OpenUniversiteitNederland

Adapting Mathematical Domain Reasoners

Bastiaan Heeren [1] Johan Jeuring [1,2]

[1] Open Universiteit Nederland
[2] Universiteit Utrecht, The Netherlands

9 July 2010 (MKM 2010)
Paris, France

MATH-BRIDGE EDUCATION SOLUTION

nkbw

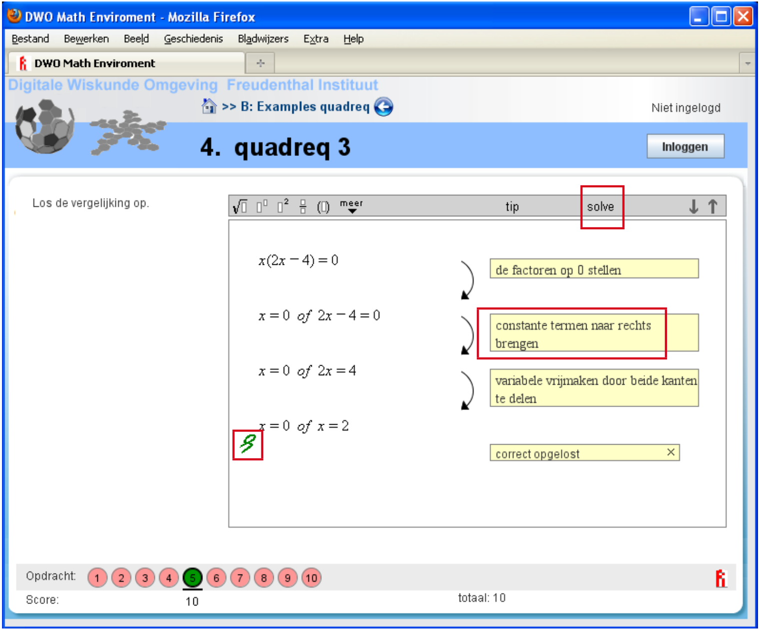
Introduction

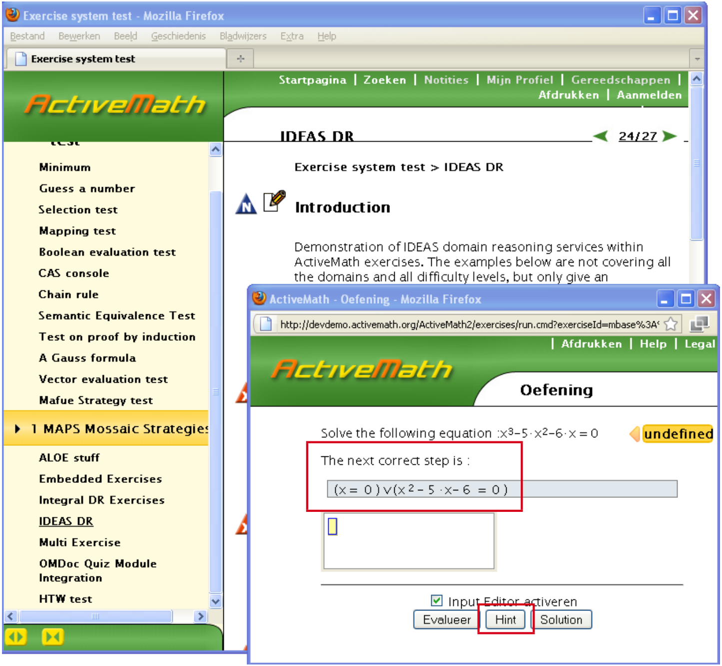
- Mathematical learning environments typically offer a wide variety of interactive exercises
- Exercise-specific parts are often delegated to specialized domain reasoners
- Design principles for instructive feedback:
  - Solve problems as the learner does
  - Show how the software solves problems
  - Make the system customizable
- Different groups of users have different customization requirements

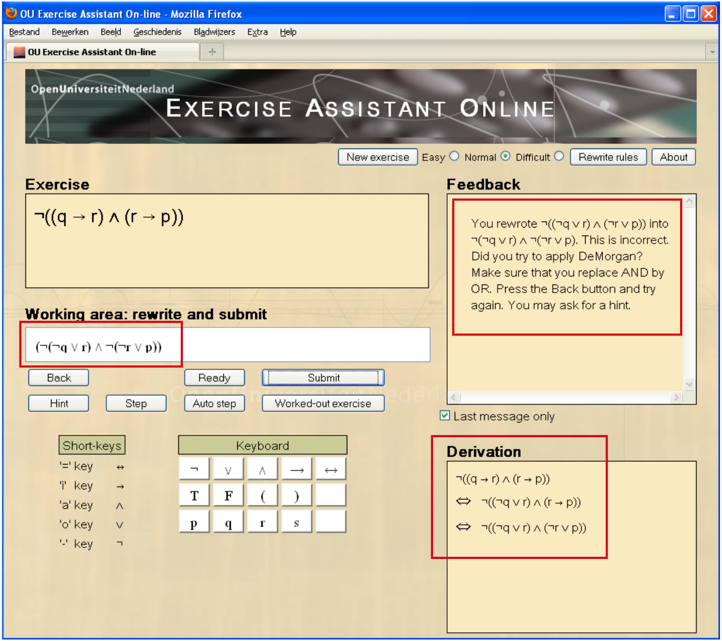
Introduction

- Mathematical learning environments typically offer a wide variety of interactive exercises
- Exercise-specific parts are often delegated to specialized domain reasoners
- Design principles for instructive feedback:
  - Solve problems as the learner does
  - Show how the software solves problems
  - Make the system customizable
- Different groups of users have different customization requirements

Examples of environments that use our domain reasoner...


DWO Math Enviroment - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
DWO Math Enviroment
Digitale Wiskunde Omgeving Freudenthal Instituut
>> B: Examples quadreq
Niet ingelogd
4. quadreq 3
Inloggen
Los de vergelijking op.
meer
tip
solve
x(2x − 4) = 0
x = 0 of 2x − 4 = 0
x = 0 of 2x = 4
x = 0 of x = 2
de factoren op 0 stellen
constante termen naar rechts brengen
variabele vrijmaken door beide kanten te delen
correct opgelost
Opdracht: 1 2 3 4 5 6 7 8 9 10
Score: 10
totaal: 10


Exercise system test - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Exercise system test
ActiveMath
Startpagina | Zoeken | Notities | Mijn Profiel | Gereedschappen |
Afdrukken | Aanmelden
IDEAS DR
24/27
Minimum
Guess a number
Selection test
Mapping test
Boolean evaluation test
CAS console
Chain rule
Semantic Equivalence Test
Test on proof by induction
A Gauss formula
Vector evaluation test
Mafue Strategy test
1 MAPS Mossaic Strategies
ALOE stuff
Embedded Exercises
Integral DR Exercises
IDEAS DR
Multi Exercise
OMDoc Quiz Module Integration
HTW test
Exercise system test > IDEAS DR
Introduction
Demonstration of IDEAS domain reasoning services within ActiveMath exercises. The examples below are not covering all the domains and all difficulty levels, but only give an
ActiveMath - Oefening - Mozilla Firefox
http://devdemo.activemath.org/ActiveMath2/exercises/run.cmd?exerciseId=mbase%3A
| Afdrukken | Help | Legal
ActiveMath
Oefening
Solve the following equation :x3−5·x2−6·x = 0
undefined
The next correct step is :
(x = 0) v(x 2 − 5 ·x− 6 = 0)
Input Editor activeren
Evalueer
Hint
Solution


OU Exercise Assistant On-line - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
OU Exercise Assistant On-line
OpenUniversiteitNederland
EXERCISE ASSISTANT ONLINE
New exercise
Easy
Normal
Difficult
Rewrite rules
About
Exercise
¬((q → r) ∧ (r → p))
Working area: rewrite and submit
(¬(¬q ∨ r) ∧ ¬(¬r ∨ p))
Back
Ready
Submit
Hint
Step
Auto step
Worked-out exercise
Short-keys
'=' key ↔
'i' key →
'a' key ∧
'o' key ∨
'-' key ¬
Keyboard
¬ ∨ ∧ → ↔
T F ( )
p q r s
Feedback
You rewrote ¬((¬q ∨ r) ∧ (¬r ∨ p)) into ¬(¬q ∨ r) ∧ ¬(¬r ∨ p). This is incorrect. Did you try to apply DeMorgan? Make sure that you replace AND by OR. Press the Back button and try again. You may ask for a hint.
Last message only
Derivation
¬((q → r) ∧ (r → p))
⇔ ¬((¬q ∨ r) ∧ (r → p))
⇔ ¬((¬q ∨ r) ∧ (¬r ∨ p))

Outline of presentation

1. Introduction
2. Case studies
3. Concepts and representation of knowledge
4. Adaptation and configuration
5. Conclusions

Perspectives for customization

1. Learners
   - Customize exercises to their level of expertise
2. Teachers
   - Specific requests how an exercise should be solved
   - Good understanding of learner's capabilities
   - Tailor exercises at a high level
3. Mathematical learning environments
   - Front-end for practicing mathematical problem solving
   - Many components are related to domain reasoners
4. Domain reasoners
   - Reusability and maintainability of code
   - Representation of (layered) mathematical knowledge

Learner: change level of detail

- Learners want to change level of detail (presented by the learning environment)
  - Smaller steps, e.g. $\sqrt{20} = \sqrt{4}\sqrt{5} = 2\sqrt{5}$
  - Only final answer

$$2x^2 + 4x - 8 = 0$$

$\Rightarrow$ *simplify polynomial*

$$x^2 + 2x - 4 = 0$$

$\Rightarrow$ *quadratic formula* $(a = 1, b = 2, c = -4, D = 20)$

$$x = \frac{-2+\sqrt{20}}{2} \text{ or } x = \frac{-2-\sqrt{20}}{2}$$

$\Rightarrow$ *simplify roots*

$$x = -1 + \sqrt{5} \text{ or } x = -1 - \sqrt{5}$$

Teacher: control solutions

- Teachers want to control how an exercise should be solved
  - Technique used
  - Step size in worked-out solutions
- Example: enable or disable "completing the square"

$$x^2 + 4x - 4 = 0$$

$\Rightarrow$ *complete square (lhs)*

$$x^2 + 4x + 4 = 8$$

$\Rightarrow$ *take square (lhs)*

$$(x + 2)^2 = 8$$

$\Rightarrow$ *square root (both sides)*

$$x + 2 = \sqrt{8} \;\; \textit{or} \;\; x + 2 = -\sqrt{8}$$

$\Rightarrow$ *simply roots*

$$x = -2 + 2\sqrt{2} \;\; \textit{or} \;\; x = -2 - 2\sqrt{2}$$

Teacher: control solutions (continued) §2

- Use distributivity rule on both sides (in a single step)

$$-3(x-2) = 3(x+4) - 7$$
$$\Rightarrow \textit{distributivity}$$
$$-3x + 6 = 3x + 12 - 7$$

- Use different number system
  - $\frac{7}{2}$ versus mixed number $3\frac{1}{2}$
  - Complex numbers with existing rewrite strategy
- Approximate as a final step

$$\ldots$$
$$x = -2 + 2\sqrt{2} \;\; \textit{or} \;\; x = -2 - 2\sqrt{2}$$
$$\Rightarrow \textit{approximate}$$
$$x \approx 0.828 \;\; \textit{or} \;\; x \approx -4.828$$

Customization for learning environment

- Create new exercises by combining existing parts
  - Example: solve linear system using an augmented matrix
  - Example: solve an inequality by turning it into an equation
  - Apply a set of rules exhaustively
- Integration with other components
  - Customize level of detail in exercise according to information from the student model
  - Update student model with domain reasoner's diagnosis

Concepts in our domain reasoners

1. Rewrite rules
   - Specify how terms can be manipulated
   - Can represent common misconceptions (a.k.a. buggy rules)
2. Rewrite strategies
   - Guides the process of applying rewrite rules
   - Defined in a strategy language, which is similar to tactic languages (theorem proving) and parser combinator libraries
3. Views and canonical forms
   - For recognizing forms and defining notational conventions
   - Composable into compound views
   - Missing link between rules and strategies
   - Examples: $ax^2 + bx + c = 0$; $3\frac{1}{2}$; $e_1 + e_2 + \ldots + e_n$

Instances of these concepts are grouped together in an exercise

Representation of knowledge

- All three concepts also correspond to mathematical knowledge appearing in textbooks
- A representation is needed for each concept
  - For communicating the internal structure
  - For interpreting specifications and customizations passed to the domain reasoner

Representation of knowledge

- All three concepts also correspond to mathematical knowledge appearing in textbooks
- A representation is needed for each concept
  - For communicating the internal structure
  - For interpreting specifications and customizations passed to the domain reasoner

Trade-offs in making exercise parts transparent:

- Restricts how parts are specified
- Hard to guarantee correctness, or to prevent excessive computations
- Can negatively affect performance

Representing rewrite rules

- Most rewrite rules can be specified by means of a left and right-hand side
- Also buggy rules can be specified this way
- Maps well onto OpenMath's Formal Mathematical Properties (FMP)

$$\textsc{SquareSides}:\quad a^2 = b^2 \rightsquigarrow a = b \text{ or } a = -b$$


```
<FMP><OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
cdbase="http://www.openmath.org/cd"><OMBIND><OMS cd="quant1"
name="forall"/><OMBVAR><OMV name="$0"/><OMV
name="$1"/></OMBVAR><OMA><OMS cd="relation1" name="eq"/><OMA><OMS
cd="relation1" name="eq"/><OMA><OMS cd="arith1" name="power"/><OMV
name="$0"/><OMI>2</OMI></OMA><OMA><OMS cd="arith1" name="power"/><OMV
name="$1"/><OMI>2</OMI></OMA></OMA><OMA><OMS cd="logic1"
name="or"/><OMA><OMS cd="relation1" name="eq"/><OMV name="$0"/><OMV
name="$1"/></OMA><OMA><OMS cd="relation1" name="eq"/><OMV
name="$0"/><OMA><OMS cd="arith1" name="unary_minus"/><OMV
name="$1"/></OMA></OMA></OMA></OMA></OMBIND></OMOBJ></FMP>
```

Representing rewrite strategies

- Strategies are specified using a small set of combinators
- Combinator approach allows for an almost literal translation of strategy definitions
- Existing rules and substrategies can also be referenced by name

```
lineq   = label "linear equation" (prepare <*> basic)

prepare = label "prepare equation"
            (repeat (merge <|> distribute <|> removeDivision))

basic   = label "basic equation"
            (try varToLeft <*> try conToRight <*> try scaleToOne)
```

Rewrite strategies in XML


```xml
<label name="linear equation">
  <sequence>
    <label name="prepare equation">
      <repeat>
        <choice>
          <rule name="merge"/>
          <rule name="distribute"/>
          <rule name="removeDivision"/>
        </choice>
      </repeat>
    </label>
    <label name="basic equation">
      <sequence>
        <orelse>
          <rule name="varToLeft"/>
          <succeed/>
        </orelse>
        <orelse>
          <rule name="conToRight"/>
          <succeed/>
        </orelse>
        <orelse>
          <rule name="scaleToOne"/>
          <succeed/>
        </orelse>
      </sequence>
    </label>
  </sequence>
</label>
```

Rewrite strategies in XML


```
<label name="linear equation">
  <sequence>
    <label name="prepare equation">
      <repeat>
        <choice>
          <rule name="merge"/>
          <rule name="distribute"/>
          <rule name="removeDivision"/>
        </choice>
      </repeat>
    </label>
    <label name="basic equation">
      <sequence>
        <orelse>
          <rule name="varToLeft"/>
          <succeed/>
        </orelse>
        <orelse>
          <rule name="conToRight"/>
          <succeed/>
        </orelse>
        <orelse>
          <rule name="scaleToOne"/>
          <succeed/>
        </orelse>
      </sequence>
    </label>
  </sequence>
</label>
```

- Substrategies can be referenced by name

```
<label name="linear equation">
  <sequence>
    <strategy name="prepare equation"/>
    <strategy name="basic equation"/>
  </sequence>
</label>
```

Configuring rewrite strategies

- Use transformations to adapt an existing strategy
- Can be mixed freely with strategy combinators

Transformations:

- remove part of a strategy (no longer used)
- collapse a substrategy into a rule (single step)
- hide a part in derivation (implicit steps)

- Inverse operations: reinsert, expand, and reveal
- These transformations address several of the case studies

```
<collapse target="basic equation">
  <strategy name="linear equation"/>
</collapse>
```

More strategy transformations

More examples of convenient strategy configurations:

- A certain rule or substrategy must be used
  - Example: using the technique of completing the square is mandatory
- It is preferred to use a particular rule
  - Same set of exercises can be solved
  - Example: try to factor polynomial before applying the quadratic formula
- Replace part of the strategy by another part

Representing views and canonical forms

- Views are more difficult to represent: in general, a view is just a pair of functions
- Possibilities:
  - Define view as a confluent set of rewrite rules
  - Define view as a rewrite strategy
- Arrow combinators (for combining views) and application of higher-order views are represented like the strategy combinators

Motivation: to substitute views in exercises for adapting the mathematical domain reasoner.

Conclusions

- Ability to adapt mathematical domain reasoners is very desirable for learning environments, teachers, and learners
- Explicit representation is needed for all concepts that make up an exercise
- These representation can be communicated, but also interpreted
- Strategy transformations are convenient for configuring existing strategies

Implementation and project webpage at http://ideas.cs.uu.nl/