Refutation of the Curry-Howard correspondence

© Copyright 2019 by Colin James III  All rights reserved.

Abstract: In the Curry-Howard correspondence, the identity and composition combinators are not tautologous. In fact, the examples result in equivalent truth table values. Further demonstrated is that the instances of Hilbert, lambda, and sequent fragments are also not tautologous with a recent paper rendered moot. These artifacts form a non tautologous fragment of the universal logic VŁ4.

We assume the method and apparatus of Meth8/VŁ4 with Tautology as the designated proof value, F as contradiction, N as truthity (non-contingency), and C as falsity (contingency). The 16-valued truth table is row-major and horizontal, or repeating fragments of 128-tables, sometimes with table counts, for more variables. (See ersatz-systems.com.)

From: en.wikipedia.org/wiki/Curry–Howard_correspondence

In programming language theory and proof theory, the Curry–Howard correspondence … is the direct relationship between computer programs and mathematical proofs.

Examples
Thanks to the Curry–Howard correspondence, a typed expression whose type corresponds to a logical formula is analogous to a proof of that formula. Here are examples.

The identity combinator seen as a proof of $a \rightarrow a$ in Hilbert-style logic

As an example, consider a proof of the theorem $a \rightarrow a$. In lambda calculus, this is the type of the identity function $I = \lambda x.x$ and in combinatory logic, the identity function is obtained by applying $S = \lambda fgx.fx(gx)$ twice to $K = \lambda xy.x$. That is, $I = ((S K) K)$. As a description of a proof, this says that the following steps can be used to prove $a \rightarrow a$:

\[
\text{LET } p, q, r: a, \beta \rightarrow a \text{, so that to obtain a proof of } (a \rightarrow ((\beta \rightarrow a) \rightarrow a)) \rightarrow ((a \rightarrow (\beta \rightarrow a)) \rightarrow (a \rightarrow a)),
\]

\[
\text{(1.1)}
\]

\[
(p > ((q > p) > p)) > ((p > (q > p)) > (p > p)) ;
\]

\[
\text{TTTT FTTT TTTT FTTT (1.2)}
\]

\[
\text{LET } p, q, r: a, \beta, \Gamma.
\]

\[
(p > ((q > p) > p)) > ((p > (q > p)) > (p > p)) ;
\]

\[
\text{TTTT FTTT TTTT FTTT (1.2)}
\]

\[
\text{LET } p, q, r: a, \beta, \Gamma.
\]

\[
(p > ((q > p) > p)) > ((p > (q > p)) > (p > p)) ;
\]

\[
\text{TTTT FTTT TTTT FTTT (1.2)}
\]
instantiate the first axiom scheme a second time with $\alpha$ and $\beta$, so that to obtain a proof of $\alpha \rightarrow (\beta \rightarrow \alpha)$,

$$\begin{array}{cccc}
\text{p} & \text{(q>p)} & \text{TTTT} & \text{FTFT} & \text{TTTT} & \text{FTFT} \\
\end{array}$$

(3.1)

The composition combinator seen as a proof of $(\beta \rightarrow \alpha) \rightarrow (\Gamma \rightarrow \beta) \rightarrow \Gamma \rightarrow \alpha$ in Hilbert-style logic

As a more complicated example, let's look at the theorem that corresponds to the $B$ function. The type of $B$ is $(\beta \rightarrow \alpha) \rightarrow (\Gamma \rightarrow \beta) \rightarrow \Gamma \rightarrow \alpha$. $B$ is equivalent to $(S (K S) K)$. This is our roadmap for the proof of the theorem

$$(\beta \rightarrow \alpha) \rightarrow (\Gamma \rightarrow \beta) \rightarrow \Gamma \rightarrow \alpha.$$  

(4.1)

$$\begin{array}{cccc}
((\text{q>p})>(\text{r>q}))>\text{r}>\text{p} & \text{TTTT} & \text{FTFT} & \text{TTTT} & \text{FTFT} \\
\end{array}$$

(4.2)

The identity combinator (Eqs. 1-3) and composition combinator (Eq. 4) are not tautologous. In fact, the example steps given result in equivalent truth table values. Further demonstrated is that the instances of Hilbert, lambda, and sequent fragments are also not tautologous. Hence, the following paper becomes moot:


Abstract We develop a generalization of existing Curry-Howard interpretations of (binary) session types by relying on an extension of linear logic with features from hybrid logic, in particular modal worlds that indicate domains. These worlds govern domain migration, subject to a parametric accessibility relation familiar from the Kripke semantics of modal logic. The result is an expressive new typed process framework for domain-aware, message-passing concurrency.