Certified Software Development with Dependent Types in Idris

Lecture 1

Vitaly Bragilevsky (bravit@sfedu.ru)

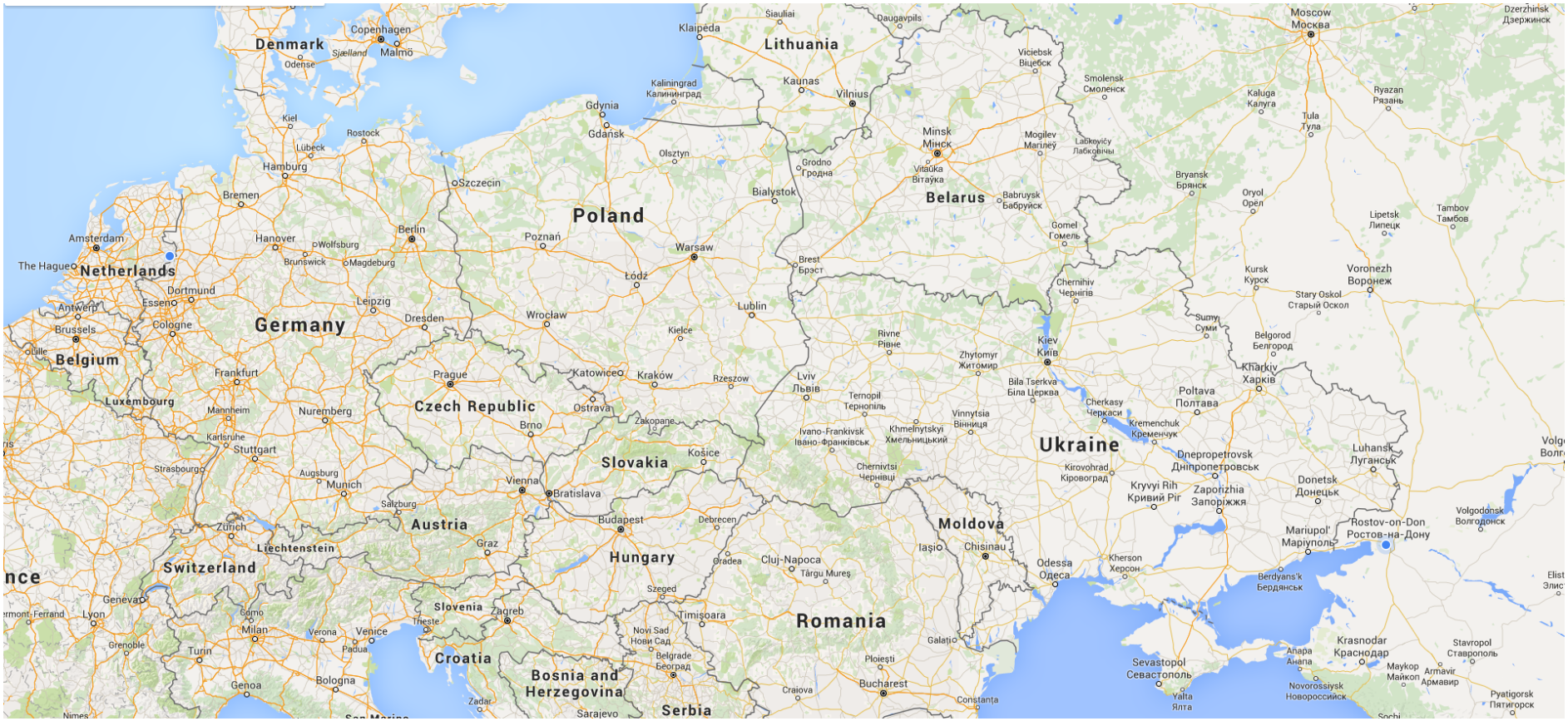
I.I. Vorovich Institute of Mathematics, Mechanics and Computer Science
Southern Federal University, Rostov-on-Don, Russia
part of Erasmus+ teaching mobility agreement with University of Twente

Content

Who am I

- Vitaly Bragilevsky
- Senior lecturer at the Southern Federal University (Rostov-on-Don, Russia)
- Translator from English to Russian
- Haskell Prime 2020 Committee member (five days since Thursday)
- Interests: functional programming, theory of programming languages, theory of computations (models of computation, computability, complexity)

Where is Rostov-on-Don



Are you aware of...

- Imperative programming (C/C++, Java, C#, etc)
- Functional programming (Lisp, ML, Haskell, etc)
- Proof assistants (Coq, Isabelle)
- Compilers and interpreters
- Dynamic and static typing
- Lazy and eager evaluation
- Algebraic datatypes
- Logic (boolean formulas, predicates, quantifiers)
- Proof theory (natural deduction, sequent calculus, modus ponens)
- Lambda calculus
- Type theory
- Curry—Howard correspondence
- Dependent types

Main topics

1. Functional programming in strongly typed languages
2. Programming in Idris
3. Dependent types and proofs

Schedule: lectures

The room number is Zilverling 1016.

- Week 1:
  2nd of May: 11.00 (!) – 12.30
  3rd of May: 10.45 – 12.30
  4th of May: 15.45 – 17.30
- Week 2:
  9th of May: 10.45 – 12.30
  10th of May: 10.45 – 12.30
  11th of May: 15.45 – 17.30
  12th of May: 15.45 – 17.30
  13th of May: 13.45 – 15.30
- Week 3:
  17th of May (Tuesday): 10.45 – 12.30
  18th of May: 15.45 – 17.30
  19th of May: 14.30 (!) – 16.30 (!)

Assignments & Interaction

1. 3 homework assignments — 60%
2. final project — 40%
3. assignments will be given as private github.com repositories
4. course materials on Facebook?
5. deadlines?

Tools

1. Idris: `http://www.idris-lang.org/download/`
   - for Windows and OS X use binary distributions
   - for Linux use Haskell platform (`https://www.haskell.org/platform/`)
2. Text editors:
   - Emacs with idris-mode
   - Vim with idris-vim
   - Atom (`https://atom.io`, `https://github.com/idris-hackers/atom-language-idris/blob/master/documentation/tutorial.md`)

Content

- Philip Wadler, Propositions as types: paper & video

- Idris has strong static type system with dependent types
- Idris allows to write certified programs

Curry—Howard Correspondence

Proposition A — can be either true or false	**Type** A — can be either inhabited or not
$A \wedge B$ (conjunction)	$A \times B$ (pair, product)
$A \vee B$ (disjunction)	$A + B$ (union, sum)
$A \to B$ (implication)	$A \to B$ (function)
$\frac{A \to B \quad A}{B}$ (proof, rule of modus ponens)	`f(x)`, function call — x is a value of type A and result is of type B
False	$\bot$ (bottom, empty type)
$\neg A$	$A \to \bot$
Predicate $P(x)$	dependent type (type depends on value)
$\forall x P(x)$	$x \to P(x)$, dependent function
$\exists x P(x)$	$(x, P(x))$, dependent pair

Certification and dependent types

- Certification means that our program satisfies some properties of correctness (predicates).
- Dependent types correspond to predicates.
- **If we give specific dependent type for our program and compiler checks it then correctness is ensured.**

Content

```
import Data.Vect

hello : String
hello = "Hello world"

zeroes : Vect 5 Int
zeroes = [0,0,0,0,0]

a : Nat
a = 42

pred' : Nat -> Nat
pred' Z = Z
pred' (S a) = a

--theorem : a+b=b+a
```

```
> S(S(S(S Z))) + (S Z)
5 : Nat
> the Nat 42
42 : Nat
> :type the
the : (a : Type) -> a -> a
> :type Vect
Vect : Nat -> Type -> Type
```

Bibliography

- Philip Wadler. Propositions as types (`http://homepages.inf.ed.ac.uk/wadler/papers/propositions-as-types/propositions-as-types.pdf`).
- The Idris Tutorial `http://docs.idris-lang.org/en/latest/tutorial/`