Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2025, gehouden op donderdag 27 maart 2025 jl. en georganiseerd door Hogeschool Windesheim). Bij elkaar zo'n 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats in 2027 en wordt dan georganiseerd door HAN University of Applied Sciences. Zodra daarover meer informatie beschikbaar is, is deze hier te vinden.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

Universiteit Utrecht

Adding testing to Ask-Elle: An Interactive Functional Programming Tutor

Johan Jeuring

Joint work with Alex Gerdes, Bastiaan Heeren, Jurriën Stutterheim

Computer Science
Utrecht University and Open Universiteit Nederland

NIOC 2013, April 2013

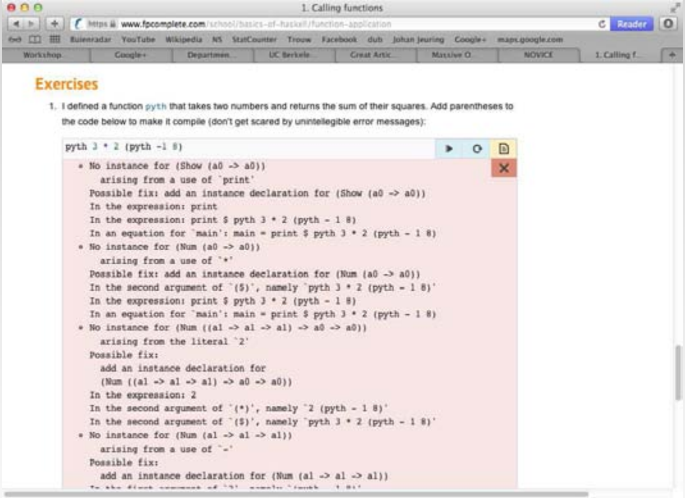
Open Universiteit
www.ou.nl


Codecademy presents
Your friends want to learn how to code too!
Tweet
12.1K
Like
1k
Code Year
56,997 people have decided to learn to code in 2012.
Why not you?
Make your New Year's resolution learning to code.
Sign up on Code Year to get a new interactive programming lesson
sent to you each week and you'll be building apps
and web sites before you know it.

Learning to program

Learning to program is hard.

- Misconceptions about the syntax and semantics of a programming language
- Analysing and creating a model of the problem that can be implemented is difficult
- Decomposing a complex problem into smaller subproblems requires experience
- Most compilers give poor error messages


1. Calling functions
https www.fpcomplete.com/school/basics-of-haskell/function-application
Reader
Buienradar YouTube Wikipedia NS StatCounter Trouw Facebook dub Johan Jeuring Google+ maps.google.com
Workshop Google+ Departmen UC Berkele Great Artic Massive O NOVICE 1. Calling f
Exercises
1. I defined a function pyth that takes two numbers and returns the sum of their squares. Add parentheses to the code below to make it compile (don't get scared by unintelligible error messages):
pyth 3 * 2 (pyth -1 8)
• No instance for (Show (a0 -> a0))
arising from a use of `print'
Possible fix: add an instance declaration for (Show (a0 -> a0))
In the expression: print
In the expression: print $ pyth 3 * 2 (pyth - 1 8)
In an equation for `main': main = print $ pyth 3 * 2 (pyth - 1 8)
• No instance for (Num (a0 -> a0))
arising from a use of `*'
Possible fix: add an instance declaration for (Num (a0 -> a0))
In the second argument of `($)', namely `pyth 3 * 2 (pyth - 1 8)'
In the expression: print $ pyth 3 * 2 (pyth - 1 8)
In an equation for `main': main = print $ pyth 3 * 2 (pyth - 1 8)
• No instance for (Num ((a1 -> a1 -> a1) -> a0 -> a0))
arising from the literal `2'
Possible fix:
add an instance declaration for
(Num ((a1 -> a1 -> a1) -> a0 -> a0))
In the expression: 2
In the second argument of `(*)', namely `2 (pyth - 1 8)'
In the second argument of `($)', namely `pyth 3 * 2 (pyth - 1 8)'
• No instance for (Num (a1 -> a1 -> a1))
arising from a use of `-'
Possible fix:
add an instance declaration for (Num (a1 -> a1 -> a1))

Programming tutors

A programming tutor supports a student when learning how to program:

- giving hints (in varying level of detail)
- showing worked-out solutions
- reporting erroneous steps

Challenges for programming tutors

Programming tutors are not widely used.

- Building a tutor is a substantial amount of work
- Using a tutor in a course is hard for a teacher: adapting or extending a tutor is often very difficult or even impossible
- Having to specify feedback with each new exercise is often a lot of work

Preferably, a programming tutor:

- supports easy specification of exercises
- automatically derives feedback and hints

This talk

Shows Ask-Elle, a programming tutor for Haskell, in action.

- Support developing beginners' Haskell programs
- Add programming exercises
- Adapt feedback
- Prove correctness
- Prove incorrectness

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

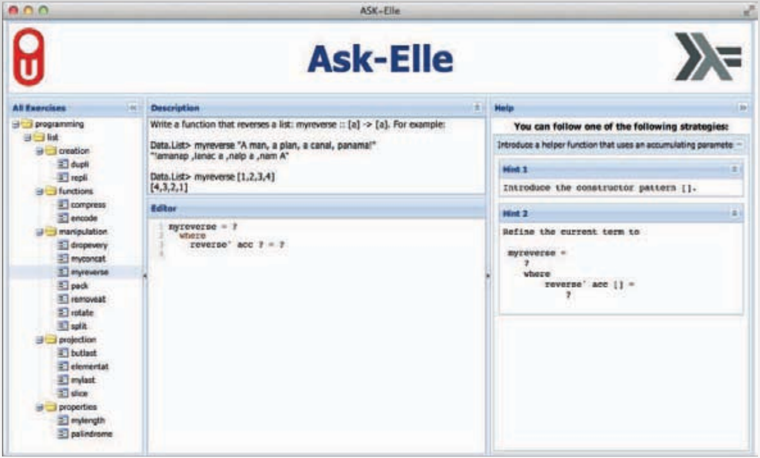
Ask-Elle: A programming tutor for Haskell

We are developing Ask-Elle: a programming tutor for Haskell. Using the tutor, a student can:

- develop her program incrementally
- receive feedback about whether or not she is on the right track
- can ask for a hint when she is stuck
- see how a complete program is stepwise constructed

A teacher specifies an exercise by means of model solutions.

The tutor targets first-year computer science students.



[NIOC 2013: Adding testing to Ask-Elle – An Interactive Functional Programming Tutor]

An example interactive session

Programming task: write a program that reverses a list:

> ?

Tutor response on Hint:

There are several ways you can proceed:

- Introduce a helper function that uses an accumulating parameter.
- Use the *Prelude* function *foldl*.
- Use explicit recursion.

An example interactive session

Programming task: write a program that reverses a list:

$$
\begin{aligned}
&\mathit{reverse} = \mathit{reverse}' \ ? \\
&\quad \textbf{where} \\
&\qquad \mathit{reverse}' \ \mathit{acc} \ ? = \ ?
\end{aligned}
$$

Tutor response on Hint:

Apply $\mathit{reverse}'$ to [], or use pattern matching for the second argument of $\mathit{reverse}'$, or refine the right-hand side of reverse'.

An example interactive session

Programming task: write a program that reverses a list:

$$\begin{aligned}&reverse = reverse'\ [\,]\\&\quad \textbf{where}\\&\qquad reverse'\ acc\ [\,] = ?\end{aligned}$$

Tutor response on Hint:

Refine the right hand side of the empty list case.

An example interactive session

Programming task: write a program that reverses a list:

$$
\begin{aligned}
&reverse = reverse'\ [\,]\\
&\quad \textbf{where}\\
&\qquad reverse'\ acc\ [\,] = [\,]
\end{aligned}
$$

Tutor response on Check:

Unexpected step, which may be incorrect.

An example interactive session

Programming task: write a program that reverses a list:

$$reverse = reverse' \; [\,]$$
$$\quad \textbf{where}$$
$$\qquad reverse' \; acc \; [\,] = acc$$

Tutor response on Hint:

Define the non-empty list case of *reverse'*

An example interactive session

Programming task: write a program that reverses a list:

$$
\begin{aligned}
&\mathit{reverse} = \mathit{reverse}'\ [\,] \\
&\quad \textbf{where} \\
&\qquad \mathit{reverse}'\ \mathit{acc}\ [\,] \qquad\ \ = \mathit{acc} \\
&\qquad \mathit{reverse}'\ \mathit{acc}\ (x : xs) = ?
\end{aligned}
$$

Tutor response on Hint:

Define the recursive call of *reverse′*

An example interactive session

Programming task: write a program that reverses a list:

$$\begin{aligned}&\mathit{reverse} = \mathit{reverse}'\ [\,]\\ &\quad \textbf{where}\\ &\qquad \mathit{reverse}'\ \mathit{acc}\ [\,] \qquad\quad = \mathit{acc}\\ &\qquad \mathit{reverse}'\ \mathit{acc}\ (x : \mathit{xs}) = \mathit{reverse}'\ (y : \mathit{acc})\ ?\end{aligned}$$

Tutor response on Check:

```
Error: Undefined variable y
```

An example interactive session

Programming task: write a program that reverses a list:

$$
\begin{aligned}
&\mathit{reverse} = \mathit{reverse}'\ []\\
&\quad \textbf{where}\\
&\qquad \mathit{reverse}'\ \mathit{acc}\ [] = \mathit{acc}\\
&\qquad \mathit{reverse}'\ \mathit{acc}\ (x : xs) = \mathit{reverse}'\ (x : \mathit{acc})\ xs
\end{aligned}
$$

Tutor response on Check:

You have correctly solved the exercise.

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

Universiteit Utrecht

What kind of feedback?

- Syntax or type error
- Correct step
- Coming soon: violates the following property: ...
- Hint, in increasing detail
- Solved

Model solutions for *reverse*

The tutor derives feedback from model solutions.

$$\begin{aligned} &reverse_1\ [\,] &&= [\,] \\ &reverse_1\ (x : xs) &&= reverse_1\ xs +\!\!+ [x] \end{aligned}$$

$$\begin{aligned} &reverse_2 = reverse_2'\ [\,] \\ &\quad \mathbf{where}\ reverse_2'\ acc\ [\,] &&= acc \\ &\qquad\qquad\ \ reverse_2'\ acc\ (x : xs) &&= reverse_2'\ (x : acc)\ xs \end{aligned}$$

$$reverse_3 = foldl\ (flip\ (:))\ [\,]$$

[NIOC 2013: Adding testing to Ask-Elle – An Interactive Functional Programming Tutor]

Adapting feedback

A teacher fine-tunes feedback by annotating a model solution.

> *reverse* = *foldl* {-# FEEDBACK Note ... #-} (*flip* (:)) []

A teacher disallows or enforces a particular solution by:

> *reverse* = {-# MUSTUSE #-} *foldl* (*flip* (:)) []

Furthermore, we can add a property to a function, and use that to recognize student solutions:

> *reverse* =
> {-# ALT foldl op e == foldr (flip op) e . reverse #-}
> *foldl* (*flip* (:)) []

[NIOC 2013: Adding testing to Ask-Elle – An Interactive Functional Programming Tutor]

Correctness

- Using annotated model solutions we can prove that a student solution is (partially) correct
- Compare (possibly partial) student solution with model solution after normalisations
- We can give hints, and show worked-out solutions
- We cannot say anything about incorrect or different solutions

[NIOC 2013: Adding testing to Ask-Elle – An Interactive Functional Programming Tutor]

Meta information for *reverse*

Besides model solutions, we store meta information about *reverse* in a configuration file:

$$
\begin{aligned}
&function = reverse \\
&type \quad\;\; = [a] \rightarrow [a] \\
&groups \;\; = programming.FP \\
&property = (\lambda xs \rightarrow whenFail \\
&\qquad\qquad\quad \texttt{"reverse does not reverse a list"} \\
&\qquad\qquad\quad (reverse\ xs \equiv reverse_1\ xs) \\
&\qquad\qquad)
\end{aligned}
$$

property is the standard property:

$$program_{student} \equiv program_{model}$$

Testing

- We use QuickCheck to test a property of a function.
- QuickCheck generates random values for which it tests the validity of a property.
- If QuickCheck finds a counterexample, it tries to shrink it to return a counterexample that is as small as possible.

Testing example

For the following erroneous student solution

$$
\begin{aligned}
&\mathit{reverse} = \mathit{reverse}'\ [\,] \\
&\quad \textbf{where}\ \mathit{reverse}'\ \mathit{acc}\ [\,] = [\,] \\
&\qquad\quad\ \mathit{reverse}'\ \mathit{acc}\ (x : \mathit{xs}) = \mathit{reverse}'\ (x : \mathit{acc})\ \mathit{xs}
\end{aligned}
$$

QuickCheck gives:

quickCheck property
Falsifiable, after 3 *tests* :
`"reverse does not reverse a list"`
`"counterexample: "` [1]

More informative properties for testing

$$property = \lambda x \rightarrow prop\_lengthatmost\ x$$
$$.\&\&.\ prop\_lengthatleast\ x$$

$$prop\_lengthatmost$$
$$= \lambda xs \rightarrow whenFail$$
$$\texttt{"reverse duplicates list elements"}$$
$$(length\ (reverse\ xs) \leqslant length\ xs)$$

$$prop\_lengthatleast$$
$$= \lambda xs \rightarrow whenFail$$
$$\texttt{"reverse throws away list elements"}$$
$$(length\ (reverse\ xs) \geqslant length\ xs)$$

Testing example revisited

For the following erroneous student solution

$$
\begin{aligned}
&\mathit{reverse} = \mathit{reverse}'\ [\,] \\
&\quad \textbf{where}\ \mathit{reverse}'\ \mathit{acc}\ [\,] = [\,] \\
&\qquad\quad \mathit{reverse}'\ \mathit{acc}\ (x : xs) = \mathit{reverse}'\ (x : \mathit{acc})\ xs
\end{aligned}
$$

QuickCheck gives:

quickCheck property
Falsifiable, after 3 *tests* :
`"reverse throws away list elements"`
`"counterexample: "` [1]

Incorrectness

- Using testing we can prove that a student solution is incorrect
- We cannot say anything about correct solutions

Universiteit Utrecht

Why not only do testing?

fromBin converts a list of binary numbers to its decimal representation:

$$fromBin\ [1,0,1,0,1,0]$$
$$\Rightarrow 42$$

A solution:

$$fromBin :: [Int] \rightarrow Int$$
$$fromBin = fromBin'\ 2$$

$$fromBin'\ n\ [\,] = 0$$
$$fromBin'\ n\ (x : xs) = x * n\ {}^\wedge\ (length\ (x : xs) - 1) + fromBin'\ n\ xs$$

Why not only do testing?

This solution satisfies the expected properties, but it contains a number of (serious) imperfections:

- The length calculation is inefficient
- It takes time quadratic in the size of the input list
- Argument n is constant and should be abstracted

These imperfections occur frequently in student solutions.

$$
\begin{aligned}
&\mathit{fromBin} :: [\mathit{Int}] \to \mathit{Int}\\
&\mathit{fromBin} = \mathit{fromBin}'\ 2\\
\\
&\mathit{fromBin}'\ n\ [\,] = 0\\
&\mathit{fromBin}'\ n\ (x : xs) = x * n \mathbin{\wedge} (\mathit{length}\ (x : xs) - 1)\\
&\qquad + \mathit{fromBin}'\ n\ xs
\end{aligned}
$$

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

Where is the error?

- Using QuickCheck we can generate counterexamples for erroneous solutions
- But where is the error?
- Interpret a property as a contract
- Infer contracts for components
- Determine contract violations using the counterexample

[NIOC 2013: Adding testing to Ask-Elle – An Interactive Functional Programming Tutor]

Testing example revisited

reverse satisfies the contract:

$$\lambda xs \rightarrow length\ (reverse\ xs) \equiv length\ xs$$

For the erroneous solution

$$\begin{aligned} &reverse = reverse'\ [\,] \\ &\quad \textbf{where}\ reverse'\ acc\ [\,] = [\,] \\ &\qquad\quad reverse'\ acc\ (x : xs) = reverse'\ (x : acc)\ xs \end{aligned}$$

we might infer that $reverse'$ satisfies the contract

$$\lambda xs \rightarrow length\ (reverse'\ xs\ ys) \equiv length\ xs + length\ ys$$

Using the inferred contract, we can show that the first line of $reverse'$ violates the contract.

Normalisation

$$
\begin{array}{l}
\mathit{range1}\ x\ y = \mathbf{if}\ x \equiv y\ \mathbf{then}\ [x]\ \mathbf{else}\ x : \mathit{range1}\ (x+1)\ y \\
\mathit{range}_2\ x\ y = \mathbf{if}\ y \equiv x\ \mathbf{then}\ [x]\ \mathbf{else}\ x : \mathit{range}_2\ (x+1)\ y \\
\mathit{range}_3\ x\ y = \mathbf{if}\ x \not\equiv y\ \mathbf{then}\ x : \mathit{range}_3\ (x+1)\ y\ \mathbf{else}\ [x] \\
\mathit{range}_4\ x\ y = \mathbf{if}\ y \not\equiv x\ \mathbf{then}\ x : \mathit{range}_4\ (x+1)\ y\ \mathbf{else}\ [x] \\
\mathit{range}_5\ x\ y = \mathbf{if}\ x \not\equiv y\ \mathbf{then}\ x : \mathit{range}_5\ (1+x)\ y\ \mathbf{else}\ [x]
\end{array}
$$

-- and the 3 variants

$$
\mathit{range}_6\ x \quad = \lambda y \rightarrow \mathbf{if}\ x \equiv y\ \mathbf{then}\ [x]\ \mathbf{else}\ x : \mathit{range}_6\ (x+1)\ y
$$

-- and the 7 variants

$$
\begin{array}{ll}
\mathit{range}_7 \quad = \lambda x \rightarrow \lambda y \rightarrow & \mathbf{if}\ x \equiv y \\
& \mathbf{then}\ [x] \\
& \mathbf{else}\ x : \mathit{range}_7\ (x+1)\ y
\end{array}
$$

-- and the 7 variants

Conclusions

- Ask-Elle is a programming tutor for Haskell with advanced feedback functionality: both for correctness and incorrectness
- Easy to add and adapt programming exercises

- J.T.Jeuring@uu.nl
- General information: `http://ideas.cs.uu.nl/`
- Experiment on-line: `http://ideas.cs.uu.nl/ProgTutor/`
- Sources: `http://ideas.cs.uu.nl/trac/wiki/Download`