Haskell and Domain-Specific Languages

Haskell nejen pro informatiky

Otakar Smrž

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University in Prague
otakar.smrz@mff.cuni.cz

https://wiki.ufal.ms.mff.cuni.cz/courses:pfl080

Part I

Curry–Howard Isomorphism

Curry–Howard Isomorphism

Discovery of a one-to-one correspondence between types in programming and propositions in logic (3, 2).

Curry–Howard Isomorphism

Discovery of a one-to-one correspondence between types in programming and propositions in logic (3, 2).

$$\frac{\Gamma, B \vdash A}{\Gamma \vdash B \rightarrow A} \qquad \frac{\Gamma \vdash B \rightarrow A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A}$$

$(\rightarrow)$ introduction $\qquad$ $(\rightarrow)$ elimination

Curry–Howard Isomorphism

Discovery of a one-to-one correspondence between types in programming and propositions in logic (3, 2).

$$\frac{\Gamma, B \vdash A}{\Gamma \vdash B \rightarrow A}$$

$(\rightarrow)$ introduction

$$\frac{\Gamma \vdash B \rightarrow A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A}$$

$(\rightarrow)$ elimination

$$\frac{\Gamma, x : B \vdash t : A}{\Gamma \vdash \lambda x.t : B \rightarrow A}$$

lambda abstraction

$$\frac{\Gamma \vdash t : B \rightarrow A \qquad \Delta \vdash u : B}{\Gamma, \Delta \vdash t(u) : A}$$

function application

Part II

Existential Types

Existential Types

The isomorphism also extends from quantifiers in intuitionistic predicate calculus to polymorphism with existential types.

Existential Types

The isomorphism also extends from quantifiers in intuitionistic predicate calculus to polymorphism with existential types.

$$
\begin{array}{rcl}
(\forall x.P \rightarrow Q) & \Leftrightarrow & P \rightarrow (\forall x.Q) \\
(\forall x.Q \rightarrow P) & \Leftrightarrow & (\exists x.Q) \rightarrow P \\
(\exists x.P \rightarrow Q) & \Rightarrow & P \rightarrow (\forall x.Q) \\
(\exists x.Q \rightarrow P) & \Rightarrow & (\forall x.P) \rightarrow P
\end{array}
$$

Existential Types

The isomorphism also extends from quantifiers in intuitionistic predicate calculus to polymorphism with existential types.

$$
\begin{array}{rcl}
(\forall x.P \to Q) & \Leftrightarrow & P \to (\forall x.Q) \\
(\forall x.Q \to P) & \Leftrightarrow & (\exists x.Q) \to P \\
(\exists x.P \to Q) & \Rightarrow & P \to (\forall x.Q) \\
(\exists x.Q \to P) & \Rightarrow & (\forall x.P) \to P \\
\\
(\forall a.\forall x.T\ a \to \tau) & \Leftrightarrow & \forall a.T\ a \to (\forall x.\tau) \\
(\forall a.\forall x.\tau \to T\ a) & \Leftrightarrow & \forall a.(\exists x.\tau) \to T\ a \\
\ldots & \ldots & \ldots
\end{array}
$$

Existential Types

The isomorphism also extends from quantifiers in intuitionistic predicate calculus to polymorphism with existential types.

$$
\begin{array}{rcl}
(\forall x.P \to Q) & \Leftrightarrow & P \to (\forall x.Q) \\
(\forall x.Q \to P) & \Leftrightarrow & (\exists x.Q) \to P \\
(\exists x.P \to Q) & \Rightarrow & P \to (\forall x.Q) \\
(\exists x.Q \to P) & \Rightarrow & (\forall x.P) \to P
\end{array}
$$

$$
\begin{array}{rcl}
(\forall a.\forall x.T\ a \to \tau) & \Leftrightarrow & \forall a.T\ a \to (\forall x.\tau) \\
(\forall a.\forall x.\tau \to T\ a) & \Leftrightarrow & \forall a.(\exists x.\tau) \to T\ a \\
\ldots & \ldots & \ldots
\end{array}
$$

For assumptions of these statements, and for precise discussion, please see (1, 2).

References

- Mark P. Jones.
  First-class Polymorphism with Type Inference.
  In *Conference Record of POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 483–496, Paris, France, 15–17 1997.
- Benjamin C. Pierce.
  Types and Programming Languages.
  MIT Press, Cambridge, MA, USA, 2002.
- Philip Wadler.
  Proofs Are Programs: 19th Century Logic and 21st Century Computing.
  December 2000.
  Appeared in Dr. Dobbs Journal as 'New Languages, Old Logic'.