E-Assessment of Creative Tasks

Herbert Kuchen

joint work with: Dominik Böhm, Susanne Gruttmann, Tim Majchrzak,
Claus Usener, Christoph Lembeck, Roger Müller, Marko Ernsting

European Research Center for Information Systems
> Introduction

• regular assessment of the learning progress required

• identify and evaluate teaching success

• mass lectures, decreasing resources and low personnel capacities

• computer-supported assessments help to reduce time and costs
> Formative Assessments

• regular exercises as complementary service for lectures:

<table>
<thead>
<tr>
<th>&gt; Students</th>
<th>&gt; Lecturers / Tutors</th>
</tr>
</thead>
<tbody>
<tr>
<td>• get an active part in learning process</td>
<td>• get an overview on learning progress of students</td>
</tr>
<tr>
<td>• reflect lecture content</td>
<td>• detect whether educational objectives are achieved</td>
</tr>
<tr>
<td>• practice and consolidate newly learned knowledge</td>
<td></td>
</tr>
<tr>
<td>• generate skills while solving corresponding tasks</td>
<td></td>
</tr>
</tbody>
</table>

> Formative assessments can enhance quality of education <
> Processes in Traditional Formative Assessments

**Preparation**
- planning assessment
- defining topics and goals
- creating exercises
- generating sample solutions

**Execution**
- collecting exercise sheets
- solving exercises
- generating submissions
- submitting work

**Follow-up**
- collecting solutions
- evaluating and marking solutions
- providing feedback
- compiling statistics
> E-Assessment Systems Today

• provide **multiple choice** questions, insertion of **short text**, and other simple forms of questions

• existing systems check **knowledge** rather than skills

> How to check creative skills? <
Creative Forms of E-Assessment

• Mathematical proofs
• Programming
• …
> Agenda

> Introduction

> Assessments in Mathematics

> The E-Assessment System EASy

> E-Assessment of Java Programs

> Conclusion
> Formative Assessment of Mathematical Proofs

• proof exercises indispensable in math-related lectures

• complexity proofs is often high

• evaluating and grading proofs is complex and time-consuming

> Electronic support for assessment of mathematical proofs? <
Shortcomings of Existing Systems for Mathematics

- only few e-assessment systems focus on mathematics
- simple knowledge checks with multiple choice etc.
- some provide support for numeric and symbolic calculations
- some systems support propositional logic
- no literature on processes for e-assessment of proofs in general

Electronic assessment of proofs is not provided
> Agenda

> Introduction

> Assessment in Mathematics

> The E-Assessment System EASy

> E-Assessment of Java Programs

> Conclusion
E-Assessment of Creative Tasks

> Requirements

- Proof Strategies
- Course-specific Rules
- Mathematical Correctness
- Stepwise Argumentation
- Ease of Handling
> The E-Assessment System EASy

- allows to establish a mathematical proof
  - step by step
  - by applying predefined rules and strategies

- a student cannot proof wrong theorems (but can get stuck)

- EASy implemented as Java applet

- based on term rewriting internally
> Using EASy for Proof Exercises
> Evaluation of EASy

• Application of EASy in „Data Structures and Algorithms“ (250 students):

<table>
<thead>
<tr>
<th>&gt; Advantages</th>
<th>&gt; Disadvantages</th>
</tr>
</thead>
</table>
| **Students** | • provides proof strategies and applicable rules  
                • controls correct use of rules  
                • supports to learn proof structure | • familiarisation complex  
                • first proofs time consuming  
                • small steps: proof of obvious term conversions necessary  
                • electronic proving vs. manual skills |
| **Tutors**   | • reduces correction effort significantly  
                • easier to read and well-structured  
                • enhances quality of submissions  
                • facilitates distribution and collection of solutions | • none |
> Evaluation of EASy
E-Assessment of Creative Tasks

> Agenda

> Introduction

> Assessment in Mathematics

> The E-Assessment System EASy

> E-Assessment of Java Programs

> Conclusion
E-Assessment of Java Programs

0. Exercise Sheet
Given Classes
Example Solution

Tutor

1. Exercise Sheet
Given Classes
Test Cases

Webserver
EASy Manager
Hidden
Test Cases

2. Solution Classes

3. Feedback

StudentPC
EASy Java Client
E-Assessment of Java Programs

• the EASy Manager generates a set of test cases from the example solution

• it checks the uploaded classes of the student
  • syntactically
  • w.r.t. programming style
  • using the given and hidden test cases

• it gives feedback to the student
Automatic Generation of Test Cases

- **Symbolic Java Virtual Machine**
  - Frame Stack
    - Frame
      - Return Addr.
      - Param.
      - loc. Vars.
      - Operand Stack
  - Heap
  - PC
  - Choice Point Stack
  - Trail

- **Constraint Solver Manager**
  - Constraint Stack

- **Constraint Solvers**
  - Linear
    - Simplex
  - Gauss
  - Elimination
  - Nonlinear
    - Bisection
    - Buchberger

- **Symbolic Execution** of Java-Byte-Code by SJVM
- **On branching instructions** (e.g. if_cmpgt, invokevirtual):
  - Constraint solver determines remaining alternatives
  - Alternatives are handled by successively by **backtracking mechanism**
Symbolic Evaluation: idiv

Frame

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>v</td>
<td>mid</td>
<td>low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>up</td>
<td></td>
</tr>
</tbody>
</table>

PC

22

CP–Stack

⋯⋯

Trail

⋯⋯

Cstrt Stack

⋯⋯

Heap

+ 2

0 0 1

⋯⋯

⋯⋯

⋯⋯
Symbolic Evaluation: \texttt{idiv}

Frame

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>v</td>
<td>mid</td>
<td>low</td>
</tr>
<tr>
<td>\cdots</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PC

23

\texttt{CP-Stack}

\cdots

\texttt{Trail}

\cdots

\texttt{Cstrt Stack}

\cdots

Heap

\cdots

\texttt{?}

0

0

1

\texttt{+}

2
Symbolic Evaluation: \textit{idiv}

<table>
<thead>
<tr>
<th>Frame</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ret. Addr.</td>
<td>23</td>
</tr>
<tr>
<td>Param.</td>
<td></td>
</tr>
<tr>
<td>this</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td></td>
</tr>
<tr>
<td>loc. Vars.</td>
<td>CP-Stack</td>
</tr>
<tr>
<td>mid</td>
<td>this</td>
</tr>
<tr>
<td>low</td>
<td>v</td>
</tr>
<tr>
<td>up</td>
<td>mid</td>
</tr>
<tr>
<td>Operand Stack</td>
<td>Trail</td>
</tr>
<tr>
<td>Stack</td>
<td>Cstrat Stack</td>
</tr>
</tbody>
</table>

Heap
Saving Previous Values: istore 4
Saving Previous Values: `istore 4`

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>v</td>
<td>mid</td>
<td>low</td>
</tr>
</tbody>
</table>

Frame

Heap

PC

CP-Stack

Trail

Cstrt Stack
Branching: if_icmpgt 61

Frame

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>⋮</td>
<td>this</td>
<td>v</td>
<td>mid</td>
</tr>
</tbody>
</table>

PC

14

CP-Stack

⋯⋯

Trail

⋯⋯

Cstrt Stack

⋯⋯

Heap

⋯⋯

⋯⋯

⋯⋯

⋯⋯

⋯⋯

⋯⋯

⋯⋯

⋯⋯
Branching: if_icmpgt 61

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>v</td>
<td>mid</td>
<td>low</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Frame

Stack

CP-Stack

PC: 17

Trail

Heap

Cstrrt Stack
Constraint Solvers

• General requirements:
  • Incremental
  • Support backtracking
  • Allow to compute a concrete solution → test case

• Dual simplex algorithm + branch & cut algorithm for linear mixed-integer constraints
  • Uses special interval arithmetic to avoid rounding errors

• Numeric non-linear (bisection) solver

• Future work: add SMT solver
E-Assessment of Creative Tasks

Bisection Solver

- Numerical approach to solution of polynomial equations
- Successive decomposition of solution space
- Each subspace $R$ is checked for roots:
  - $\min \{ p(x) \mid x \in R \} < 0$ and $\max \{ p(x) \mid x \in R \} > 0$?
  - Yes: decompose $R$
  - No: discard $R$
Soundness and Completeness?

• Soundness due to interval arithmetic

• Completeness:
  • Theoretically impossible (→ halting problem)
  • No serious problem in practice
## Which Constraints Appear?

<table>
<thead>
<tr>
<th>Example</th>
<th>Type of Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear</td>
</tr>
<tr>
<td>Ackermann</td>
<td>√</td>
</tr>
<tr>
<td>Binary search</td>
<td>√</td>
</tr>
<tr>
<td>bubblesort</td>
<td>√</td>
</tr>
<tr>
<td>Bresenham</td>
<td>√</td>
</tr>
<tr>
<td>factorial</td>
<td>√</td>
</tr>
<tr>
<td>Gaussian elimination</td>
<td>√</td>
</tr>
<tr>
<td>GCD</td>
<td>√</td>
</tr>
<tr>
<td>histogram</td>
<td>√</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>√</td>
</tr>
<tr>
<td>Matrix multiplication</td>
<td>√</td>
</tr>
<tr>
<td>Text search</td>
<td>√</td>
</tr>
<tr>
<td>log</td>
<td></td>
</tr>
<tr>
<td>sin</td>
<td></td>
</tr>
<tr>
<td>sqrt</td>
<td></td>
</tr>
</tbody>
</table>
> Agenda

> Introduction

> Assessment in Mathematics

> The E-Assessment System EASy

> E-Assessment of Java Programs

> Conclusion
> Conclusion

- development, application and evaluation of the EASy prototype
- e-assessments of mathematical proofs
- e-assessment of Java programs based on test cases generated from an example solution
- design, application and evaluation of a process model for formative e-assessment of proofs and Java programs