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 $\varphi \to \psi$ 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, e.g.,

\[
\begin{array}{c}
[\varphi]^x \\
\vdots \\
\vdots \\
\hline
z \\
\hline
\varphi \to \psi \\
\hline
\text{Intro}
\end{array}
\]

corresponds to

\[
\begin{array}{c}
x : \varphi \Rightarrow N : \psi \\
\hline
\Rightarrow \lambda x^\varphi . N^\psi : \varphi \Rightarrow \psi \\
\hline
\Rightarrow \lambda
\end{array}
\]

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 $\varphi \to \psi$.

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

\[
\begin{array}{c}
\varphi \to \psi \\
\hline
\Rightarrow P : \varphi \to \psi \\
\Rightarrow Q : \varphi \\
\hline
\Rightarrow P^\varphi \to \psi Q^\psi : \psi
\end{array}
\]

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

\[
\begin{array}{c}
[\varphi]^x \\
\vdots \\
\vdots \\
\hline
z \\
\hline
\varphi \to \psi \\
\hline
\text{Intro} \\
\hline
\varphi
\end{array}
\]

\[
\begin{array}{c}
\varphi \\
\hline
\Rightarrow \psi \\
\hline
\text{Elim} \\
\hline
\psi
\end{array}
\]

which corresponds to the beta reduction of lambda terms

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

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:
<table>
<thead>
<tr>
<th>logic</th>
<th>program</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposition</td>
<td>type</td>
</tr>
<tr>
<td>proof</td>
<td>term</td>
</tr>
<tr>
<td>assumption</td>
<td>variable</td>
</tr>
<tr>
<td>discharged assumption</td>
<td>bind variable</td>
</tr>
<tr>
<td>not discharged assumption</td>
<td>free variable</td>
</tr>
<tr>
<td>implication</td>
<td>function type</td>
</tr>
<tr>
<td>conjunction</td>
<td>product type</td>
</tr>
<tr>
<td>disjunction</td>
<td>sum type</td>
</tr>
<tr>
<td>absurdity</td>
<td>bottom type</td>
</tr>
</tbody>
</table>

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**