Refutation of the Curry-Howard correspondence

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.)

LET ~ Not, ¬ ; + Or, ∨, ∪, ⊔ ; - Not Or; & And, ∧, ∩, ⊓, ·; \ Not And;
\> Imply, greater than, →, ⇒, ↦, ≻, ⊃, ↠ ; < Not Imply, less than, ∈, ≺, ⊂, ⊬, ⊭, ↞, ≲ ;
= Equivalent, ≡, :=, ⇔, ↔, ≜, ≈, ≃ ; @ Not Equivalent, ≠;
% possibility, for one or some, ∃, ◊, M; # necessity, for every or all, ∀, □, L;
(z=z) T as tautology, ⊤, ordinal 3; (z@z) **F** as contradiction, Ø, Null, ⊥, zero;
(%z>#z) N as non-contingency, Δ, ordinal 1; (%z<#z) C as contingency, ∇, ordinal 2;
~(y < x) (x ≤ y), (x ⊆ y), (x ⊑ y); (A=B) (A~B).
Note for clarity, we usually distribute quantifiers onto each designated variable.

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 $\alpha \to \alpha$ in Hilbert-style logic

As an example, consider a proof of the theorem $\alpha \to \alpha$. 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 $\alpha \to \alpha$:

instantiate the second axiom scheme with the formulas α, $\beta \to \alpha$ and α, so that to obtain a proof of $(\alpha \to ((\beta \to \alpha) \to \alpha)) \to ((\alpha \to (\beta \to \alpha)) \to (\alpha \to \alpha))$, (1.1)

LET p, q, r: α, β, Γ.

(p>((q>p)>p))>((p>(q>p))>(p>p)) ;
TTTT **F**T**F**T TTTT **F**T**F**T (1.2)

instantiate the first axiom scheme once with α and $\beta \to \alpha$, so that to obtain a proof of $\alpha \to ((\beta \to \alpha) \to \alpha)$, (2.1)

p>((q>p)>p) ; TTTT **F**T**F**T TTTT **F**T**F**T (2.2) instantiate the first axiom scheme a second time with α and β, so that to obtain a proof of $\alpha \to (\beta \to \alpha)$, (3.1)

p>(q>p) ; TTTT **F**T**F**T TTTT **F**T**F**T (3.2)

The composition combinator seen as a proof of $(\beta\to\alpha) \to (\Gamma\to\beta) \to \Gamma \to \alpha$ 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 $(\beta\to\alpha) \to (\Gamma\to\beta) \to \Gamma \to \alpha$. **B** is equivalent to (**S** (**K** **S**) **K**). This is our roadmap for the proof of the theorem

$$(\beta \to \alpha) \to (\Gamma \to \beta) \to \Gamma \to \alpha. \tag{4.1}$$

(((q>p)>(r>q))>r)>p ; TTTT **F**T**F**T TTTT **F**T**F**T (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:

Caires, L.; Pérez, J.A.; Pfenning, F.; Toninho, B. (2019).
Domain-aware session types (extended version). arxiv.org/pdf/1907.01318.pdf

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.