (2010). Bayesian networks for student model engineering. *Computers & Education*, 55(4):1663–1683.
- [Mitrovic, 2002] Mitrovic, A. (2002). Normit, a web-enabled tutor for database normalization. In Kinshuk, R.Lewis, Akahori, K., Kemp, R., T.Okamoto, Henderson, L., and Lee, C.-H., editors, *Proc. ICCE 2002*, pages 1276–1280, Auckland.
- [Mitrovic et al., 2003a] Mitrovic, A., Koedinger, K., and Martin, B. (2003a). A comparative analysis of cognitive tutoring and constraint-based modeling. In et al., P. B., editor, *Proceedings of the 9th International Conference on User Modeling (UM2003)*, number 2702 in LNAI, pages 313–322, Berlin Heidelberg New York. Springer-Verlag.
- [Mitrovic et al., 2003b] Mitrovic, A., Koedinger, K. R., and Martin, B. (2003b). A comparative analysis of cognitive tutoring and constraint-based modeling. In [Brusilovsky et al., 2003], pages 313–322.
- [Mitrovic et al., 2008] Mitrovic, A., McGuigan, N., Martin, B., Suraweera, P., Milik, N., and Holland, J. (2008). Authoring constraint-based tutors in aspire: a case study of a capital investment tutor. In *ED-MEDIA 2008, Vienna, 30.6.-4.7.2008*, pages 4607–4616.
- [Mitrovic and Ohlsson, 1999] Mitrovic, A. and Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10:238–256.
- [Mitrovic and Ohlsson, 2006] Mitrovic, A. and Ohlsson, S. (2006). A critique of kodaganallur, weitz and rosenthal, "a comparison of model-tracing and constraint-based intelligent tutoring paradigms". *International Journal of Artificial Intelligence in Education*, 16(3):277–289.
- [MuPAD, 2009] MuPAD (1994-2009). MuPAD computer algebra system ©1994-2009 The MathWorks, Inc. <http://www.mupad.de>.
- [Murray, 1991] Murray, T. (1991). Facilitating teacher participation in intelligent computer tutor design: Tools and design methods. Technical report,



- Univ. of Massachusetts. Ed.D. Dissertation, Computer Science Tech. Report 91-95.
- [Murray, 1998] Murray, T. (1998). Authoring knowledge based tutors: Tools for content, instructional strategy, student model, and interface design. *Journal of the Learning Sciences*, 7(1):5–64.
- [Murray, 1999] Murray, T. (1999). Authoring intelligent tutoring systems: Analysis of the state of the art. *Int. J. of AI and Education*, 10(1):98–129.
- [Murray, 2003a] Murray, T. (2003a). Eon: Authoring tools for content, instructional strategy, student model and interface design. In Murray, Ainsworth, and Blessing, editors, *Authoring Tools for Advanced Technology Learning Environments*, pages 309–339. Kluwer Academic Publishers.
- [Murray, 2003b] Murray, T. (2003b). Principles for pedagogy-oriented knowledge based tutor authoring systems: Lessons learned and a design meta-model. In Murray, Ainsworth, and Blessing, editors, *Authoring Tools for Advanced Technology Learning Environments*, pages 439–467. Kluwer Academic Publishers.
- [Murray, 2004] Murray, T. (2004). Design tradeoffs in usability and power for advanced educational software authoring tools. *Educational Technology*, 44(5):10–16.
- [Narciss, 2004] Narciss, S. (2004). The impact of informative tutoring feedback and self-efficacy on motivation and achievement in concept learning. *Experimental Psychology*, 51(3):214–228.
- [Narciss, 2006] Narciss, S. (2006). *Informatives tutorielles Feedback. Entwicklungs- und Evaluationsprinzipien auf der Basis instruktionspsychologischer Erkenntnisse*. Waxmann, Münster.
- [Narciss, 2008] Narciss, S. (2008). Feedback strategies for interactive learning tasks. In Spector, J. M., Merrill, M. D., van Merriënboer, J. J. G., and Driscoll, M. P., editors, *Handbook of Research on Educational Communications and Technology*, pages 125–144. Lawrence Erlbaum Associates, Mahwah, NJ, 3rd edition.
- [Narciss and Huth, 2004] Narciss, S. and Huth, K. (2004). How to design informative tutoring feedback for multi-media learning. In Leutner, D.,

- Niegemann, H., and Brünken, R., editors, *Instructional design for multimedia learning. Proceedings of the 5th International Workshop of SIG 6 Instructional Design of the European Association for Research on Learning and Instruction (EARLI), June 27-29, 2002 in Erfurt*, pages 181–195, Münster. Waxmann.
- [Narciss and Huth, 2006] Narciss, S. and Huth, K. (2006). Fostering achievement and motivation with bug-related tutoring feedback in a computer-based training for written subtraction. *Learning and Instruction*, 16:310–322.
- [Narciss et al., 2004] Narciss, S., Koerndle, H., Reimann, G., and Mueller, C. (2004). Feedback-seeking and feedback efficiency in web-based learning: How do they relate to task and learner characteristics? In Gerjets, P., Kirschner, P., Elen, J., and Joiner, R., editors, *Instructional design for effective and enjoyable computer-supported learning. Proceedings of the first joint meeting of EARLI SIGs Instructional Design and Learning and Instruction*, pages 377–388, Tübingen. Knowledge Media Research Center.
- [Nicaud et al., 2003] Nicaud, J., Bouhineau, D., Chaachoua, H., and amd A. Bronner, T. H. (2003). A computer program for the learning of algebra: description and first experiment. In *Proceedings of PEG2003*, Saint Petersburg. Springer-Verlag.
- [Niss, 2002] Niss, M. (2002). Mathematical competencies and the learning of mathematics: the danish KOM project. Technical report, Roskilde University. [http://www7.nationalacademies.org/mseb/Mathematical\\_Competencies\\_and\\_the\\_Learning\\_of\\_Mathematics.pdf](http://www7.nationalacademies.org/mseb/Mathematical_Competencies_and_the_Learning_of_Mathematics.pdf).
- [Nwana, 1990] Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4:251–277.
- [Ohlsson, 1996] Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, 103(2):241–262.
- [Palomo, 2006] Palomo, A. G. (2006). Exercise system in ACTIVEMATH. Technical report, Saarland University.
- [Polushkina, 2009] Polushkina, S. (2009). Selbstreguliert modellieren lernen mit einer e-lernumgebung für schüler/innen: Kompetenzförderung

- durch lernunterstützungen. In *Tagungsband zur 43. Jahrestagung der Gesellschaft für Didaktik der Mathematik, Oldenburg, 2009*.
- [Reiss et al., 2005] Reiss, K., Moormann, M., Groß, C., and Ullrich, C. (2005). Formalized pedagogical strategies. Deliverable D20, LeActiveMath Consortium.
- [Rickel and Johnson, 1999] Rickel, J. and Johnson, W. L. (1999). Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382.
- [Roll et al., 2006] Roll, I., Alevan, V., McLaren, B. M., Ryu, E., de Baker, R. S. J., and Koedinger, K. R. (2006). The help tutor: Does metacognitive feedback improve students’ help-seeking actions, skills and learning? In [Ikeda et al., 2006], pages 360–369.
- [Rosé et al., 2001] Rosé, C. P., Jordan, P., Ringenberg, M., Siler, S., Van Lehn, K., and Weinstein, A. (2001). Interactive conceptual tutoring in atlas-andes. In Moore, J. D., Redfield, C. L., and Johnson, W. L., editors, *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future, Proceedings of AIED 2001*, pages 256–266, Amsterdam. IOS Press.
- [Sangwin, 2004] Sangwin, C. J. (2004). Assessing mathematics automatically using computer algebra and the internet. *Teaching Mathematics and its Applications*, 23(1):1–14.
- [Sangwin, 2008] Sangwin, C. J. (2008). Assessing elementary algebra with stack. *International Journal of Mathematical Education in Science and Technology*, 38(8):987–1002.
- [Schiller, 2010] Schiller, M. (2010). *Granularity Analysis for Tutoring Mathematical Proofs*. PhD thesis, Department of Computer Science, Saarland University. Submitted.
- [Shapiro, 2005] Shapiro, J. A. (2005). An algebra subsystem for diagnosing students’ input in a physics tutoring system. *International Journal of Artificial Intelligence in Education*, 15(3):205–228.
- [Siekmann et al., 2000] Siekmann, J., Benz Müller, C., Fiedler, A., Franke, A., Gogvadze, G., Horacek, H., Kohlhase, M., Libbrecht, P., Meier, A.,

- Melis, E., Pollet, M., Sorge, V., and Ullrich, C. (2000). Adaptive course generation and presentation. In Peylo, C., editor, *Proceedings of the International Workshop on Adaptive and Intelligent Webbased Education Systems held in conjunction with 5th International Conference on Intelligent Tutoring Systems (ITS'2000), Montreal, Canada. Technical Report of the Institute for Semantic Information Processing Osnabrück.*
- [Skinner, 1954] Skinner, B. (1954). The science of learning and the art of teaching. *Harvard Educational Review*, 24:86–97.
- [Smets, 1994] Smets, P. (1994). What is Dempster-Shafer's model? In Yager, R. R., Kacprzyk, J., and Fedrizzi, M., editors, *Advances in the Dempster-Shafer theory of evidence*, pages 5–34. John Wiley & Sons, Inc, New York, NY, USA.
- [Spaniol et al., 2009] Spaniol, M., Li, Q., Klamma, R., and Lau, R. W. H., editors (2009). *Advances in Web Based Learning - ICWL 2009, 8th International Conference, Aachen, Germany, August 19-21, 2009. Proceedings*, volume 5686 of *LNCS*. Springer.
- [The MONET Project, 2002] The MONET Project (2002). Monet. <http://monet.nag.co.uk/monet>.
- [Tsovaltzi et al., 2010a] Tsovaltzi, D., McLaren, B., Melis, E., Meyer, A.-K., Dietrich, M., and Gogvadze, G. (2010a). Learning from erroneous examples. In V. Aleven, J. Kay, and J. Mostow, editors, *Proceedings of Intelligent Tutoring Systems, 10th International Conference (ITS 2010)*, volume 6095 of *Lecture Notes in Computer Science, Subseries: Programming and Software Engineering*, pages 420–422, Pittsburgh, PA, USA. Springer-Verlag Berlin Heidelberg. Proceedings, Part II.
- [Tsovaltzi et al., 2010b] Tsovaltzi, D., Melis, E., McLaren, B., Meyer, A.-K., Dietrich, M., and Gogvadze, G. (2010b). Learning from erroneous examples: When and how do students benefit from them? In Wolpers, M., Kirschner, P., Scheffel, M., Lindstaedt, S., and Dimitrova, V., editors, *Proceedings of the 5th European Conference on Technology Enhanced Learning (EC-TEL 2010)*, volume 6383 of *Lecture Notes in Computer Science, Sustaining TEL: From Innovation to Learning and Practice*, pages 357–373, Barcelona, Spain. Springer-Verlag Berlin Heidelberg. Proceedings, Part II.

- [Tsovaltzi et al., 2009] Tsovaltzi, D., Melis, E., McLaren, B. M., Dietrich, M., Gogvadze, G., and Meyer., A.-K. (2009). Erroneous examples: A preliminary investigation into learning benefits. In Cress, U., Dimitrova, V., and Specht, M., editors, *Proceedings of the Fourth European Conference on Technology Enhanced Learning, Learning in the Synergy of Multiple Disciplines (EC-TEL 2009)*, number 5794 in LNCS, pages 688–693, Nice, France. Springer-Verlag Berlin Heidelberg.
- [Turner et al., 2005] Turner, T., Macasek, M., Nuzzo-Jones, G., Heffernan, N., and Koedinger, K. (2005). The assistment builder: A rapid development tool for its. In Looi, C., McCalla, G., Bredeweg, B., and Breuker, J., editors, *Proceedings of the 12th International Conference on Artificial Intelligence In Education*, pages 929–931, Amsterdam. ISO Press.
- [Ullrich, 2008] Ullrich, C. (2008). Course generation as a hierarchical task network planning problem. *KI*, 3:72–74.
- [Ullrich et al., 2004] Ullrich, C., Libbrecht, P., Winterstein, S., and Mühlenbrock, M. (2004). A flexible and efficient presentation-architecture for adaptive hypermedia: Description and technical evaluation. In *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, Joensuu, Finland.
- [Ullrich and Melis, 2009] Ullrich, C. and Melis, E. (2009). Pedagogically founded courseware generation based on htn-planning. *Expert Systems with Applications*, 36(5):9319–9332.
- [Van Lehn, 2006] Van Lehn, K. (2006). The behavior of tutoring systems, international journal of artificial intelligence in education. *International Journal of Artificial Intelligence in Education*, 16:227–265.
- [Van Lehn et al., 2005] Van Lehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. (2005). The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3).
- [Woolf et al., 2008] Woolf, B. P., Aïmeur, E., Nkambou, R., and Lajoie, S. P., editors (2008). *Intelligent Tutoring Systems, 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008, Proceedings*, volume 5091 of LNCS. Springer.

- [Zinn, 2006] Zinn, C. (2006). Supporting tutorial feedback to student help requests and errors in symbolic differentiation. In Ashley, K. and Ikeda, M., editors, *Proceedings of Intelligent Tutoring Systems 8th. International Conference ITS-2006*, volume 4053 of *LNCS*, pages 349–359.

# Appendix A

## Curve Sketching Problem

We present a commented listing of the multi-exercise on curve sketching as an example, which was also briefly addressed in Section 3.5.11.

This exercise can serve as a good reference for authors, since it includes most of the constructs of the exercise format, such as different interactive elements, manually authored and automatically generated sub exercises, and application of tutorial strategies to sub exercises.

Here the learner has to investigate properties of a polynomial function  $f(x) = x^3 - 5x^2 - 6x$ . This exercise consists of several sub exercises each corresponding to a particular task in curve sketching process.

In the preamble of the exercise, the title and other metadata are authored, such as relations to the mathematical concepts involved in this exercise, difficulty of the exercise, and the competency metadata. This top-level metadata of the exercise is used by the **Tutorial Component** of **ACTIVEMATH** for selecting this exercise when generating an adaptive course. Also, the search component of **ACTIVEMATH** allows to search for learning objects by their metadata, so the learner can find this exercise in the knowledge base by its metadata.

The problem statement introduces the function to be investigated. It is followed by a **transition**, forwarding the learner to the first subtask.

```
<exercise id="kurvendiskussion">
  <metadata>
    <Title xml:lang="en">Curve Sketch Exercise</Title>
    <extradata>
      <relation type="for"><ref xref="functions/function"/></relation>
      <relation type="for"><ref xref="functions/domain"/></relation>
      <relation type="for"><ref xref="functions/range"/></relation>
      <relation type="for"><ref xref="functions/min"/></relation>
      <relation type="for"><ref xref="functions/max"/></relation>
    </extradata>
  </metadata>
</exercise>
```

```

<relation type="for"><ref xref="calculus1/diff"/></relation>
<relation type="for"><ref xref="functions/convex"/></relation>
<relation type="for"><ref xref="functions/concave"/></relation>
<relation type="for"><ref xref="calculus1/int"/></relation>
<difficulty value="medium"/>
<competency value="model"/>
<competency value="solve"/>
</extradata>
</metadata>
<CMP xml:lang="en">Investigate properties of a polynomial function.</CMP>
<task id="problem_statement">
  <content>
    <CMP xml:lang="en">
      Consider the function
       $f(x) = x^3 - 5x^2 - 6x$ 
      <br/> In this exercise we investigate
      different properties of this function.
    </CMP>
  </content>
  <transition to="domain"/>
</task>

```

1. The first sub-exercise is to identify the domain of a function. It is a manually authored exercise realized as multiple choice questions. The task of this sub-exercise includes metadata, as described in Section 3.3.1.1. First, there is a relation to the concept of the domain of the function. Also the difficulty value of the task is "easy" and the competency value is "solve".

The task is followed by the **interaction** element, containing a multiple choice question block. The representation for multiple choice questions was introduced in Section 3.3.3.2.

```

<!-- domain selection -->
<task id="domain">
  <task_metadata>
    <relation type="for"><ref xref="functions/domain"/></relation>
    <difficulty value="easy"/>
    <competency value="solve"/>
  </task_metadata>
  <content>
    <CMP xml:lang="en">
      1. Identify the domain of the given function.
    </CMP>
  </content>
  <transition to="domain_interactive"/>
</task>
<interaction id="domain_interactive">
  <content>
    <CMP xml:lang="en">
      Choose one of the following : <br/>
      <with xref="domain_selection"/>
    </CMP>
  </content>

```



```

</content>
<interaction_map>
  <selection id="domain_selection">
    <choice><CMP> $\mathbb{R}$ </CMP></choice>
    <choice><CMP> $\{x \in \mathbb{R} \mid x > 0\}$ </CMP></choice>
    <choice><CMP> $\mathbb{R} \setminus \{0\}$ </CMP></choice>
    <choice><CMP> $\mathbb{Z}$ </CMP></choice>
  </selection>
</interaction_map>

```

In the `transition_map`, two transitions are authored. The first transition matches the correct answer, and the second default transition matches any other answer. Depending on the learner's answer, two possible KR (knowledge of result) feedbacks can follow. These feedbacks are annotated with metadata, determining the type of the feedback, as described in Section 3.3.2.

```

<transition_map>
  <transition to="domain_selection_correct">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <composite>
        <compare>1</compare>
        <compare>0</compare>
        <compare>0</compare>
        <compare>0</compare>
      </composite>
    </condition>
  </transition>
  <transition type="default" to="domain_selection_wrong">
    <diagnosis>
      <achievement value="0.0" />
    </diagnosis>
  </transition>
</answer_map>
</interaction>
<feedback id="domain_selection_correct">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="domain_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="range" />
</feedback>
<feedback id="domain_selection_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="domain_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">No. This set does not correctly
      describe the domain of the given function.<br/>
      Try again:</CMP>
  </content>
</feedback>

```

```

</content>
<transition to="domain_interactive"/>
</interaction>

```

2. The second subtask is to identify the range of the function, which is also a manually authored multiple choice exercise. The annotations here are analogous to the previous sub-exercise. The task is now related to the concept of the range of a function.

```

<!-- range selection -->

<task id="range">
  <task_metadata>
    <relation type="for"><ref xref="functions/range"/></relation>
    <difficulty value="easy"/>
    <competency value="solve"/>
  </task_metadata>
  <content>
    <CMP xml:lang="en">
      2. Identify the range of this function.
    </CMP>
  </content>
  <transition to="range_interactive"/>
</task>
<interaction id="range">
  <content>
    <CMP xml:lang="en">
      Choose on of the following :<br/>
      <with xref="range_selection"/>
    </CMP>
  </content>
  <interaction_map>
    <selection id="range_selection">
      <choice><CMP> $\mathbb{R}$ </CMP></choice>
      <choice><CMP> $\{x \in \mathbb{R} \mid x > 0\}$ </CMP></choice>
      <choice><CMP> $\{x \in \mathbb{R} \mid x < 0\}$ </CMP></choice>
      <choice><CMP> $\mathbb{R} \setminus \{0\}$ </CMP></choice>
    </selection>
  </interaction_map>
  <transition_map>
    <transition to="range_selection_correct">
      <diagnosis>
        <achievement value="1.0"/>
      </diagnosis>
      <condition>
        <composite>
          <compare>1</compare>
          <compare>0</compare>
          <compare>0</compare>
          <compare>0</compare>
        </composite>
      </condition>
    </transition>
    <transition type="default" to="range_selection_wrong">

```

```

    <diagnosis>
      <achievement value="0.0" />
    </diagnosis>
  </transition>
</transition_map>
</interaction>
<feedback id="range_selection_correct">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="range_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="crossaxis-x" />
</feedback>
<feedback id="range_selection_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="range_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">No. This set does not correctly
      describe the range of the given function.<br/>
      Try again:</CMP>
  </content>
  <transition to="range_interactive" />
</feedback>

```

**3.** The next task is to find out where the function crosses the  $x$ -axis. For this the learner has to solve the equation  $x^3 - 5x^2 - 6x = 0$ . This sub-exercise is automatically generated by the domain reasoner **Exercise Generator**. This generator takes several parameters, such as a formally represented task, and the identifier of the mathematical context that helps choosing a concrete domain reasoner for this task.

```

<!-- crossing axis x -->
<task id="crossaxis-x">
  <task_metadata>
    <relation type="for"><ref xref="functions/root"/></relation>
  </task_metadata>
  <exercise_generator name="IDEASGenerator">
    <parameter name="problemstatement">
      3. Now find the points at which the graph of the
      function is crossing the  $x$  axis. In order to
      do this, solve the following equation:
    </parameter>
    <parameter name="context">algebra.equations.polynomial</parameter>
    <parameter name="task">
       $x^3 - 5x^2 - 6x = 0$ 
    </parameter>
  </exercise_generator>

```

```
<transition to="crossaxis-y" />
</task>
```

4. The next task is to find out where the function crosses the  $y$ -axis. For this, the learner has to calculate the value of the function  $f(x)$  for  $x = 0$ . This sub-exercise is manually authored, but uses an additional automated tutorial strategy. In this strategy, after a certain number of unsuccessful attempts, the learner will be presented with a correct answer for the task. The strategy is assigned using an `exercise_generator` element. It takes several parameters, such as the identifiers of the initial and the final nodes, the identifier of a special `feedback` element, containing the correct answer to the task, and the parameter `noftrials` specifying the number of unsuccessful trials before the correct solution is presented to the learner.

```
<!-- crossing axis y -->
```

```
<task id="crossaxis-y">
  <content keep="yes">
    <CMP xml:lang="en">
      4. Now find the coordinates of crossing axis
      y. In order to do this, calculate  $f(0)$ :
    </CMP>
  </content>
  <transition to="crossaxis-y-interactive" />
</task>
<interaction id="crossaxis-y-interactive">
  <exercise_generator name="KRKCR" type="strategy">
    <parameter name="initialnode" xref="crossaxis-y-interactive" />
    <parameter name="finalnode" xref="crossaxis-y_kcr" />
    <parameter name="finalnode" xref="crossaxis-y_correct" />
    <parameter name="kcrid" xref="crossaxis-y_kcr" />
    <parameter name="noftrials">2</parameter>
  </exercise_generator>
  <content>
    <CMP xml:lang="en">
      <OMOB:
        <OMA>
          <OMS cd="relation1" name="eq" />
          <OMA>
            <OMV name="f" />
            <OMI>0</OMI>
          </OMA>
          <OMV xref="crossaxis-y_blank" />
        </OMA>
      </OMOB:
    </CMP>
  </content>
  <interaction_map>
    <blank id="crossaxis-y_blank" />
  </interaction_map>
```

The `interaction` contains a `blank` to fill in, and the `transition_map` contains three `transitions`. The first `transition` uses syntactic comparison to match the learner's input with the correct answer. The second `transition` employs a semantic comparison to match the learner's input that is not literally equal, but semantically equivalent to the correct answer. The third default `transition` is covering the case of any other answer. Finally, three KR `feedbacks` correspond to each of the `transitions`. The `feedbacks` are annotated with the corresponding metadata describing their types.

```

<transition_map>
  <transition to="crossaxis-y-correct">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <compare>0</compare>
    </condition>
  </transition>
  <transition to="crossaxis-y-equivalent">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <compare context="semantic">0</compare>
    </condition>
  </transition>
  <transition type="default" to="crossaxis-y-wrong">
    <diagnosis>
      <achievement value="0.0" />
    </diagnosis>
  </transition>
</transition_map>
</interaction>
<feedback id="crossaxis-y-correct">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="crossaxis-y-interactive" keep="yes" input_decoration="good" />
  <content keep="yes">
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="constant-sign-intervals" />
</feedback>
<feedback id="crossaxis-y-equivalent">
  <feedback_metadata>
    <type category="product" value="KR" />
    <type category="procedure" value="KH" />
  </feedback_metadata>
  <content from="crossaxis-y-interactive" keep="yes" input_decoration="near" />
  <content keep="yes">
    <CMP xml:lang="en">
      Correct! Go on and simplify your answer.
    </CMP>
  </content>

```

```

    <transition to="crossaxis-y-interactive" />
  </feedback>
  <feedback id="crossaxis-y-wrong">
    <feedback_metadata>
      <type category="product" value="KR" />
    </feedback_metadata>
    <content from="crossaxis-y-interactive" keep="yes" input_decoration="bad" />
    <content keep="yes">
      <CMP xml:lang="en">
        Something went wrong, try again :
      </CMP>
    </content>
    <transition to="crossaxis-y-interactive" />
  </feedback>
  <feedback id="crossaxis-y-kcr">
    <feedback_metadata>
      <type category="product" value="KCR" />
    </feedback_metadata>
    <content keep="yes">
      <CMP xml:lang="en">
        The correct answer is :  $f(0) = 0$ 
      </CMP>
    </content>
    <transition to="constant-sign-intervals" />
  </feedback>

```

5. The next task is to identify the constant sign intervals of the function. This exercise is manually authored and does not use any additional strategy. The learner interaction is represented using a **mapping** element, described in Section 3.3.3.4. In this mapping each of the four intervals  $(-\infty, -1)$ ,  $(-1, 0)$ ,  $(0, 6)$ , and  $(6, \infty)$  has to be mapped to either the statement  $f(x) > 0$  or to  $f(x) < 0$ .

```

<!-- constant sign intervals -->

<task id="constant-sign-intervals">
  <content keep="yes">
    <CMP xml:lang="en">
      5. Now fill in the constant sign intervals. <br/>
      We already know the roots of the function. These are :
       $x = -1$ ,  $x = 0$ , and  $x = 6$ .
      Therefore, the intervals at which the function
      has different signs are also known. <br/>
      Select the corresponding signs of the function on
      these intervals.
    </CMP>
  </content>
  <transition to="constant-sign-intervals-interactive" />
</task>
<interaction id="constant-sign-intervals-interactive">
  <content>
    <CMP xml:lang="en">
      <with xref="sign-int-mapping" />
    </CMP>
  </content>

```

```

</CMP>
</content>
<interaction_map>
  <selection type="mapping" style="table" id="sign_int_mapping">
    <selection>
      <choice><CMP>(-∞, -1)</CMP></choice>
      <choice><CMP>(-1, 0)</CMP></choice>
      <choice><CMP>(0, 6)</CMP></choice>
      <choice><CMP>(6, ∞)</CMP></choice>
    </selection>
    <selection>
      <choice><CMP>f(x) > 0</CMP></choice>
      <choice><CMP>f(x) < 0</CMP></choice>
    </selection>
  </selection>
</interaction_map>

```

In the `transition_map`, two transitions are authored. The transition matching the correct answer has  $4 \times 2 = 8$  elements, and represents the mapping matrix

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix},$$

as explained in Section 3.4.2.2. The second default transition matches all other answers. Finally, the corresponding KR feedbacks are authored for the two transitions.

```

<transition_map>
  <transition to="constant_sign_intervals_correct">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <composite>
        <compare>0</compare>
        <compare>1</compare>
        <compare>1</compare>
        <compare>0</compare>
        <compare>0</compare>
        <compare>1</compare>
        <compare>1</compare>
        <compare>0</compare>
      </composite>
    </condition>
  </transition>
  <transition type="default" to="constant_sign_intervals_wrong">
    <diagnosis>
      <achievement value="0.0" />
    </diagnosis>
  </transition>
</transition_map>
</interaction>
<feedback id="constant_sign_intervals_correct">
  <feedback_metadata>

```

```

    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="constant_sign_intervals_interactive" keep="yes"
    input_decoration="automatic" />
  <content>
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="derivative" />
</feedback>
<feedback id="constant_sign_intervals_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="constant_sign_intervals_interactive" keep="yes"
    input_decoration="automatic" />
  <content>
    <CMP xml:lang="en">Something went wrong. Try again:</CMP>
  </content>
  <transition to="constant_sign_intervals_interactive" />
</feedback>

```

6. The next task is to calculate the derivative of the function. This sub-exercise is automatically generated by the domain reasoner **Exercise Generator**. This generator takes several parameters, such as a formally represented task, and the identifier of the mathematical context that helps choosing a concrete domain reasoner for this task.

```

<!-- derivative -->
<task id="derivative">
  <task_metadata>
    <relation type="for"><ref xref="calculus1/diff"/></relation>
  </task_metadata>
  <exercise_generator name="IDEASGenerator">
    <parameter name="problemstatement">
      6. Now find the derivative of the given function
    </parameter>
    <parameter name="initialnode" xref="derivative_first_step" />
    <parameter name="context">calculus.differentiation.polynomial</parameter>
    <parameter name="task">(x3 - 5x2 - 6x)'</parameter>
  </exercise_generator>
  <transition to="critical-points" />
</task>

```

7. The next task is to find the critical points of the function. In order to do this the zeros of the derivative function have to be calculated. Thus, the quadratic equation  $3x^2 - 10x - 6 = 0$  has to be solved. This exercise is automatically generated by the domain reasoner **Exercise Generator**. As



always, among other parameters, the generator needs the formally represented task and the identifier of the context.

```
<!-- critical points -->
```

```
<task id="critical-points">
  <exercise_generator name="IDEASGenerator">
    <parameter name="problemstatement">
      7. Now find stationary points of the given function.
      In order to do this, find zeros of the equation:
    </parameter>
    <parameter name="initialnode" xref="derivative_zeros_first_step"/>
    <parameter name="context">algebra.equations.quadratic</parameter>
    <parameter name="task">3x2 - 10x - 6 = 0</parameter>
  </exercise_generator>
  <transition to="second-derivative"/>
</task>
```

8. The next task is to find the second derivative of the function. This is equivalent to finding the derivative of the derivative which is already found in one of the previous tasks, so the derivative of  $f'(x) = 3x^2 - 10x - 6$  has to be calculated. This exercise is automatically generated using the same domain reasoner as in the case of calculating the first derivative.

```
<!-- second derivative -->
```

```
<task id="second-derivative">
  <exercise_generator name="IDEASGenerator">
    <parameter name="problemstatement">
      8. Now find the second derivative:
    </parameter>
    <parameter name="initialnode" xref="second-derivative_first_step"/>
    <parameter name="context">calculus.differentiation.polynomial</parameter>
    <parameter name="task">(3x2 - 10x - 6)'</parameter>
  </exercise_generator>
  <transition to="min-or-max"/>
</task>
```

9. The next task is to find out whether the function reaches a local minimum or maximum at any of the critical points. For this, the values of the second derivative at both stationary points have to be calculated and compared to zero. If the value is positive the function has a local minimum, and if it is positive the function has a local maximum.

This exercise consists of two manually authored exercises, containing blanks for learner interaction.

```
<!-- minimum or maximum -->
```

```
<task id="min-or-max">
```

```

<content keep="yes">
  <CMP xml:lang="en">
    9. Using the second derivative test find out at which
    of the stationary points  $x_1 = \frac{5}{3} + \frac{1}{3}\sqrt{43}$ ,
     $x_2 = \frac{5}{3} - \frac{1}{3}\sqrt{43}$  does this function have local
    maximum or minimum. <br/>
  </CMP>
</content>
<transition to="min" />
</task>

```

The first sub-exercise deals with the first stationary point  $x_1$ , which will appear to be a local minimum point. This exercise uses an additional tutorial strategy, which gives the learner 2 trials before giving out the correct solution. In the content of the `interaction` we present the full OPENMATH encoding of a mathematical equation in order to show the position of the placeholder for the `blank`, contained inside the equation.

```

<!-- min -->
<task id="min">
  <task_metadata>
    <relation type="for"><ref xref="functions/min"/></relation>
  </task_metadata>
  <content keep="yes">
    <CMP xml:lang="en">
      First consider  $f''(x_1)$ :
    </CMP>
  </content>
  <transition to="min_interactive" />
</task>
<interaction id="min_interactive">
  <exercise_generator name="KRKCR" type="strategy">
    <parameter name="initialnode" xref="min_interactive" />
    <parameter name="finalnode" xref="min_kcr" />
    <parameter name="finalnode" xref="min_correct" />
    <parameter name="kcrid" xref="min_kcr" />
    <parameter name="noftrials">2</parameter>
  </exercise_generator>
  <content>
    <CMP xml:lang="en">
      <OMOBJ>
        <OMA>
          <OMS cd="relation1" name="eq" />
          <OMA>
            <OMS cd="arith1" name="minus" />
            <OMA>
              <OMS cd="arith1" name="times" />
              <OMD>6</OMD>
              <OMA>
                <OMS cd="arith1" name="plus" />
                <OMA>
                  <OMS cd="arith1" name="divide" />

```

```

      <OMD>5</OMD>
      <OMD>3</OMD>
    </OMA>
    <OMA>
      <OMS cd="arith1" name="times" />
      <OMA>
        <OMS cd="arith1" name="divide" />
        <OMD>1</OMD>
        <OMD>3</OMD>
      </OMA>
      <OMA>
        <OMS cd="arith1" name="root" />
        <OMD>43</OMD>
        <OMD>2</OMD>
      </OMA>
    </OMA>
  </OMA>
</OMA>
<OMD>10</OMD>
</OMA>
<OMV name="blank" xref="min_blank" />
</OMA>
</OMOB>
</CMP>
</content>
<interaction_map>
  <blank id="min_blank" />
</interaction_map>

```

The `transition_map` encodes transitions matching the correct answer using syntactic and semantic comparisons, as well as a default transition. Each transition leads to a corresponding feedback.

```

<transition_map>
  <transition to="min_correct">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <compare>2√43</compare>
    </condition>
  </transition>
  <transition to="min_equivalent">
    <diagnosis>
      <achievement value="0.8" />
    </diagnosis>
    <condition>
      <compare context="semantic">2√43</compare>
    </condition>
  </transition>
  <transition type="default" xref="min_wrong">
    <diagnosis>
      <achievement value="0.0" />
    </diagnosis>
  </transition>
</transition_map>

```

```

</interaction>
<feedback id="min_correct">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="min_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="min_conclusion" />
</feedback>
<feedback id="min_equivalent">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="min_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Correct, but can be further simplified.</CMP>
  </content>
  <transition to="min_interactive" />
</feedback>
<feedback id="min_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="min_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Something went wrong. Try again:</CMP>
  </content>
  <transition to="min_interactive" />
</feedback>

```

Special KCR (knowledge of correct result) feedback is authored for the needs of the tutorial strategy. Finally, the sub-exercise ends with the concluding feedback.

```

<feedback id="min_kcr">
  <feedback_metadata>
    <type category="product" value="KCR" />
  </feedback_metadata>
  <content from="min_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">
      The correct answer is :  $\sqrt{172}$ 
    </CMP>
  </content>
  <transition to="min_conclusion" />
</feedback>
<feedback id="min_conclusion">
  <feedback_metadata>
    <type category="product" value="KCR" />
  </feedback_metadata>
  <content keep="yes">
    <CMP xml:lang="en">
      Since  $\sqrt{172} > 0$ , the function
      has a minimum at the point  $x_1$ .
    </CMP>
  </content>
</feedback>

```

```

</CMP>
</content>
<transition to="max" />
</feedback>

```

The next sub-exercise deals with the second stationary point  $x_2$ , which will appear to be a local maximum point. This exercise uses the same tutorial strategy, which gives the learner 2 trials before giving out the correct solution.

```

<!-- max -->

<task id="max">
  <task_metadata>
    <relation type="for"><ref xref="functions/max"/></relation>
  </task_metadata>
  <content keep="yes">
    <CMP xml:lang="en">
      Now consider  $f''(x_2)$ :
    </CMP>
  </content>
  <transition to="max_interactive" />
</task>

<interaction id="max_interactive">
  <exercise_generator name="KRKCR" type="strategy">
    <parameter name="initialnode" xref="max_interactive" />
    <parameter name="finalnode" xref="max_kcr" />
    <parameter name="finalnode" xref="max_correct" />
    <parameter name="kcrid" xref="max_kcr" />
    <parameter name="noftrials">2</parameter>
  </exercise_generator>
  <content>
    <CMP xml:lang="en">
      <OMOBJ>
        <OMA>
          <OMS cd="relation1" name="eq" />
          <OMA>
            <OMS cd="arith1" name="minus" />
            <OMA>
              <OMS cd="arith1" name="times" />
              <OMI>6</OMI>
            <OMA>
              <OMS cd="arith1" name="minus" />
              <OMA>
                <OMS cd="arith1" name="divide" />
                <OMI>5</OMI>
                <OMI>3</OMI>
              </OMA>
            <OMA>
              <OMS cd="arith1" name="times" />
              <OMA>
                <OMS cd="arith1" name="divide" />
                <OMI>1</OMI>
                <OMI>3</OMI>
              </OMA>
            <OMA>

```

```

        <OMS cd="arith1" name="root" />
        <OMD>43</OMD>
        <OMD>2</OMD>
        </OMA>
        </OMA>
        </OMA>
        </OMA>
        <OMD>10</OMD>
        </OMA>
        <OMV name="blank" xref="max_blank" />
        </OMA>
        </OMOBJ>
        </CMP>
    </content>
    <interaction_map>
        <blank id="max_blank" />
    </interaction_map>

```

The transitions and feedbacks are similar to the previous case.

```

<transition_map>
  <transition to="max_correct">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <compare>-2√43</compare>
    </condition>
  </transition>
  <transition to="max_equivalent">
    <diagnosis>
      <achievement value="0.8" />
    </diagnosis>
    <condition>
      <compare context="semantic">-2√43</compare>
    </condition>
  </transition>
  <transition type="default" to="max_wrong">
    <diagnosis>
      <achievement value="0.0" />
    </diagnosis>
  </transition>
</transition_map>
</interaction>
<feedback id="max_correct">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="max_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="max_conclusion" />
</feedback>
<feedback id="max_equivalent">
  <feedback_metadata>

```

```

    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="max_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">Correct, but can be further simplified.</CMP>
  </content>
  <transition to="max_interactive" />
</feedback>
<feedback id="max_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="max_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">
      Something went wrong. Try again:
    </CMP>
  </content>
  <transition to="max_interactive" />
</feedback>
<feedback id="max_kcr">
  <feedback_metadata>
    <type category="product" value="KCR" />
  </feedback_metadata>
  <content from="max_interactive" keep="yes" input_decoration="automatic" />
  <content keep="yes">
    <CMP xml:lang="en">
      The correct answer is :  $-\sqrt{172}$ .
    </CMP>
  </content>
  <transition to="max_conclusion" />
</interaction>
<feedback id="max_conclusion">
  <content keep="yes">
    <CMP xml:lang="en">
      Since  $-\sqrt{172} < 0$ , the function
      has a maximum at the point  $x_2$ .
    </CMP>
  </content>
  <transition to="inflection-point" />
</interaction>

```

10. The next task is to find the inflection points of the function. In order to do so, the zeros of the second derivative have to be found. For this, the linear equation  $6x - 10 = 0$  has to be solved. This exercise is automatically generated by the domain reasoner **Exercise Generator**.

```
<!-- inflection point -->
```

```

<task id="inflection-point">
  <exercise_generator name="IDEASGenerator">
    <parameter name="problemstatement">
      10. Now find the inflection point by finding a zero
      of the second derivative.
    </parameter>
  </exercise_generator>
</task>

```

```

</parameter>
<parameter name="initialnode" xref="inflection-point-first-step"/>
<parameter name="context">algebra.equations.linear</parameter>
<parameter name="task">6x - 10 = 0</parameter>
</exercise_generator>
<transition to="convex-concave"/>
</taskn>

```

11. The next task is to choose the intervals where the function is convex and where it is concave. This is a manually authored exercise, in which the two intervals  $(-\infty, \frac{5}{3})$  and  $(\frac{5}{3}, \infty)$  are mapped to the the properties "convex" and "concave".

```

<!-- convex concave-->
<task id="convex-concave">
  <content keep="yes">
    <CMP xml:lang="en">
      11. Now Select intervals of convex/concave behaviour
      of the function, based on the sign of the second derivative.
      We know that the second derivative changes its sign at the
      point  $x_0 = \frac{5}{3}$ . So, there are two intervals :
       $(-\infty, \frac{5}{3})$  and  $(\frac{5}{3}, \infty)$ .
      Now choose where the  $f(x)$  is convex and where concave.
    </CMP>
  </content>
  <transition to="convex-concave_interactive"/>
</task>

```

The interaction consists of a 2x2 mapping.

```

<interaction id="convex-concave_interactive">
  <content>
    <CMP xml:lang="en">
      <with xref="convex-concave_mapping"/>
    </CMP>
  </content>
  <interaction_map>
    <selection type="mapping" id="convex-concave_mapping">
      <selection>
        <choice><CMP> $(-\infty, \frac{5}{3})$ </CMP></choice>
        <choice><CMP> $(\frac{5}{3}, \infty)$ </CMP></choice>
      </selection>
      <selection id="convex-concave_selection1">
        <choice><CMP>convex</CMP></choice>
        <choice><CMP>concave</CMP></choice>
      </selection>
    </selection>
  </interaction_map>

```

The transition\_map contains two transitions. The first transition matching the correct answer, has  $2 \times 2 = 4$  elements and represents the map-



ping matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The second default **transition** matches all other answers. Finally, the corresponding KR **feedbacks** are authored for the two **transitions**.

```

<transition_map>
  <transition to="convex-concave_correct">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
    <condition>
      <composite>
        <compare>0</compare><compare>1</compare>
        <compare>1</compare><compare>0</compare>
      </composite>
    </condition>
  </transition>
  <transition type="default" to="convex-concave_wrong">
    <diagnosis>
      <achievement value="1.0" />
    </diagnosis>
  </transition>
</transition_map>
</interaction>
<feedback id="convex-concave_correct">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="convex-concave_interactive" input_decoration="automatic" keep="yes" />
  <content keep="yes">
    <CMP xml:lang="en">Well done!</CMP>
  </content>
  <transition to="integral" />
</feedback>
<feedback id="convex-concave_wrong">
  <feedback_metadata>
    <type category="product" value="KR" />
  </feedback_metadata>
  <content from="convex-concave_interactive" input_decoration="automatic" keep="yes" />
  <content keep="yes">
    <CMP xml:lang="en">Incorrect. Try again:</CMP>
  </content>
  <transition to="convex-concave_interactive" />
</feedback>

```

**12.** The last task is to calculate an integral of the given function. This exercise is automatically generated by the domain reasoner for integrals.

```

<!-- integral -->
<interaction id="integral">
  <exercise_generator name="IntegralGenerator">
    <parameter name="problemstatement">

```

```

    12. Finally, find the integral of the given function
  </parameter>
  <parameter name="initialnode" xref="integral_first_step"/>
  <parameter name="context">calculus.integral</parameter>
  <parameter name="task"> $\int(x^3 - 5x^2 - 6x)dx$ </parameter>
</exercise_generator>
<transition to="conclusion"/>
</interaction>

```

Finally, the concluding feedback is presented and the exercise is finished.

```

<feedback id="conclusion">
  <content>
    <CMP xml:lang="en">
      This was the final step of our investigation. Well Done!
    </CMP>
  </content>
</feedback>
</exercise>

```

# Appendix B

## Sample Complex Strategy

In this addition we present in more detail the most complex tutorial and presentation strategies implemented by the author in `ACTIVEMATH` exercise system for the purposes of the `MAPS MOSSAIC` project, realized within the research training group on Feedback Based Quality Management in eLearning at the Technische Universität Darmstadt.

This project has been carried out in cooperation of the working groups on Didactics of Mathematics, conducted by Mrs. Prof. Dr. Bruder, and Educational Psychology, conducted by Mr. Prof. Dr. Schmitz. The tutorial strategies developed within two doctoral research projects (S. Polushkina and B. Benz) and implemented by the author of this thesis are aimed at investigating the learning effect of self-regulated learning strategy in combination with presentation of worked examples in a context of a complex modeling problem.

As briefly described in Section 3.5.9, a collection of exercises is presented to a learner, in which the sequencing of exercises, and the tutorial strategies for each exercise are varied for several user groups. The main instructional aspects investigated in this project are the self-regulated learning strategies and the usage of worked examples. These two aspects are combined in various ways within eight complex tutorial strategies, implemented by the author of this thesis.

Below we briefly describe two of these strategies, concentrating on the aspects that show the capabilities of the tutorial and presentation strategies of the `Exercise Subsystem` of `ACTIVEMATH`.

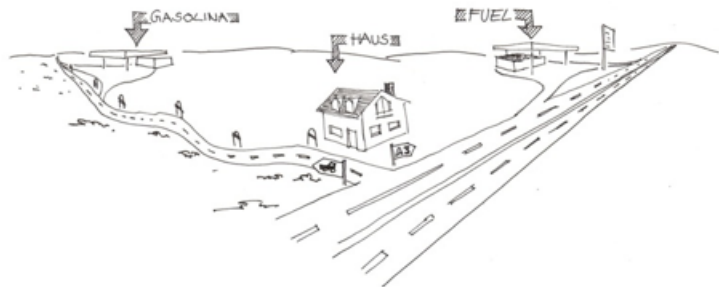
### B.0.1 Self-Regulation with Solicited Worked Example

In this sample strategy, we show the self-regulated scenario with solicited worked examples.

In this real world problem, a brother and sister (Kevin and Mareike) are solving a complex modeling problem. They want to find out which petrol station is the optimal one to tank at, considering factors like distance form their home, driving time, and petrol prices.

Mareike und Kevin wohnen mit ihren Eltern außerhalb der Stadt. In etwa gleichem Abstand von ihrem Haus gibt es zwei Tankstellen:

- Die Tankstelle „Gasolina“ ist über die Landstraße zu erreichen.
- Zur Tankstelle „Fuel“ gelangt man über die Autobahn.



Mareike und Kevin wollen ihren Eltern bei der Auswahl der besseren Tankstelle helfen. Um diese knifflige Frage zu meistern, brauchen sie Deine Hilfe.

Weiter

They also have a concrete plan of how to proceed: first, they have to simplify the question to define what is given and what has to be found, then collect information about subproblems, solve the subproblems, and give the final answer to the problem. After they decide upon this procedure, in the consequent steps a progress bar appears showing which stage of this plan they are currently working at.

Die Geschwister haben einen Plan, wie Ihr gemeinsam zu einer sinnvollen Antwort findet:

1. **Frage vereinfachen:** Ihr startet mit der kniffligen Frage. Die knifflige Frage könnt Ihr in Teilfragen zerlegen. Dazu legt Ihr fest, welche Eigenschaften des Autos und der Tankstellen für Euch wichtig für die Auswahl der Tankstelle sind.
2. **Informationen sammeln:** Nun arbeitet Ihr an den Teilfragen. Die Teilfragen könnt Ihr in mathematische Teilaufgaben umformulieren. Dazu sammelt Ihr mathematische Informationen.
3. **Teilaufgaben bearbeiten:** Jetzt sind Euch die einzelnen Teilaufgaben klar. Aus den mathematischen Teilaufgaben erhaltet Ihr mathematische Teilergebnisse. Dazu löst Ihr die Teilaufgaben.
4. **Frage beantworten:** An der Stelle habt Ihr alle mathematischen Teilergebnisse zusammen.

Den Fortschritt kannst Du ab jetzt in der oberen Leiste verfolgen.

Weiter

In the next step they have to select relevant factors in a multiple-choice exercise, using the AUC (answer until correct) tutorial strategy.

|                              |   |                          |
|------------------------------|---|--------------------------|
| Frage<br>vereinfachen<br>(1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage<br>beantworten (4) |
|------------------------------|---|--------------------------|

Kevin und Mareike überlegen gerade, was wirklich von Bedeutung ist, um die Tankstelle für ihre Eltern auszuwählen.

Bitte hilf den beiden bei der Auswahl und klicke die relevanten Einflussfaktoren an!

- Benzinpreis an den Tankstellen
- Durchfahrtshöhe der Tankstellen
- Höchste zulässige Beladung des Autos
- Fahrtdauer zu den Tankstellen
- Baujahr des Autos
- Benzinverbrauch für die Fahrt zu den Tankstellen



Weiter

After the relevant factors are selected, the main subproblems are identified.

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Super! Nun steht fest, was Du alles wissen musst, um die Frage nach der besseren Tankstelle zu beantworten.

|                               |  |                                |
|-------------------------------|--|--------------------------------|
| Fahrtdauer zu den Tankstellen | Benzinverbrauch für die Fahrt zu den Tankstellen | Benzinpreis an den Tankstellen |
|-------------------------------|--|--------------------------------|

Mareike und Kevin beschäftigen sich zuerst mit dem Benzinpreis, danach mit der Fahrtdauer und zuletzt mit dem Benzinverbrauch.

Sie wollen für jede Teilfrage alle nötigen **Informationen sammeln** und die jeweiligen **Teilergebnisse bestimmen**. Dadurch können sie die Unterschiede zwischen den beiden Tankstellen erkennen.

Weiter

The learner is presented with the first problem which consists of several subtasks. In the self-regulation phase the learner has to bring the subtasks in the correct order by dragging the boxes with task statements.

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Sie müssen also mit mehreren Schritten zum Ziel kommen. Bitte hilf Mareike und Kevin dabei, die Schritte in eine sinnvolle Reihenfolge zu bringen.

|  |
|--|
| Restvolumen des Tanks  |
| Benzinpreis für das Volltanken an den Tankstellen: Gasolina / Fuel |
| Unterschied im Benzinpreis für das Volltanken                      |
| Benzinpreis pro Liter an den Tankstellen: Gasolina / Fuel          |

*Ein Kästchen entspricht jeweils einem der oben beschriebenen Schritte. Ziehe die Kästchen in die richtige Reihenfolge.*

Weiter

If the selected order of subtasks is possible the learner is allowed to proceed. He is also allowed to go back and change the selected order.

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Du kannst mit dem nächsten Schritt in dieser Reihenfolge fortfahren.

After the order is selected the learner is presented with the first subtask. On the left, the structure of the subtasks is shown. The learner is given 3 trials for the task, after failing which the correct solution is presented.

|  |   |
|--|---|
| Restvolumen des Tanks<br><input type="text" value="1"/>  |   |
| Benzinpreis (Volltanken) pro Liter an den Tankstellen:<br>Gasolina <input type="text"/> €<br>Fuel <input type="text"/> €   | Restvolumen des Tanks: <span style="background-color: red; color: white;">30</span> l   |
| Unterschied im Benzinpreis für das Volltanken :<br>An der Tankstelle <input type="text"/><br>kostet das Volltanken <input type="text"/> €<br>weniger, als an der Tankstelle <input type="text"/> | Leider Falsch! Versuchen Sie es noch einmal.<br>Aus dem Fahrzeugschein können Mareike und Kevin ablesen, wie viele Liter der Tank des Autos insgesamt fasst: Das sind 60 Liter.   |
| Benzinpreis pro Liter an den Tankstellen:<br>Gasolina <input type="text"/> € / l<br>Fuel <input type="text"/> € / l  | Die Geschwister gehen davon aus, dass der Tank nicht ganz leer ist, wenn ihre Eltern bei der Tankstelle ankommen. Sie nehmen an, dass bei Erreichen der Tankstelle noch 10 Liter im Tank sind.<br>Wie viele Liter passen dann noch in den Tank? |
|  | Bitte trage das Restvolumen des Tanks in das Kästchen unten ein.<br>Restvolumen des Tanks: <input type="text" value=""/>  |

After solving the first subtask, the student wants to proceed, but the system tells him that the next subtasks in the sequence as the student has selected before is not possible so he has to change the sequence of the rest of subtasks in order to be able to proceed.

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Du kannst den nächsten Schritt in dieser Reihenfolge noch nicht machen. Du müsstest also die Reihenfolge der einzelnen Schritte ändern.

Reihenfolge der Schritte ändern

The selection interface is then presented again and the student can select the new sequence for the remaining subtasks.

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Restvolumen des Tanks

|  |
|--|
| Benzinpreis pro Liter an den Tankstellen: Gasolina / Fuel          |
| Benzinpreis für das Volltanken an den Tankstellen: Gasolina / Fuel |
| Unterschied im Benzinpreis für das Volltanken                      |

*Ein Kästchen entspricht jeweils einem der oben beschriebenen Schritte. Ziehe die Kästchen in die richtige Reihenfolge.*

Weiter

After reordering the remaining subtasks the learner can proceed further

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Du kannst mit dem nächsten Schritt in dieser Reihenfolge fortfahren.

Reihenfolge der Schritte ändern   Weiter



The next subtask starts with the working example dealing with a problem similar to the current subtask.

Frage  
vereinfachen (1)

Informationen sammeln (2) und Teilaufgaben bearbeiten (3)

Frage  
beantworten (4)

**Restvolumen des Tanks**  
 l

**Benzinpreis pro Liter an den Tankstellen:**  
 Gasolina  € / l  
 Fuel  € / l

**Unterschied im Benzinpreis für das Volltanken :**  
 An der Tankstelle   
 kostet das Volltanken  €  
 weniger, als an der Tankstelle


**Benzinpreis (Volltanken) pro Liter an den Tankstellen:**  
 Gasolina  €  
 Fuel  €

Mareike und Kevin grübeln, wie sie den **Benzinpreis pro Liter** bestimmen können.  
 Mareike erinnert sich an ein Beispiel aus der Schule:  
 Ein 5-kg-Sack Bio-Kartoffeln kostet 10 €. Der Preis für 1 kg ist gefragt. Mareike schaut sich die Rechnung in ihrem Schulheft noch mal an:

| Menge (in kg) | Preis (in €)          |
|---------------|-----------------------|
| :5 ↻ 5<br>1   | 10 ↻ :5<br>10 : 5 = 2 |

Die Geschwister überlegen sich, wie sie dieses Beispiel auf die Berechnung des Benzinpreises übertragen können.

Kevin: Ja klar! Die Kartoffeln entsprechen dem Benzin und die Kilogramms den Litern.



Weiter

After the worked example is presented, the learner is forwarded to the analogous problem in the current subtask. He is also given an opportunity to view the previously given worked example in the same window, by clicking the button on the right.

Frage vereinfachen (1) Informationen sammeln (2) und Teilaufgaben bearbeiten (3) Frage beantworten (4)

Restvolumen des Tanks  
50 l

Benzinpreis pro Liter an den Tankstellen:  
Gasolina € / l  
Fuel € / l

Unterschied im Benzinpreis für das Volltanken :  
An der Tankstelle  kostet das Volltanken  €  
weniger, als an der Tankstelle

Benzinpreis (Volltanken) pro Liter an den Tankstellen:  
Gasolina €  
Fuel €

Mit Hilfe der Kassenbons können Mareike und Kevin den **Benzinpreis pro Liter** berechnen.

| GASOLINA       |              | FUEL           |              |
|----------------|--------------|----------------|--------------|
| Volumen (in l) | Preis (in €) | Volumen (in l) | Preis (in €) |
| 35             | 53,20        | 30             | 47,40        |

Beispiel zeigen

Ein 5-kg-Sack Bio-Kartoffeln kostet 10 €. Der Preis für 1 kg ist gefragt. Mareike schaut sich die Rechnung in ihrem Schulheft noch mal an:

| Menge (in kg) | Preis (in €) |
|---------------|--------------|
| 5             | 10           |
| 1             | $10 : 5 = 2$ |

Welche Angaben auf den Kassenbons sind dafür wichtig?  
Berechne den Benzinpreis pro Liter auf dem Papier und trage die Ergebnisse in die vorgesehenen Felder ein:  
Wenn Deine Ergebnisse richtig sind, gelangst Du auf die nächste Seite.

Gasolina : \_\_\_\_\_ €  
Fuel : \_\_\_\_\_ €

Weiter

Similar strategy is used for other subtasks. Finally, after all the subtasks are done and the summarizing feedback is presented.

Restvolumen des Tanks  
50 l

Benzinpreis pro Liter an den Tankstellen:  
Gasolina 1,52 € / l  
Fuel 1,58 € / l

Benzinpreis (Volltanken) pro Liter an den Tankstellen:  
Gasolina 76 €  
Fuel 79 €

Unterschied im Benzinpreis für das Volltanken :  
An der Tankstelle  Gasolina  
kostet das Volltanken  3 €  
weniger, als an der Tankstelle  Fuel

SUPER! Du hast zusammen mit Mareike und Kevin den **Unterschied im Benzinpreis für das Volltanken** bestimmt:

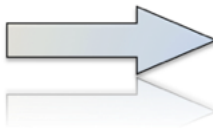
An der Tankstelle Gasolina kostet das Volltanken 3€ weniger, als an der Tankstelle Fuel.

Damit steht das erste Teilergebnis fest.

Weiter

After that, the reflection upon the changes in the students subtask planning is presented. This feedback is automatically generated by the system using the history of learner's actions. Finally, the student is asked to describe his solving strategy.

Schau Dir an, welche Reihenfolge der Schritte Du gewählt hattest, um diese Teilfrage zu bearbeiten.

|  |   |  |
|--|---|--|
| Restvolumen des Tanks  |  | Restvolumen des Tanks  |
| Benzinpreis für das Volltanken an den Tankstellen: Gasolina / Fuel |   | Benzinpreis pro Liter an den Tankstellen: Gasolina / Fuel          |
| Unterschied im Benzinpreis für das Volltanken                      |   | Benzinpreis für das Volltanken an den Tankstellen: Gasolina / Fuel |
| Benzinpreis pro Liter an den Tankstellen: Gasolina / Fuel          |   | Unterschied im Benzinpreis für das Volltanken                      |

Beschreibe Dein Vorgehen!

[Weiter](#)

After the reflection on the first task, the next task in the modeling problem is selected. Each of the tasks is represented as a separate exercise, so it is possible to apply different strategies to it. In this version, the self-regulation strategy is applied to all tasks.

|                        |   |                       |
|------------------------|---|-----------------------|
| Frage vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|------------------------|---|-----------------------|

Benzinpreis an den Tankstellen


Unterschied im Benzinpreis für das Volltanken:

An der Tankstelle

kostet das Volltanken  €

weniger, als an der Tankstelle

Fahrdauer zu den Tankstellen



Benzinverbrauch für die Fahrt zu den Tankstellen

Mareike und Kevin gehen nun zur nächsten Teilfrage über.

[Weiter](#)

The learner orders the subtasks as in the previous case and solves them using the same strategy employing worked examples as hints.

|                           |   |                          |
|---------------------------|---|--------------------------|
| Frage<br>vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage<br>beantworten (4) |
|---------------------------|---|--------------------------|

Bitte hilf Mareike und Kevin dabei, die Schritte in eine sinnvolle Reihenfolge zu bringen.

|  |
|--|
| Fahrdauer zu den Tankstellen:<br>Gasolina / Fuel             |
| Unterschied in der Fahrdauer zu den Tankstellen              |
| Entfernung Haus - Tankstelle:<br>Landstraße / Autobahn       |
| Durchschnittliche Fahrgeschwindigkeit: Landstraße / Autobahn |

Ein Kästchen entspricht jeweils einem der oben beschriebenen Schritte. Ziehe die Kästchen in die richtige Reihenfolge.

Weiter

After all the tasks are completed the overview of the results is presented to the learner again.

|                           |   |                       |
|---------------------------|---|-----------------------|
| Frage<br>vereinfachen (1) | Informationen sammeln (2) und Teilaufgaben bearbeiten (3) | Frage beantworten (4) |
|---------------------------|---|-----------------------|

|   |   |  |
|---|---|--|
| Benzinpreis an den Tankstellen  | Fahrdauer zu den Tankstellen  | Benzinverbrauch für die Fahrt zu den Tankstellen   |
| Unterschied im Benzinpreis für das Volltanken:<br>An der Tankstelle<br>Gasolina<br>kostet das Volltanken<br>3 €<br>weniger, als an der Tankstelle<br>Fuel | Unterschied in der Fahrdauer:<br>Zu der Tankstelle<br>Gasolina<br>fährt man<br>5 min<br>länger, als zu der Tankstelle<br>Fuel | Unterschied im Benzinverbrauch:<br>Für die Fahrt zu der Tankstelle<br>Gasolina<br>verbraucht man<br>0,025 l<br>weniger Benzin, als für die Fahrt zu der Tankstelle<br>Fuel |

Weiter

After that the learner is requested to submit the final answer to the modeling problem.

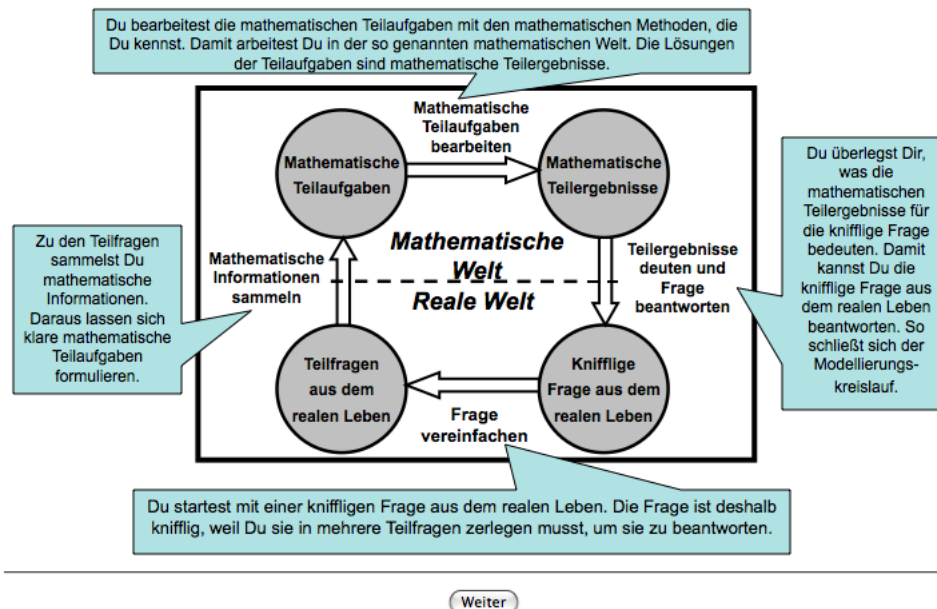
Nun packen Mareike und Kevin den vierten und letzten Punkt an: Sie wollen sich die Teilergebnisse verdeutlichen und die Frage nach der Tankstelle beantworten.

Hast Du schon eine Meinung dazu? Was sollen Deiner Meinung nach Mareike und Kevin ihren Eltern empfehlen?

Gib eine Empfehlung und begründe sie!

Weiter

Finally, a series of feedbacks is presented reviewing the process of solving a modeling problem, which concludes the exercise.



## B.0.2 Control Strategy

The control strategy is designed to allow the student to edit the subtasks independently, without the self regulation step. However, if the subtask the student decides to work upon can not be performed yet because it depends on some other subtask, the student is forwarded back to the task selection page. In the beginning all the subtasks are presented without any order in the form of clickable buttons.



Sie müssen also mit mehreren Schritten zum Ziel kommen. Führe die Schritte in einer sinnvollen Reihenfolge aus.

Tankmenge

Benzinpreis für das Volltanken an den Tankstellen: Gasolina / Fuel

Unterschied im Benzinpreis für das Volltanken

Benzinpreis pro Liter an den Tankstellen: Gasolina / Fuel

By clicking on a button, the student is forwarded to the corresponding subtask.



Nun kennen Mareike und Kevin den **Benzinpreis pro Liter** an den beiden Tankstellen und die **Tankmenge**. Damit wollen sie den **Benzinpreis für das Volltanken** an den beiden Tankstellen berechnen. Der **Benzinpreis pro Liter** der beiden Tankstellen und die **Tankmenge** sind also schon bekannt.

| Volumen (in l) | Preis: Gasolina (in €) | Preis: Fuel (in €) |
|----------------|------------------------|--------------------|
| 1              | 1,52                   | 1,58               |
| 50             | ?                      | ?                  |

Berechne damit den **Benzinpreis für das Volltanken** an den beiden Tankstellen und trage Deine Ergebnisse ein:

Gasolina:  €

Fuel:  €

Weiter

Tipp 1

Each of the subtasks has series of hints, that are sequenced in the order of occurrence.

Frage vereinfachen (1) Informationen sammeln (2) und Teilaufgaben bearbeiten (3) Frage beantworten (4)

Der **Benzinpreis pro Liter** der beiden Tankstellen und die **Tankmenge** sind also schon bekannt.

| Volumen (in l) | Preis: Gasolina (in €) | Preis: Fuel (in €) |
|----------------|------------------------|--------------------|
| 1              | 1,52                   | 1,58               |
| 50             | ?                      | ?                  |

Vervollständige diese Tabellen und berechne damit für beide Tankstellen **den Preis für das Volumen, das getankt werden soll.**

Gasolina

Fuel

| Volumen (in l) | Preis (in €) | Volumen (in l) | Preis (in €) |
|----------------|--------------|----------------|--------------|
| 1              | 1,52         | 1              | 1,58         |
| 50             | ?            | 50             | ?            |

Berechne damit den **Benzinpreis für das Volltanken** an den beiden Tankstellen und trage Deine Ergebnisse ein:

Gasolina:  €  
 Fuel:  €

Weiter Tipp 2

Each next hint is placed in place of the previous one, and the previous one disappears.

Frage vereinfachen (1) Informationen sammeln (2) und Teilaufgaben bearbeiten (3) Frage beantworten (4)

Der **Benzinpreis pro Liter** der beiden Tankstellen und die **Tankmenge** sind also schon bekannt.

| Volumen (in l) | Preis: Gasolina (in €) | Preis: Fuel (in €) |
|----------------|------------------------|--------------------|
| 1              | 1,52                   | 1,58               |
| 50             | ?                      | ?                  |

Vervollständige diese Tabellen und berechne damit für beide Tankstellen **den Preis für das Volumen, das getankt werden soll.**

Gasolina

Fuel

| Volumen (in l) | Preis (in €) | Volumen (in l) | Preis (in €) |
|----------------|--------------|----------------|--------------|
| 1              | 1,52         | 1              | 1,58         |
| 50             | ?            | 50             | ?            |

Berechne damit den **Benzinpreis für das Volltanken** an den beiden Tankstellen und trage Deine Ergebnisse ein:

Gasolina:  €  
 Fuel:  €

Bestimme den Wert für die Tankstelle Gasolina.  
 Gehe für die Tankstelle Fuel ähnlich vor, um den Wert zu bestimmen.

Weiter

After solving the current task, the student is forwarded again to the task selection page. The tasks that are already performed are not clickable and are filled with the correct solution.

| Frage vereinfachen (1)                        | Informationen sammeln (2) und Teilaufgaben bearbeiten (3)                           |
|---|---|
| Tankmenge<br>50 l                             | Benzinpreis (Volltanken) an den Tankstellen:<br>Gasolina 76 €<br>Fuel 79 €          |
| Unterschied im Benzinpreis für das Volltanken | Benzinpreis pro Liter an den Tankstellen:<br>Gasolina 1,52 € / l<br>Fuel 1,58 € / l |

After all the subtasks are solved, the next part of the modeling problem is presented. After solving all parts, the review of the modeling process is presented in the form of a sequence of feedbacks.

In total, eight different strategies were implemented by the author as it was required by the conditions of the study at TU-Darmstadt, in which different scaffolding was used to elaborate on subtask selection process, in combination with solicited or unsolicited presentation of worked examples and sequenced hints for each subtask.