-NoValue-

pty.1 Introduction

int:pty:int:sec

Historically the lambda calculus and intuitionistic logic were developed separately. Haskell Curry and William Howard independently discovered a close similarity: types in a typed lambda calculus correspond to formulas in intuitionistic logic in such a way that a derivação of a fórmula corresponds directly to a typed lambda term with that fórmula as its type. Moreover, beta reduction in the typed lambda calculus corresponds to certain transformations of derivações.

For instance, a derivação of $\varphi \to \psi$ corresponds to a term $\lambda x^{\varphi}.\, N^{\psi}$, which has the function type $\varphi \to \psi$. The inference rules of natural deduction correspond to typing rules in the typed lambda calculus, p.ex.,

$$\dfrac{\begin{array}{c}[\varphi]^x \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\, {\scriptstyle x}\,{\to}\text{Intro} \qquad \text{corresponds to} \qquad \dfrac{x : \varphi \;\Rightarrow\; N : \psi}{\Rightarrow\; \lambda x^{\varphi}.\, N^{\psi} : \varphi \to \psi}\,\lambda$$

where the rule on the right means that if x is of type φ and N is of type ψ, then $\lambda x^{\varphi}.\, N$ is of type $\varphi \to \psi$.

The $\to$Elim rule corresponds to the typing rule for composition terms, O.s.,

$$\dfrac{\dfrac{\varphi \to \psi \qquad \varphi}{\psi}\,{\to}\text{Elim}}{\dfrac{\Rightarrow\; P : \varphi \to \psi \qquad \Rightarrow\; Q : \varphi}{\Rightarrow\; P^{\varphi\to\psi} Q^{\varphi} : \psi}\,\text{app}} \quad \text{corresponds to}$$

If a $\to$Intro rule is followed immediately by a $\to$Elim rule, the derivação can be simplified:

$$\dfrac{\dfrac{\begin{array}{c}[\varphi]^x \\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\,{\scriptstyle x}\,{\to}\text{Intro} \qquad \begin{array}{c}\vdots \\ \varphi\end{array}}{\psi}\,{\to}\text{Elim} \qquad \to \qquad \begin{array}{c}[\varphi] \\ \vdots \\ \psi\end{array}$$

which corresponds to the beta reduction of lambda terms

$$(\lambda x^{\varphi}.\, P^{\psi})Q \quad \to \quad P[Q/x].$$

Similar correspondences hold between the rules for $\wedge$ and "product" types, and between the rules for $\vee$ and "sum" types.

This correspondence between terms in the simply typed lambda calculus and natural deduction derivações is called the "Curry–Howard", or "propositions as types" correspondence. In addition to fórmulas (propositions) corresponding to types, and proofs to terms, we can summarize the correspondences as follows:

logic	program
proposition	type
proof	term
assumption	variable
discharged assumption	bind variable
not discharged assumption	free variable
implication	function type
conjunction	product type
disjunction	sum type
absurdity	bottom type

The Curry–Howard correspondence is one of the cornerstones of automated proof assistants and type checkers for programs, since checking a proof witnessing a proposition (as we did above) amounts to checking if a program (term) has the declared type.

Créditos das Imagens

Referências Bibliográficas