Curry–Howard by example

Tiny Explanations 4

Carl Joshua Quines

June 4, 2021

- The type of π is real.
- The type of 3 is int, for integer.
- The type of 3 can also be real. But the int 3 and real 3 are different, because they have different types. For the rest of the examples we'll assume 3 has type int.
- The type of $x \mapsto \lfloor x \rfloor$ is $\mathsf{real} \to \mathsf{int}$, because it takes a real number to an integer.
- We write $(x \mapsto \lfloor x \rfloor)(\pi) = 3$. By applying a function of type $\mathsf{real} \to \mathsf{int}$ to something of type real, we get something of type int.
- The type of $x \mapsto -x$ is $\mathsf{int} \to \mathsf{int}$ when it takes integers to integers. It can also be $\mathsf{real} \to \mathsf{real}$, when it takes reals to reals. These are two different functions, because they have different types.
- The type of $x \mapsto x + 3$, when it takes integers, is $\mathsf{int} \to \mathsf{int}$.
- We cannot write $(x \mapsto x + 3)(\pi)$, because the types don't match. $x \mapsto x + 3$ wants an input of type int, but π is of type real.
- Functions can return functions. Think of $+$ as a function that takes an int, like 3, and returns a function like $x \mapsto x + 3$, which has type $\mathsf{int} \to \mathsf{int}$. Concretely, $+$ is $x \mapsto (y \mapsto y + x)$. This has type $\mathsf{int} \to (\mathsf{int} \to \mathsf{int})$.
- We write $+(3)(4) = 7$. First, $+$ has type $\mathsf{int} \to (\mathsf{int} \to \mathsf{int})$, so $+(3)$ has type $\mathsf{int} \to \mathsf{int}$. So, $+(3)(4)$ should have type int.
- As another example, consider $A = y \mapsto (z \mapsto y(z))$. This is "apply". You can write $A(x \mapsto -x) = z \mapsto (x \mapsto -x)(z) = z \mapsto -z$.
- If we have some function f, and we're given that $f(\pi) = 3$, then we can *infer* that f has type $\mathsf{real} \to \mathsf{int}$.
- If we have some function f, and we see $f(\pi)$, we can also infer that f has type $\mathsf{real} \to \alpha$, for some type α.
- If we have some function f, and we see $f(x)$, we can infer that f has type $\alpha \to \beta$ and x has type α, for some types α and β.
- The type of $I = x \mapsto x$ is $\alpha \to \alpha$ for some type α. What is α? It depends on the type of x. We say that the type of this function is *parametrized* over α. Because this function can take different types, this function is *polymorphic*.

- The type of $K = x \mapsto (y \mapsto x)$ is $\alpha \to (\beta \to \alpha)$.
- What is the type of $A = y \mapsto (z \mapsto y(z))$? Say z has type α. Then:
  - From $y(z)$, we infer y takes the type of z, which is α.
  - There are no other restrictions to the output type, so let's say $y(z)$ has type β.
  - This means y itself is $\alpha \to \beta$.
  - This means A itself has type $(\alpha \to \beta) \to (\alpha \to \beta)$.
- What is the type of $S = x \mapsto (y \mapsto (z \mapsto x(z)(y(z))))$? Say z has type α. Then:
  - From $y(z)$, we infer y takes the type of z, which is α.
  - There are no other restrictions to the output type, so let's say $y(z)$ has type β.
  - This means y itself is $\alpha \to \beta$.
  - From $x(z)$, we infer x takes the type of z, which is α.
  - From $x(z)(y(z))$, we infer $x(z)$ takes the type of $y(z)$, which is β.
  - There are no other restrictions, so let's say $x(z)(y(z))$ has type γ.
  - This means $x(z)$ itself has type $\beta \to \gamma$.
  - This means x itself has type $\alpha \to (\beta \to \gamma)$.
  - This means S itself has type $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$.
- This process doesn't always work. As a simple example, what is the type of $x \mapsto x(x)$?
- What is the type of $A(I)$?
  - We see I has type $\alpha \to \alpha$.
  - We see A has type $(\alpha \to \beta) \to (\alpha \to \beta)$. For clarity, we'll use different letters, and say A has type $(A \to B) \to (A \to B)$.
  - In particular, A takes input of type $A \to B$, and outputs type $A \to B$.
  - For the type of I to match the input type of A, we want to match $\alpha \to \alpha$ with $A \to B$. To do this, we take $A = \alpha$ and $B = \alpha$.
  - The output type of A is $A \to B$. So after applying I, the type of $A(I)$ will be $\alpha \to \alpha$.
- What is the type of $S(K)$?
  - We see K has type $\alpha \to (\beta \to \alpha)$.
  - We see S has type $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$.
  - In particular, S takes input of type $A \to (B \to C)$, and outputs type $(A \to B) \to (A \to C)$.
  - For the type of K to match the input type of S, we want to match $\alpha \to (\beta \to \alpha)$ with $A \to (B \to C)$. To do this, we take $A = \alpha$, $B = \beta$, and $C = \alpha$.
  - The output type of S is $(A \to B) \to (A \to C)$. So after applying K, the type of $S(K)$ will be $(\alpha \to \beta) \to (\alpha \to \alpha)$.
- What is the type of $S(K)(K)$?

  - We see K has type $\alpha \to (\beta \to \alpha)$.
  - We see $S(K)$ has type $(A \to B) \to (A \to A)$.
  - For the type of K to match the input type of $S(K)$, we want to match $\alpha \to (\beta \to \alpha)$ with $A \to B$. To do this, we take $A = \alpha$ and $B = \beta \to \alpha$.
  - The output type of $S(K)$ is $A \to A$. So after applying K, the type of $S(K)(K)$ will be $\alpha \to \alpha$.

- In fact, we can rewrite $S(K)(K)(x)$ as x: $S(K)(K)(x) = K(x)(K(x)) = x$.

- If I gave you the proof of A, and a proof of $A \implies B$, you can produce a proof of B. If I gave you a term of type α, and a term of type $\alpha \to \beta$, you can produce a term of type β.

- Our derivation for the type of $S(K)(K)$ can be thought of as a proof, that given the propositional schema
$$\kappa : A \implies (B \implies A)$$
and
$$\sigma : (A \implies (B \implies C)) \implies ((A \implies B) \implies (A \implies C)),$$
we can derive the theorem $A \implies A$. First we apply κ to σ, then we apply κ again. The term $S(K)(K)$ *itself* carries the proof.

- Given the propositional schema κ and σ, we can derive the theorem
$$(B \implies C) \implies ((A \implies B) \implies (A \implies C)).$$
The proof is to consider the type of $S(K(S))(K)$. (Check this!)

- This relationship between type systems and proofs is known as the *Curry–Howard correspondence*, and it forms part of the basis of several versions of type theory.