

Software technology for learning and teaching

Part 1: Introduction

Bastiaan Heeren¹ and Johan Jeuring^{1,2}

¹ Open University of the Netherlands

² Utrecht University

26 January 2015, IPA course, Eindhoven

DU
MathMatch Practice Session - No credit awarded **MathMatch DU**

View Stats View Details X Quit

Question 9: Score 0/1

Vereenvoudig zoveel mogelijk: $\frac{3r^2 - 4s^2}{3r + 2s}$ **X** INCORRECT

Your Answer: $\frac{9r - 4s}{5}$

Comment: $\frac{9r^2 - 4s^2}{3r + 2s} = \frac{(3r)^2 - (2s)^2}{3r + 2s} = \frac{(3r - 2s)(3r + 2s)}{3r + 2s} = 3r - 2s$



1.1 De eenheidskracht

In een driehoekje worden aan de hoeken 3 eenheidskrachten toegevoerd. Het aantal eenheidskrachten dat nodig is om de driehoek in tweeën te splitsen wordt gegeven door de formule $\frac{3n^2 - 3n + 1}{2}$, waarbij n het aantal eenheidskrachten aan de hoeken is.

Wat is de waarde van n als er 10 eenheidskrachten nodig zijn?

Antwoord: $n = 4$

Uw antwoord is correct!

Wass delta

The Wass delta is important for the water supply in Amsterdam. It consists of six sluices, one in each of the six arms of the delta. The sluices are used to regulate the water level in the arms of the delta, to prevent flooding and to ensure a sufficient water supply to the city.

Legend:

- Hoeding
- Industrie
- Recreatie
- Overstroomgebied
- Agriculture
- Natuur
- Stad
- Water
- Wijk
- Infrastructuur



The screenshot shows a web browser window displaying the Khan Academy interface. The browser's address bar shows the URL <https://www.khanacademy.org/mission/math/task/6099883449122816>. The page title is "Understanding the process for solving quadratic equations | Dashboard | The World of Math | Khan Academy".

The interface includes a navigation bar with "LEARN" and "COACH" tabs, a search bar, and a user profile for "JOHAN T. JEURING". The main content area is titled "The World of Math" and displays the current mission: "Understanding the process for solving quadratic equations". A progress indicator shows "Get 5 correct in a row" with five blue checkmarks.

The task instruction is: "Create a list of steps, in order, that will solve the following equation." The equation is $5(x - 3)^2 + 4 = 129$.

Available solution steps are:

- Add 3 to both sides
- Add 4 to both sides
- Divide both sides by 5
- Subtract 3 from both sides
- Subtract 4 from both sides
- Square both sides
- Take the square root of both sides

The "Solution steps:" area contains:

- Subtract 4 from both sides
- Divide both sides by 5

On the right side, there is an "Answer" section with a "Check Answer" button, a "Show me how" section with a "I'd like another hint (3 hints left)" button, and a "Stuck? Watch a video." section with a video player titled "Order of steps exercise exam".



Quality of feedback?

§1

`http://studio.code.org/hoc/2`



The screenshot shows a web browser window with the URL `http://studio.code.org/hoc/2`. The browser's address bar shows `learn.code.org/hoc/2`. The page content includes a grid-based maze game with a red bird character. A white error message box is overlaid on the game, featuring a red bird icon and the text: "You are using all of the necessary types of blocks but not in the right way." Below the message is a green "Try again" button. The background shows a grid of green blocks and a "Reset" button.





- ▶ Simplified tasks
- ▶ Bad feedback
- ▶ No feedback



Use

- ▶ languages and grammars
- ▶ algebra's

To

- ▶ determine what a student has done
- ▶ determine what a student should do
- ▶ explain instead of show why a student performs badly



$x^2 + 20 = 9x$

$x^2 - 9x + 20 = 0$

✗ $(x-5)(x+4) = 0$

Tip:
drieterm ontbinden

$x^2 - 9x + 20 = 0$
wordt dan:
 $(x-4)(x-5) = 0$



1. Introduction
2. Procedural skills
3. Strategy specification language
4. Feedback services
5. Application domains
 - Logic
 - Mathematics
 - Serious games
 - Programming
6. Concluding remarks



1. Introduction
2. Procedural skills
3. Strategy specification language
4. Feedback services
5. Application domains
 - Logic
 - Mathematics
 - Serious games
 - Programming
6. Concluding remarks



In many subjects students have to acquire procedural skills:

- ▶ **Mathematics:** find the derivative of a function
- ▶ **Linear Algebra:** solve a system of linear equations
- ▶ **Logic:** rewrite a proposition to disjunctive normal form
- ▶ **Computer Science:** construct a program from a specification using Dijkstra's calculus
- ▶ **Physics:** calculate the resistance of a circuit
- ▶ **Biology:** calculate inheritance values using Mendel's laws
- ▶ ...



Theorie B

Het oplossen van kwadratische vergelijkingen

Om de vergelijking $x^2 - 7x - 18 = 0$ op te lossen, ontbind je eerst het linkerlid in factoren.

Vervolgens pas je toe $A \cdot B = 0$ geeft $A = 0 \vee B = 0$.

Je krijgt

$$x^2 - 7x - 18 = 0$$

$$(x - 9)(x + 2) = 0$$

$$x - 9 = 0 \vee x + 2 = 0$$

$$x = 9 \vee x = -2$$

Ontbind in factoren.

Pas toe $A \cdot B = 0$ geeft $A = 0 \vee B = 0$.

Het teken \vee betekent of.

Bij het oplossen van een kwadratische vergelijking gebruik je het volgende werkschema.

Werkschema: zo los je een kwadratische vergelijking op

- 1 Maak het rechterlid nul.
- 2 Ontbind het linkerlid in factoren.
- 3 Gebruik: uit $A \cdot B = 0$ volgt $A = 0 \vee B = 0$.



- ▶ Typical **features** of these tools:
 - Generate exercises
 - Stepwise construction of a solution
 - Select rewriting rule or transformation
 - Suggest how to continue
 - Check correctness of a step/solution

- ▶ Such tools offer many **advantages** to users:
 - User can work at any time
 - User can select material and exercises
 - Tool can select exercises based on a user-profile
 - Mistakes can be logged, and reported back to teachers
 - Tool can give immediate feedback



- ▶ Tutoring systems
- ▶ Serious games



1. Introduction
2. Procedural skills
3. Strategy specification language
4. Feedback services
5. Application domains
 - Logic
 - Mathematics
 - Serious games
 - Programming
6. Concluding remarks



<http://ideas.cs.uu.nl/logex/>

ideas - LogEX NL EN Help Logout

Convert to disjunctive normal form Convert to conjunctive normal form Proof logical equivalence

New exercise Rule Justification Correction per step

$\neg(q \wedge p) \vee \neg p$

$\Leftrightarrow \neg(q \wedge p) \wedge \neg\neg p$ De Morgan ×

$\Leftrightarrow \neg(q \wedge p) \wedge p$ Double negation ×

$\Leftrightarrow \neg(q \wedge p) \wedge p$ Rule..



- ▶ Rewrite rules for logical propositions:

$$\begin{array}{ll} \neg\neg\phi \Rightarrow \phi & \phi \wedge (\psi \vee \chi) \Rightarrow (\phi \wedge \psi) \vee (\phi \wedge \chi) \\ \neg(\phi \wedge \psi) \Rightarrow \neg\phi \vee \neg\psi & (\phi \vee \psi) \wedge \chi \Rightarrow (\phi \wedge \chi) \vee (\psi \wedge \chi) \\ \neg(\phi \vee \psi) \Rightarrow \neg\phi \wedge \neg\psi & \end{array}$$

- ▶ Exercise: bring $\neg(\neg(p \vee q) \wedge r)$ to DNF



- ▶ Rewrite rules for logical propositions:

$$\begin{aligned}\neg\neg\phi &\Rightarrow \phi & \phi \wedge (\psi \vee \chi) &\Rightarrow (\phi \wedge \psi) \vee (\phi \wedge \chi) \\ \neg(\phi \wedge \psi) &\Rightarrow \neg\phi \vee \neg\psi & (\phi \vee \psi) \wedge \chi &\Rightarrow (\phi \wedge \chi) \vee (\psi \wedge \chi) \\ \neg(\phi \vee \psi) &\Rightarrow \neg\phi \wedge \neg\psi\end{aligned}$$

- ▶ Exercise: bring $\neg(\neg(p \vee q) \wedge r)$ to DNF

$$\begin{aligned}&\neg(\neg(p \vee q) \wedge r) \\ \Rightarrow &\neg\neg(p \vee q) \vee \neg r \\ \Rightarrow &p \vee q \vee \neg r\end{aligned}$$



- ▶ Rewrite rules for logical propositions:

$$\begin{aligned}\neg\neg\phi &\Rightarrow \phi & \phi \wedge (\psi \vee \chi) &\Rightarrow (\phi \wedge \psi) \vee (\phi \wedge \chi) \\ \neg(\phi \wedge \psi) &\Rightarrow \neg\phi \vee \neg\psi & (\phi \vee \psi) \wedge \chi &\Rightarrow (\phi \wedge \chi) \vee (\psi \wedge \chi) \\ \neg(\phi \vee \psi) &\Rightarrow \neg\phi \wedge \neg\psi\end{aligned}$$

- ▶ Exercise: bring $\neg(\neg(p \vee q) \wedge r)$ to DNF

$$\begin{aligned}&\neg(\neg(p \vee q) \wedge r) && \neg(\neg(p \vee q) \wedge r) \\ \Rightarrow &\neg\neg(p \vee q) \vee \neg r && \Rightarrow \neg((\neg p \wedge \neg q) \wedge r) \\ \Rightarrow &p \vee q \vee \neg r && \Rightarrow \neg(\neg p \wedge \neg q) \vee \neg r \\ &&& \Rightarrow \neg\neg p \vee \neg\neg q \vee \neg r \\ &&& \Rightarrow p \vee \neg\neg q \vee \neg r \\ &&& \Rightarrow p \vee q \vee \neg r\end{aligned}$$



- ▶ Naive strategy:

Apply rewrite rules exhaustively



▶ Naive strategy:

Apply rewrite rules exhaustively

▶ Algorithmic strategy:

- (1) Remove constants*
- (2) Unfold definitions of implication/equivalence*
- (3) Push negations inside (top-down)*
- (4) Then use the distribution rule*



▶ Naive strategy:

Apply rewrite rules exhaustively

▶ Algorithmic strategy:

- (1) *Remove constants*
- (2) *Unfold definitions of implication/equivalence*
- (3) *Push negations inside (top-down)*
- (4) *Then use the distribution rule*

▶ Expert strategy:

Apply the algorithmic strategy, but use rules for tautologies and contradictions whenever possible



To model intelligence in a computer program, Bundy (*The Computer Modelling of Mathematical Reasoning*, 1983) identifies three important, basic needs:

1. The need to have knowledge about the domain
2. The need to reason with that knowledge
3. The need for knowledge about how to direct or guide that reasoning



To model intelligence in a computer program, Bundy (*The Computer Modelling of Mathematical Reasoning*, 1983) identifies three important, basic needs:

1. The need to have knowledge about the domain
2. The need to reason with that knowledge
3. The need for knowledge about how to direct or guide that reasoning

In our running example:

1. The domain consists of **logical propositions**
2. Reasoning uses **rewrite rules** for logical propositions
3. **Strategies** guide that reasoning



We need the following concepts for specifying a strategy:

- ▶ apply a basic rewrite rule (*" \wedge distributes over \vee "*)
- ▶ sequence (*"first ... then ..."*)
- ▶ choice (*"use one of the rules for \neg "*)
- ▶ apply exhaustively (*"repeat ... as long as possible"*)
- ▶ traversals (*"apply ... top down"*)

The same concepts are found in:

- ▶ (program) transformation languages
- ▶ proof plans and tacticals
- ▶ workflow languages



► Basic strategy combinators:

1. Sequence $s \langle \star \rangle t$
2. Choice $s \langle \triangleright \rangle t$
3. Unit elements *succeed, fail*
4. Labels *label* $l s$
5. Recursion *fix* f



- ▶ Basic strategy combinators:

- | | |
|------------------|--------------------------------------|
| 1. Sequence | $s \langle \star \rangle t$ |
| 2. Choice | $s \langle \triangleright \rangle t$ |
| 3. Unit elements | <i>succeed, fail</i> |
| 4. Labels | <i>label</i> $l s$ |
| 5. Recursion | <i>fix</i> f |

- ▶ Many more combinators can be added:

option $s = s \langle \triangleright \rangle \textit{succeed}$

many $s = \textit{fix} (\lambda x \rightarrow \textit{option} (s \langle \star \rangle x))$

repeat $s = \textit{many} s \langle \star \rangle \textit{not} s$



1. Introduction
2. Procedural skills
3. Strategy specification language
4. Feedback services
5. Application domains
 - Logic
 - Mathematics
 - Serious games
 - Programming
6. Concluding remarks



With a strategy, we can calculate several kinds of feedback:

- ▶ Feedback after a step by a user
 - ▶ Hints on how to continue
 - ▶ Worked-out solutions
 - ▶ Strategy unfolding (problem decomposition)
 - ▶ Completion problems
 - ▶ Progress (number of steps remaining)
 - ▶ Report common mistakes
-
- ▶ Most categories appear in the **tutoring principles of Anderson**
 - ▶ Offered as (web-)services to other **learning environments**



- ▶ Formulate misconceptions as **buggy rules**:

$$\neg(\phi \wedge \psi) \not\Rightarrow \neg\phi \wedge \neg\psi$$
$$\phi \wedge (\psi \vee \chi) \not\Rightarrow (\phi \wedge \psi) \vee \chi$$

- ▶ Buggy rules can be recognized and reported with a specialized feedback text
- ▶ Also: **buggy strategies** to describe procedural mistakes



- ▶ Strategies have a **hierarchical structure**
- ▶ Use structure to decompose an exercise
 - First ask for the final answer
 - If the answer is incorrect, decompose the problem into subparts and let the user try again
 - Example from linear algebra: split the Gaussian Elimination method into a forward and a backward pass
- ▶ The structure of a strategy and its labels also provide a way to **adapt** and **customize** the strategy



The main idea:

- ▶ A strategy describes valid sequences of rules
- ▶ View a strategy specification as a **context-free grammar**
- ▶ This turns tracking intermediate steps into a **parsing problem**



The main idea:

- ▶ A strategy describes valid sequences of rules
- ▶ View a strategy specification as a **context-free grammar**
- ▶ This turns tracking intermediate steps into a **parsing problem**

Feedback service	Parsing problem
ready	is the empty sentence (ϵ) accepted?
provide hint	compute the “first set”
worked-out solution	construct a sentence
after a step	try to recognize the rewrite rule that was used, and parse this rule as the next symbol of the input



1. Introduction
2. Procedural skills
3. Strategy specification language
4. Feedback services
5. **Application domains**
 - Logic
 - Mathematics
 - Serious games
 - Programming
6. Concluding remarks



- ▶ Logic
- ▶ Mathematics
- ▶ Communication skills
- ▶ Infection and Immunology
- ▶ Programming



- ▶ Use strategies to prove the equivalence of logical propositions
- ▶ Allow student to make **forward steps** and **backward steps**
- ▶ Joint work with Josje Lodder

$$\begin{aligned} & \neg ((p \rightarrow q) \rightarrow (p \wedge q)) \\ & \Leftrightarrow \{\text{implication elimination}\} \\ & \neg (\neg (p \rightarrow q) \vee (p \wedge q)) \\ & \Leftrightarrow \{\text{De Morgan}\} \\ & \neg \neg (p \rightarrow q) \wedge \neg (p \wedge q) \\ & \Leftrightarrow \{\text{double negation}\} \\ & (p \rightarrow q) \wedge \neg (p \wedge q) \\ & \Leftrightarrow \{\text{De Morgan}\} \\ & (p \rightarrow q) \wedge (\neg p \vee \neg q) \end{aligned}$$



- ▶ The strategy rewrites a pair of propositions
- ▶ Rewrite both parts to disjunctive normal form, and then towards equal forms
- ▶ Two simple techniques simplify the generated proofs:
 - Try to **decompose the proof** into subproofs by inspecting the top-level operators
 - Search for **common subformulas**

$$\neg (\boxed{p \rightarrow q} \rightarrow (p \wedge q))$$

$$\Leftrightarrow \{ \dots \}$$

$$\boxed{p \rightarrow q} \wedge (\neg p \vee \neg q)$$



- ▶ We collaborate with the **Freudenthal Institute** to extend their applets with our feedback facilities
 - Covers most topics in secondary school mathematics: polynomial equations, inequalities, calculating with powers, derivatives, etc.
 - Applets are used by many schools (and a popular textbook)
- ▶ We participated in the **Math-Bridge** project
 - Large European consortium around the ActiveMath learning environment
 - Aims at providing a math bridging course to higher education
- ▶ We try to apply our approach to different types of exercises



The screenshot shows a web browser window titled "DWO Math Environment - Mozilla Firefox". The address bar shows "DWO Math Environment" and the page title is "Digitale Wiskunde Omgeving Freudenthal Instituut". The page content is titled "4. quadreq 3" and includes a "Inloggen" button. The main area displays the instruction "Los de vergelijking op." followed by the equation $x(2x - 4) = 0$. The solution steps are shown as follows:

- $x = 0$ of $2x - 4 = 0$
- $x = 0$ of $2x = 4$
- $x = 0$ of $x = 2$

Feedback messages on the right side of the solution steps are:

- de factoren op 0 stellen
- constante termen naar rechts brengen
- variabele vrijmaken door beide kanten te delen
- correct opgelost

At the bottom, a progress bar shows 10 tasks, with the 4th task completed. The score is 10 out of 10.

Tool by Peter Boon (Freudenthal Institute)



- ▶ Support for canonical forms
 - To test for equality
 - To control the granularity of steps
 - To simplify terms

Examples:

- $2\sqrt{2}$ versus $\sqrt{8}$, $3\frac{1}{2}$ versus $\frac{7}{2}$ (or even 3.5)
- $x + (-3)$ versus $x - 3$
- pattern $ax + b$ versus $3 - 5x$

- ▶ Flexibility in strategies (customization)
- ▶ Parameterized rewrite steps (“divide both sides by 5”)



What does a step look like?

§5.2

$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!



What does a step look like?

§5.2

$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$3 * (4 * x - 1) + 3$$



What does a step look like?

§5.2

$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \end{aligned}$$



$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3 * 1) + 3 \end{aligned}$$



$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3) + 3 \end{aligned}$$



$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3) + 3 \\ \Rightarrow & (12 * x + (-3)) + 3 \end{aligned}$$



What does a step look like?

$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3) + 3 \\ \Rightarrow & (12 * x + (-3)) + 3 \\ \Rightarrow & 12 * x + (-3 + 3) \end{aligned}$$



What does a step look like?

$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3) + 3 \\ \Rightarrow & (12 * x + (-3)) + 3 \\ \Rightarrow & 12 * x + (-3 + 3) \\ \Rightarrow & 12 * x + 0 \end{aligned}$$



$$3 * (4 * x - 1) + 3 = 7 * x - 14 \Rightarrow 12 * x = 7 * x - 14?$$

You are doing a lot in this step!

$$\begin{aligned} & 3 * (4 * x - 1) + 3 \\ \Rightarrow & (3 * 4 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3 * 1) + 3 \\ \Rightarrow & (12 * x - 3) + 3 \\ \Rightarrow & (12 * x + (-3)) + 3 \\ \Rightarrow & 12 * x + (-3 + 3) \\ \Rightarrow & 12 * x + 0 \\ \Rightarrow & 12 * x \end{aligned}$$



- ▶ **Economy of rules:** I want to describe

$$a * (b + c) \Rightarrow a * b + a * c$$

but preferably not also:

$$a * (b - c) \Rightarrow a * b - a * c$$

$$-a * (b + c) \Rightarrow -a * b - a * c$$

- ▶ **Canonical forms:** $a + (-b)$ should be presented as $a - b$
- ▶ **Granularity:** users at different levels need different granularity of rules
- ▶ **Recognizing user steps:** when showing steps to users, we want to apply some simplifications automatically. When recognising steps, however, such simplifications are not obligatory



A **view** views an expression in a particular format:

- ▶ a **match** function returns an equivalent value in a different format, for example:

$$\begin{aligned} \text{match plusView } (a - b) &\Rightarrow a + (-b) \\ \text{match plusView } (-(a + b)) &\Rightarrow -a + -b \end{aligned}$$

- ▶ a **build** function to return to the original domain, for example:

$$\begin{aligned} &3 * (4 * x - 1) \\ \Rightarrow &\{ \text{match plusView on } 4 * x - 1 \} \\ &3 * (4 * x + (-1)) \\ \Rightarrow &\{ \text{distribute * over +} \} \\ &3 * 4 * x + 3 * (-1) \\ \Rightarrow &\{ \text{simplify using rationalView} \} \\ &12 * x - 3 \end{aligned}$$



- ▶ Many rules use one or more views for matching on the left-hand side
- ▶ Many rules use one or more views to clean up a result expression after rewriting
- ▶ Views and parametrized rules solve the problem of making all steps in solving an exercise explicit



Communicate!

science-vs75.science.uu.nl/backend/index.php/scenario/index/scenarios.18

Scenarios Profiel Groepen Scriptbeheer Gebruikersbeheer Uitloggen

COMMUNICATE

Ja, helemaal.

▲ Geschiedenis

Toestemming vragen om advies te geven reflectie geven doorvragen om advies te kunnen geven voor

1. Zal ik u een advies geven wat u het beste kan doen?



Editor - Communicatel

science-vs75.science.uu.nl/backend/index.php/editor/18

Buienradar YouTube Wikipedia NS StatCounter Facebook dub Johan Jeuring Google+ maps.google.com

Scriptbeheer Kenmerken Klادblad Opslaan Media Speler Patiënt Gesprek Onderwerp

Intenties Ouders Ordenen Valideer Handleiding Scenario: baliegesprek demo

[-] Naam onderwerp

Klادblok Validatierapport

Nieuw item Alles naar nodes Alles verwijderen

S/P	Zin	Intentie	Emotie	Feedback
.....	S		(geen)	

Emoties
boosheid delta

Parameters
contact delta -1

Intenties
reflectie geven

Media

Visueel: Geen
Audio: Geen

Feedback

Lij je samenvatting bleek al dat je mevrouw begrepen had. De reflectie is daardoor overbodig.

Commentaar

Einde gesprek

Eindknoop



Pause

5 Scenario: 1

Actions

- A: Opsonisation
- B: Chemotaxis
- C: Lysis
- D: Neutrophil
- E: Macrophage
- F: Monocyte
- G: Cytokine
- H: Pre-existing antibodies
- I: Produced antibodies
- J: B-cell
- K: T-Helper

{(H)C | [A,B,H]D }



We have developed programming tutors for

- ▶ Evaluating functional expressions
- ▶ Learning functional programming
- ▶ Learning imperative programming

More about this in the last lecture.



1. Introduction
2. Procedural skills
3. Strategy specification language
4. Feedback services
5. Application domains
 - Logic
 - Mathematics
 - Serious games
 - Programming
6. Concluding remarks



- ▶ 10:00 - 11:00 Lecture 1: Introduction & general overview (Johan Jeuring)
- ▶ 11:00 - 11:15 Coffee
- ▶ 11:15 - 12:30 Lecture 2: Rewriting & strategies (Bastiaan Heeren)
- ▶ 12:30 - 13:30 Lunch
- ▶ 13:30 - 14:45 Lab (Bastiaan Heeren and Johan Jeuring)
- ▶ 14:45 - 15:00 Tea/coffee
- ▶ 15:00 - 16:00 Lecture 3: Programming tutors (Johan Jeuring)



- ▶ We introduced a strategy language to make the procedure for solving an exercise explicit
- ▶ This language is what differentiates us from other tools
- ▶ Feedback is calculated from the strategy by turning feedback services into parsing problems
- ▶ Strategies can be used in many learning tools



Bastiaan Heeren and Johan Jeuring. Feedback services for stepwise exercises. Science of Computer Programming Special Issue on Software Development Concerns in the e-Learning Domain, volume 88, 110 - 129, 2014.

Bastiaan Heeren, Johan Jeuring, and Alex Gerdes. Specifying rewrite strategies for interactive exercises. In Mathematics in Computer Science 3(3), 349 - 370, 2010.

- ▶ Accessible via <http://www.jeuring.net/homepage/Publications/index.html>
- ▶ Project webpage at <http://ideas.cs.uu.nl/>
- ▶ For more information, contact us at bhr@ou.nl, J.T.Jeuring@uu.nl

