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

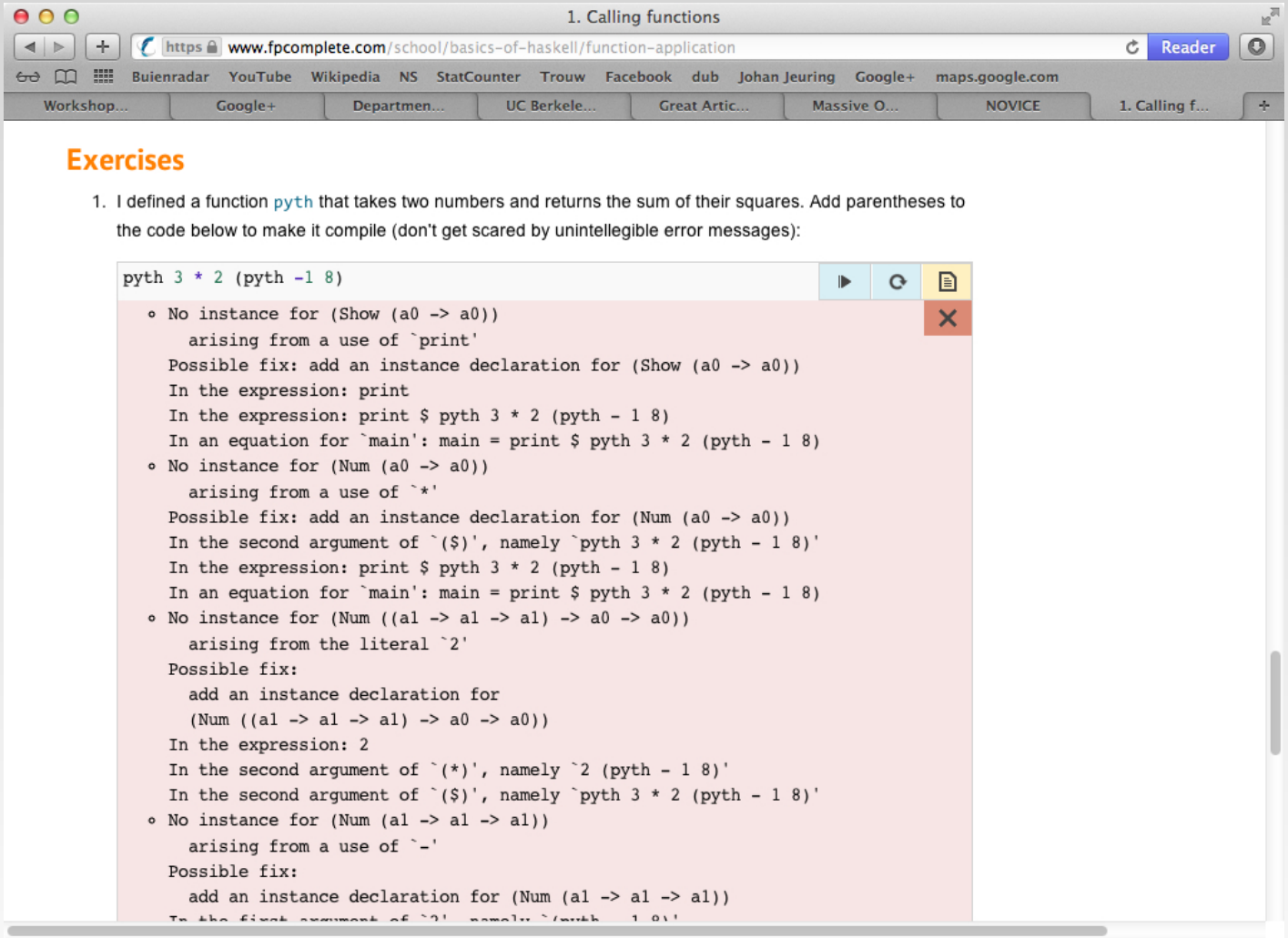
Open Universiteit
www.ou.nl

Universiteit Utrecht

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



Universiteit Utrecht

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

Universiteit Utrecht

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

Universiteit Utrecht

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

Universiteit Utrecht

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

Universiteit Utrecht

Ask-Elle: A programming tutor for Haskell

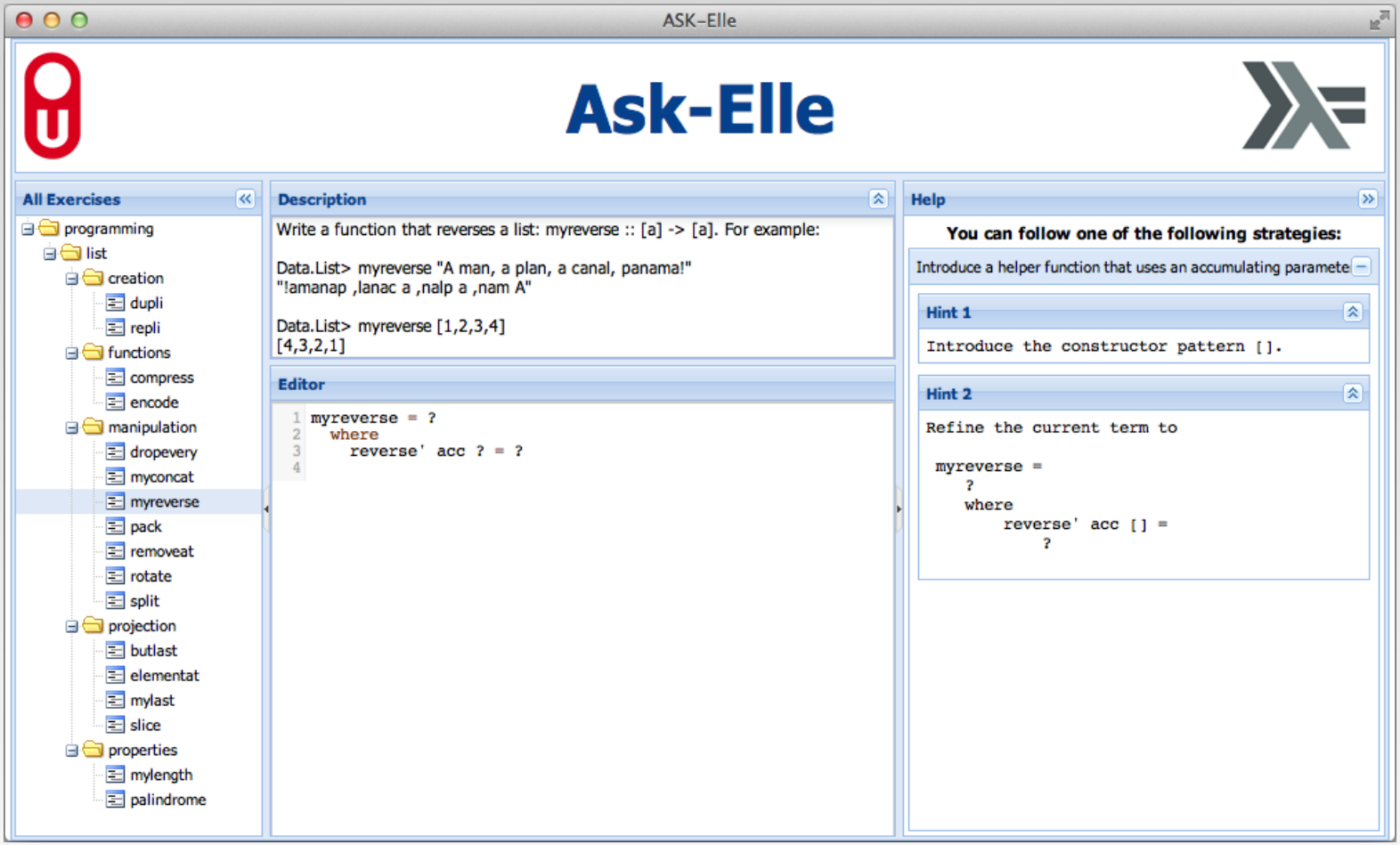
We are developing Ask-Elle: a programming tutor for Haskell. Using the tutor, a student can:

- 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.

Universiteit Utrecht


ASK-Elle
Ask-Elle
All Exercises
programming
list
creation
dupli
repli
functions
compress
encode
manipulation
dropevery
myconcat
myreverse
pack
removeat
rotate
split
projection
butlast
elementat
mylast
slice
properties
mylength
palindrome
Description
Write a function that reverses a list: myreverse :: [a] -> [a]. For example:
Data.List> myreverse "A man, a plan, a canal, panama!"
"!amanap ,lanac a ,nalp a ,nam A"
Data.List> myreverse [1,2,3,4]
[4,3,2,1]
Editor
1 myreverse = ?
2 where
3 reverse' acc ? = ?
4
Help
You can follow one of the following strategies:
Introduce a helper function that uses an accumulating paramete
Hint 1
Introduce the constructor pattern [].
Hint 2
Refine the current term to
myreverse =
?
where
reverse' acc [] =
?

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:

$$
\begin{array}{l}
\mathit{reverse} = \mathit{reverse}'\ ? \\
\quad \textbf{where} \\
\quad\quad \mathit{reverse}'\ \mathit{acc}\ ?\ =\ ?
\end{array}
$$

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'.

Universiteit Utrecht

An example interactive session

Programming task: write a program that reverses a list:

$$
\begin{array}{l}
reverse = reverse'\ [\,] \\
\quad \textbf{where} \\
\quad\quad reverse'\ acc\ [\,] = ?
\end{array}
$$

Tutor response on Hint:

Refine the right hand side of the empty list case.

Universiteit Utrecht

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.

Universiteit Utrecht

An example interactive session

Programming task: write a program that reverses a list:

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

Tutor response on Hint:

Define the non-empty list case of $\mathit{reverse}'$

Universiteit Utrecht

An example interactive session

Programming task: write a program that reverses a list:

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

Tutor response on Hint:

Define the recursive call of $\mathit{reverse}'$

Universiteit Utrecht

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
```

Universiteit Utrecht

An example interactive session

Programming task: write a program that reverses a list:

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

Tutor response on Check:

You have correctly solved the exercise.

Universiteit Utrecht

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.

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

$$
\begin{aligned}
&reverse_2 = reverse'_2\ [\,] \\
&\quad \textbf{where}\ reverse'_2\ acc\ [\,] &&= acc \\
&\qquad\qquad reverse'_2\ acc\ (x : xs) &&= reverse'_2\ (x : acc)\ xs
\end{aligned}
$$

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

Universiteit Utrecht

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* (:)) []

Universiteit Utrecht

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

Meta information for *reverse*

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

$$
\begin{aligned}
&\mathit{function} = \mathit{reverse}\\
&\mathit{type} \quad\;\; = [a] \to [a]\\
&\mathit{groups} \quad = \mathit{programming}.\mathit{FP}\\
&\mathit{property} = (\lambda \mathit{xs} \to \mathit{whenFail}\\
&\qquad\qquad \texttt{"reverse does not reverse a list"}\\
&\qquad\qquad (\mathit{reverse}\ \mathit{xs} \equiv \mathit{reverse}_1\ \mathit{xs})\\
&\qquad\quad)
\end{aligned}
$$

property is the standard property:

$$\mathit{program}_{\mathit{student}} \equiv \mathit{program}_{\mathit{model}}$$

Universiteit Utrecht

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

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

QuickCheck gives:

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

Universiteit Utrecht

More informative properties for testing

```
property = λx  →  prop_lengthatmost x
                  .&&. prop_lengthatleast x

prop_lengthatmost
    = λxs → whenFail
          "reverse duplicates list elements"
          (length (reverse xs) ⩽ length xs)

prop_lengthatleast
    = λxs → whenFail
          "reverse throws away list elements"
          (length (reverse xs) ⩾ length xs)
```

Universiteit Utrecht

Testing example revisited

For the following erroneous student solution

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

QuickCheck gives:

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

Universiteit Utrecht

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:

$$
\begin{aligned}
&\mathit{fromBin}\ [1,0,1,0,1,0] \\
&\quad \Rightarrow 42
\end{aligned}
$$

A solution:

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

Universiteit Utrecht

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.

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

Outline of presentation

Motivation

Ask-Elle: demo

Feedback

Future work and conclusions

Universiteit Utrecht

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

Testing example revisited

reverse satisfies the contract:

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

For the erroneous solution

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

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

$$
\begin{aligned}
&range1\ x\ y = \textbf{if}\ x \equiv y\ \textbf{then}\ [x]\ \textbf{else}\ x : range1\ (x+1)\ y \\
&range_2\ x\ y = \textbf{if}\ y \equiv x\ \textbf{then}\ [x]\ \textbf{else}\ x : range_2\ (x+1)\ y \\
&range_3\ x\ y = \textbf{if}\ x \not\equiv y\ \textbf{then}\ x : range_3\ (x+1)\ y\ \textbf{else}\ [x] \\
&range_4\ x\ y = \textbf{if}\ y \not\equiv x\ \textbf{then}\ x : range_4\ (x+1)\ y\ \textbf{else}\ [x] \\
&range_5\ x\ y = \textbf{if}\ x \not\equiv y\ \textbf{then}\ x : range_5\ (1+x)\ y\ \textbf{else}\ [x] \\
&\qquad \text{-- and the 3 variants} \\
&range_6\ x \quad = \lambda y \rightarrow \textbf{if}\ x \equiv y\ \textbf{then}\ [x]\ \textbf{else}\ x : range_6\ (x+1)\ y \\
&\qquad \text{-- and the 7 variants} \\
&range_7 \qquad = \lambda x \rightarrow \lambda y \rightarrow \textbf{if}\ x \equiv y \\
&\qquad\qquad\qquad\qquad\qquad\quad \textbf{then}\ [x] \\
&\qquad\qquad\qquad\qquad\qquad\quad \textbf{else}\ x : range_7\ (x+1)\ y \\
&\qquad \text{-- and the 7 variants}
\end{aligned}
$$

Universiteit Utrecht

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`