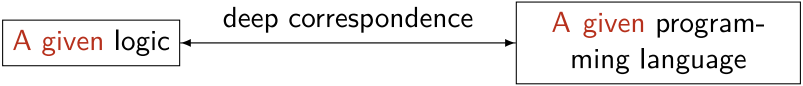
Propositions as Types in Agda

Andrés Sicard-Ramírez

Universidad EAFIT

Encuentro Álgebra y Lógica
Universidad Tecnológica de Pereira
9 December 2015

Propositions as Types: Introduction



Propositions as Types: Introduction

Correspondence's levels

(Wadler 2015)

P. Wadler [2015]. Propositions as Types. Communications of the ACM.

Propositions as Types: Introduction

Correspondence's levels

(Wadler 2015)

1. Propositions as types

   'For each proposition in the logic there is a corresponding type in the programming language—and vice versa.'

P. Wadler [2015]. Propositions as Types. Communications of the ACM.

Propositions as Types: Introduction

Correspondence's levels

(Wadler 2015)

1. Propositions as types

   'For each proposition in the logic there is a corresponding type in the programming language—and vice versa.'

2. Proofs as programs

   'For each proof of a given proposition, there is a program of the corresponding type—and vice versa.'

P. Wadler [2015]. Propositions as Types. Communications of the ACM.

Propositions as Types: Introduction

Correspondence's levels

(Wadler 2015)

1. Propositions as types

   'For each proposition in the logic there is a corresponding type in the programming language—and vice versa.'

2. Proofs as programs

   'For each proof of a given proposition, there is a program of the corresponding type—and vice versa.'

3. Simplification of proofs as evaluation of programs

   'For each way to simplify a proof there is a corresponding way to evaluate a program—and vice versa.'

P. Wadler [2015]. Propositions as Types. Communications of the ACM.

Agda: Introduction

Interactive proof assistants

'Proof assistants are computer systems that allow a user to do mathematics on a computer, but not so much the computing (numerical or symbolical) aspect of mathematics but the aspects of proving and defining. So a user can set up a mathematical theory, define properties and do logical reasoning with them.' (Geuvers 2009, p. 3.)

Examples

Agda, Coq and Isabelle among others.

H. Geuvers [2009]. Proof Assistants: History, Ideas and Future. Sadhana.

Agda: Introduction

Agda

- Chalmers University of Technology and University of Gothenburg (Sweden)
- Based on Martin-Löf type theory
- Direct manipulation of proofs-objects
- Back-ends to Haskell (GHC and UHC)
- Written in Haskell
- Current version: Agda 2.4.2.4

Agda: Introduction

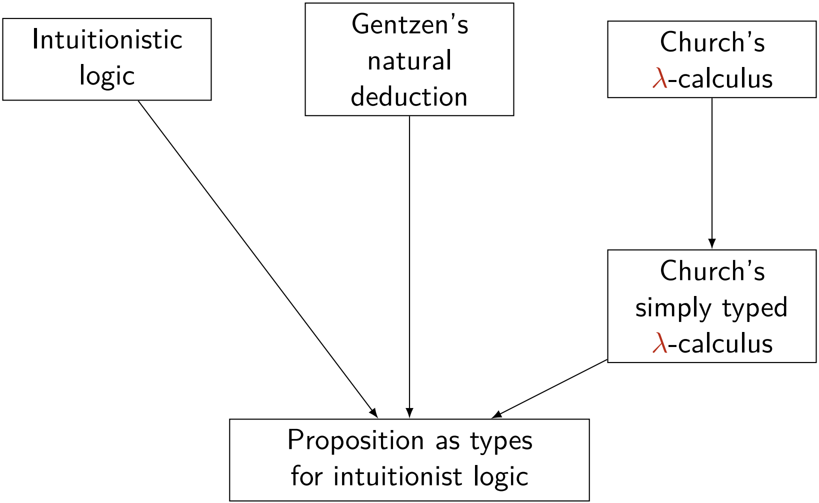
Agda

- Chalmers University of Technology and University of Gothenburg (Sweden)
- Based on Martin-Löf type theory
- Direct manipulation of proofs-objects
- Back-ends to Haskell (GHC and UHC)
- Written in Haskell
- Current version: Agda 2.4.2.4

Isabelle

- University of Cambridge (England) and Technical University of Munich (German)
- Based on higher-order logic
- Tactic-based
- Extraction of programs to Haskell, OCaml, Scala and SML
- Written in SML
- Integration with ATPs and SMT solvers
- Current version: Isabelle2015

Propositions as Types: First Presentation



Constructive Interpretation of the Logical Constants

a proof of the proposition	consist of (Brower-Heyting-Kolmogorov interpretation)	has the form
$A \wedge B$	a proof of A and a proof of B	(a, b), where a is a proof of A and b is a proof of B
$A \vee B$	a proof of A or a proof of B	$\mathsf{inl}(a)$, where a is a proof of A, or $\mathsf{inr}(b)$, where b is a proof of B
$\perp$	has not proof	
$A \supset B$	a method which takes any proof of A into a proof of B	$\lambda x.b(x)$, where $b(a)$ is a proof of B provided a is a proof of A

Gentzen's Natural Deduction

Inference rules: Introduction and elimination

$$\frac{A \qquad B}{A \,\&\, B}\ \&\text{-I} \qquad \frac{A \,\&\, B}{A}\ \&\text{-E}_1 \qquad \frac{A \,\&\, B}{B}\ \&\text{-E}_2$$

$$\frac{\begin{array}{c}[A]^x \\ \vdots \\ B\end{array}}{A \supset B}\ \supset\text{-I}^x \qquad \frac{A \supset B \qquad A}{B}\ \supset\text{-E}$$

(Figure 1 of Wadler (2015))

Gentzen's Natural Deduction

Example (Proof example)

$$
\dfrac{\dfrac{[B \mathbin{\&} A]^z}{A}\ \&\text{-E}_2 \qquad \dfrac{[B \mathbin{\&} A]^z}{B}\ \&\text{-E}_1}{\dfrac{A \mathbin{\&} B}{(B \mathbin{\&} A) \supset (A \mathbin{\&} B)}\ \supset\text{-I}^z}\ \&\text{-I}
$$

(Figure 1 of Wadler (2015))

Church's Simply Typed λ-Calculus

Type assignment rules: Introduction and elimination

$$\frac{M : A \qquad N : B}{\langle M, N\rangle : A \times B}\ \times\text{-I} \qquad \frac{L : A \times B}{\pi_1\, L : A}\ \times\text{-E}_1 \qquad \frac{L : A \times B}{\pi_2\, L : B}\ \times\text{-E}_2$$

$$\frac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x. N : A \to B}\ \to\text{-I}^x \qquad \frac{L : A \to B \qquad M : A}{L\, M : B}\ \to\text{-E}$$

(Figure 5 of Wadler (2015))

Church's Simply Typed λ-Calculus

Example (Program example)

$$
\dfrac{
  \dfrac{[z : B \times A]^{z}}{\pi_2\, z : A}\ \times\text{-E}_2
  \qquad
  \dfrac{[z : B \times A]^{z}}{\pi_1\, z : B}\ \times\text{-E}_1
}{
  \dfrac{\langle \pi_2\ z, \pi_1\, z \rangle : A \times B}{\lambda z\, .\, \langle \pi_2\, z, \pi_1\, z \rangle : (B \times A) \to (A \times B)}\ \to\text{-I}^{z}
}\ \times\text{-I}
$$

(Figure 6 of Wadler (2015))

Agda demo

Propositions as Types on the Logical Constants

(conjunction)	$A \wedge B = A \times B$	(product type)
(disjunction)	$A \vee B = A + B$	(sum type)
(implication)	$A \supset B = A \to B$	(function type)
(falsehood)	$\bot = \bot$	(empty type)
(negation)	$\neg A = A \to \bot$	

Further Subjects

- Propositions as types on predicate logic (which requires dependent types on the programming language)
- Propositions as types on other (e.g. classical, modal, linear) logics
- Verification of programs using dependently typed λ-calculus

Further Reading

Propositions as types

- P. Wadler [2015]. Propositions as Types. Communications of the ACM
- M.-H. Sørensen and P. Urzyczyn [2006]. Lectures on the Curry-Howard Isomorphism.

Agda

- A. Bove and P. Dybjer [2009]. Dependent Types at Work.
- U. Norell [2009]. Dependently Typed Programming in Agda.

Thanks!