



Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Ideas

Part 1: Procedural skills

Johan Jeuring

Utrecht University

Monday, September 7, 2016

1. Introduction



DU MathMatch Practice Session - No credit awarded MathMatch DU

Question 9: Score 0/1

Vereenvoudig zoveel mogelijk: $\frac{9r^2 - 4s^2}{3r + 2s}$ **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 techniekje wordt het de fase of fase-geslacht... **Deeluit 2**

1. Bepaal $\sin(2\pi)$ en $\cos(2\pi)$.
 2. Bepaal $\sin(\pi)$ en $\cos(\pi)$.
 3. Bepaal $\sin(\frac{\pi}{2})$ en $\cos(\frac{\pi}{2})$.
 4. Bepaal $\sin(\frac{3\pi}{2})$ en $\cos(\frac{3\pi}{2})$.

Unit circle diagram showing sine and cosine values for various angles.

Waa's delta

The Waa's delta is a map of the Waa's delta area in the Netherlands. It is a map of the Waa's delta area in the Netherlands. It is a map of the Waa's delta area in the Netherlands.

Legend: Housing, Industry, Recreation, Greenhouses, Agriculture, Nature, Other, Water, Infrastructure.



Free input?

§1

The screenshot shows a web browser window with 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 user is logged in as "JOHAN T. JEURING".

The main content area is titled "The World of Math" and shows a progress bar for the mission "Understanding the process for solving quadratic equations". The progress bar indicates that 5 out of 5 steps are completed, with a "Get 5 correct in a row" notification.

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

Below the equation are two columns of buttons representing possible steps:

- Column 1: 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.
- Column 2 (Solution steps): Subtract 4 from both sides, Divide both sides by 5.

On the right side, there is an "Answer" section with a "Check Answer" button and a "Show me how" section with a "I'd like another hint (3 hints left)" button. Below that is a "Stuck? Watch a video." section with a video player showing a math problem.



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 page title is "Code.org - The Maze #2". The browser's address bar shows `learn.code.org/hoc/2`. The page content includes a grid-based maze with a red bird character and a pig character. A feedback message box is overlaid on the grid, 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 dark interface with various code blocks like "turn left", "turn right", "move forward", and "reset".





- ▶ 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:
 $(x-4)(x-5) = 0$



Outline of presentation

§1

Introduction

Procedural skills

Strategy specification language

Feedback services

Application domains

Logic

Mathematics

Serious games

Programming

Lab assignment

Concluding remarks



Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

2. Procedural skills



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



Do they work?

§2

- ▶ Tutoring systems
- ▶ Serious games



3. Strategy specification language



<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

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

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



- ▶ 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



- ▶ Strategies can be used for any kind of procedural activities (not just maths)
- ▶ Alternatives: ACT-R (next week), CTAT (Cognitive Tutor Authoring Tools), Andes, many more
- ▶ Strategies are a declarative and compositional alternative

Our running example in ACT-R

1. The domain consists of **logical propositions**
2. Reasoning uses **production rules** for logical propositions
3. Reasoning is implemented by an interpreter which chooses which productions to fire. There is no explicit representation of complex cognitive skills



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 *succeed, fail*
4. Labels *label ℓ s*
5. Recursion *fix 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 $s = \textit{many} s \langle \star \rangle \textit{not} s$



4. Feedback services



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**:

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

- ▶ 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



5. Application domains



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



5.1 Logic



- ▶ Use strategies to prove the equivalence of logical propositions
- ▶ Allow student to make **forward steps** and **backward steps**

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



- ▶ 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**

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

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

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



5.2 Mathematics



- ▶ 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



DWO Math Environment - Mozilla Firefox

Gestand Bewerken Beeld Geschiedenis Blgdwijzers Extra Help

DWO Math Enviroment

Digitale Wiskunde Omgeving Freudenthal Instituut

>> B: Examples quadreq Niet ingelogd

4. quadreq 3

Inloggen

Los de vergelijking op.

$\sqrt{\quad}$ \square \square^2 $\frac{\square}{\square}$ \square \square meer tip solve \downarrow \uparrow

$x(2x - 4) = 0$

$x = 0$ 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



- ▶ 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!

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



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!

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



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!

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



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!

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



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

You are doing a lot in this step!

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



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

You are doing a lot in this step!

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



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

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:

$$\begin{aligned} a * (b - c) &\Rightarrow a * b - a * c \\ -a * (b + c) &\Rightarrow -a * b - a * c \end{aligned}$$

- ▶ **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:

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



- ▶ 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



5.3 Serious games



A communication skills game

§5.3

Communicate!

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

Scenarios Profiel Groepen Scriptbeheer Gebruikersbeheer Uitloggen

COMMUNICATE

Ja, helemaal.

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

Dus u merkt dat uw enkels minder dik zijn. Over het algemeen is het geen probleem om de furonamide in te nemen, u onthoudt het door het gelijk met het ontbijt in te nemen. Alleen op zaterdag is het onhandig met hardlopen. Dus u vraagt zich af wat u nu het beste kan doen. Klopt dat?

Ja, helemaal.

Kunt u op een ander moment ook hardlopen?

Het is heel vervelend voor u met hardlopen.

Ik loop ook op andere dagen, maar niet met mijn loopgroep.

Ja, dat zei ik al.

Einde gesprek

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
Lijkt je samenvatting bleek af dat je mevrouw begrepen had. De reflectie is daardoor overbodig.

Commentaar

Einde gesprek

Eindknoop



Pause [Progress Bar] 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 }



5.4 Programming



We have developed programming tutors for

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

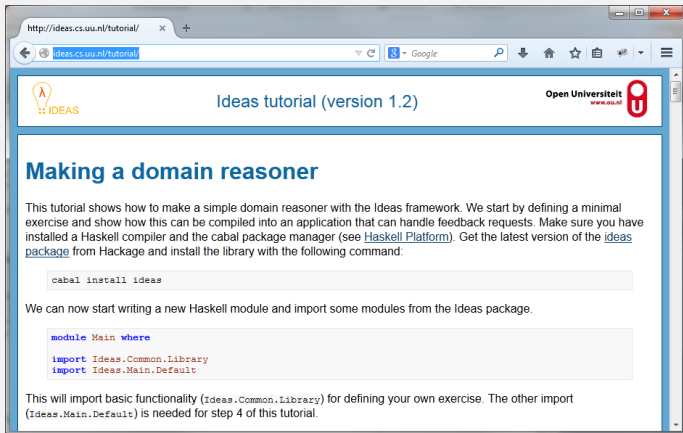
More about this in the third lecture about ideas.



6. Lab assignment



Visit <http://ideas.cs.uu.nl/tutorial/>

A screenshot of a web browser displaying the 'Ideas tutorial (version 1.2)' page. The browser's address bar shows 'http://ideas.cs.uu.nl/tutorial/'. The page header includes the 'IDEAS' logo (a lightbulb with a lambda symbol), the title 'Ideas tutorial (version 1.2)', and the 'Open Universiteit' logo with the URL 'www.uu.nl'. The main content area has a blue heading 'Making a domain reasoner'. Below the heading, a paragraph explains the tutorial's goal: to create a simple domain reasoner using the Ideas framework. It instructs the user to install a Haskell compiler and the cabal package manager, and to get the latest version of the 'ideas' package from Hackage. A code block shows the command 'cabal install ideas'. Another paragraph states that the user can now start writing a new Haskell module and import some modules from the Ideas package. A second code block shows Haskell code: 'module Main where', 'import Ideas.Common.Library', and 'import Ideas.Main.Default'. A final paragraph explains that the first import provides basic functionality, while the second is needed for a later step in the tutorial.

http://ideas.cs.uu.nl/tutorial/

IDEAS

Ideas tutorial (version 1.2)

Open Universiteit
www.uu.nl

Making a domain reasoner

This tutorial shows how to make a simple domain reasoner with the Ideas framework. We start by defining a minimal exercise and show how this can be compiled into an application that can handle feedback requests. Make sure you have installed a Haskell compiler and the cabal package manager (see [Haskell Platform](#)). Get the latest version of the [ideas package](#) from Hackage and install the library with the following command:

```
cabal install ideas
```

We can now start writing a new Haskell module and import some modules from the Ideas package.

```
module Main where
import Ideas.Common.Library
import Ideas.Main.Default
```

This will import basic functionality (`Ideas.Common.Library`) for defining your own exercise. The other import (`Ideas.Main.Default`) is needed for step 4 of this tutorial.



Start version, see <http://ideas.cs.uu.nl/tutorial/>, has,

- ▶ Simple arithmetic expression language
- ▶ Two evaluation rules

data $Expr = Add\ Expr\ Expr \mid Negate\ Expr \mid Con\ Int$

1. Add multiplication to the expression language (and extend the evaluation strategy)
2. Add distribution rules to the strategy
3. Add support for calculating with fractions (e.g. $\frac{5}{7} + \frac{1}{2}$)
 - Find the least common multiple of the denominators
 - Rewrite top-heavy fractions to mixed fractions (e.g. $1\frac{3}{14}$)



- ▶ Latest release: version 1.5 (May 2016)
- ▶ Over 10,000 lines of Haskell code (in 110 modules)
- ▶ <http://hackage.haskell.org/package/ideas>

How to interact with a domain reasoner?

- ▶ Develop a client that calls the (server/cgi) domain reasoner
- ▶ Use the Haskell interpreter (ghci)
- ▶ Compile to a cgi binary (with support for HTML) and deploy on your localhost; use a browser
- ▶ Compile and send a request from the command-line (file)



The screenshot shows a web browser window with the address bar containing `http://localhost/Tutorial.cgi?input=<`. The page header includes the 'IDEAS' logo, navigation buttons for 'Index', 'Exercises', and 'Services', and the 'Open Universiteit' logo with the URL `www.ou.nl`.

Exercise eval.full

Description
Evaluate an expression (full)

Derivations

- 1.

Add (Con 5) (Negate (Con 2))
 \Rightarrow eval.negate
Add (Con 5) (Con (-2))
 \Rightarrow eval.add
Con 3

Exercise

- information
- strategy
- rules
- examples
- derivations
- test report



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.



- ▶ You can discuss the lab amongst each other, but you cannot reuse code from somebody else
- ▶ Hand in your solution via email to me on or before 28/9



7. Concluding remarks



- ▶ 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

