Introduction

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 derivation of a formula corresponds directly to a typed lambda term with that formula as its type. Moreover, beta reduction in the typed lambda calculus corresponds to certain transformations of derivations.

For instance, a derivation of $\varphi \rightarrow \psi$ corresponds to a term $\lambda x^\varphi. N^\psi$, which has the function type $\varphi \rightarrow \psi$. The inference rules of natural deduction correspond to typing rules in the typed lambda calculus, e.g.,

\[
\frac{\varphi \rightarrow \psi}{x : \varphi \Rightarrow N : \psi \Rightarrow N^\varphi : \varphi \rightarrow \psi} \lambda
\]

where the rule on the right means that if $x$ is of type $\varphi$ and $N$ is of type $\psi$, then $\lambda x^\varphi. N$ is of type $\varphi \rightarrow \psi$.

The $\rightarrow$ Elim rule corresponds to the typing rule for composition terms, i.e.,

\[
\frac{\varphi \rightarrow \psi \quad \varphi \rightarrow P : \varphi \rightarrow \psi \quad Q : \varphi \Rightarrow P^\varphi \rightarrow Q^\psi : \psi}{\Rightarrow P^\varphi \rightarrow Q^\psi : \psi}
\]

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

\[
\frac{\varphi \rightarrow \psi \quad \hat{\varphi} \rightarrow \psi}{\Rightarrow P^\varphi : \psi \rightarrow P[\hat{\psi}]}
\]

which corresponds to the beta reduction of lambda terms

\[
(\lambda x^\varphi. P^\psi)Q \rightarrow P[Q/x].
\]

Similar correspondences hold between the rules for $\land$ and “product” types, and between the rules for $\lor$ and “sum” types.

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

### Photo Credits

### Bibliography