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).

\[
\begin{align*}
\Gamma, B & \vdash A \\
\Gamma & \vdash B \rightarrow A \\
\end{align*}
\]

(\rightarrow) introduction

\[
\begin{align*}
\Gamma & \vdash B \rightarrow A \\
\Delta & \vdash B \\
\Gamma, \Delta & \vdash A \\
\end{align*}
\]

(\rightarrow) elimination
Curry–Howard Isomorphism

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

\[ \Gamma, B \vdash A \quad \Gamma \vdash B \rightarrow A \]
\[ \rightarrow \text{ introduction} \]

\[ \Gamma \vdash B \rightarrow A \quad \Delta \vdash B \quad \Gamma, \Delta \vdash A \]
\[ \rightarrow \text{ elimination} \]

\[ \Gamma, x : B \vdash t : A \]
\[ \Gamma \vdash \lambda x.t : B \rightarrow A \]
\[ \text{lambda abstraction} \]

\[ \Gamma \vdash t : B \rightarrow A \quad \Delta \vdash u : B \]
\[ \Gamma, \Delta \vdash t(u) : A \]
\[ \text{function application} \]
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.

\[(\forall x. P \to Q) \iff P \to (\forall x. Q)\]
\[(\forall x. Q \to P) \iff (\exists x. Q) \to P\]
\[(\exists x. P \to Q) \implies P \to (\forall x. Q)\]
\[(\exists x. Q \to P) \implies (\forall x. P) \to P\]

For assumptions of these statements, and for precise discussion, please see [1, 2].
Existential Types

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

\[(\forall x. P \to Q) \iff P \to (\forall x. Q)\]
\[(\forall x. Q \to P) \iff (\exists x. Q) \to P\]
\[(\exists x. P \to Q) \implies P \to (\forall x. Q)\]
\[(\exists x. Q \to P) \implies (\forall x. P) \to P\]

\[(\forall a.\forall x. T a \to \tau) \iff \forall a. T a \to (\forall x. \tau)\]
\[(\forall a.\forall x. \tau \to T a) \iff \forall a. (\exists x. \tau) \to T a\]

\[\ldots \ldots \ldots \ldots\]
Existential Types

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

\[(\forall x. P \to Q) \iff P \to (\forall x. Q)\]
\[(\forall x. Q \to P) \iff (\exists x. Q) \to P\]
\[(\exists x. P \to Q) \implies P \to (\forall x. Q)\]
\[(\exists x. Q \to P) \implies (\forall x. P) \to P\]

\[(\forall a. \forall x. T \ a \to \tau) \iff \forall a. T \ a \to (\forall x. \tau)\]
\[(\forall a. \forall x. \tau \to T \ a) \iff \forall a. (\exists x. \tau) \to T \ a\]

\[\ldots \quad \ldots \quad \ldots \quad \ldots \]

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

Mark P. Jones.
First-class Polymorphism with Type Inference.

Benjamin C. Pierce.
Types and Programming Languages.

Philip Wadler.
December 2000.
Appeared in Dr. Dobbs Journal as ‘New Languages, Old Logic’.