Introduction to Propositions as Types

Donovan Crichton

March 2024

Preliminaries

Slides and Examples available at:
https://github.com/donovancrichton/Talks
This talk: BFPG/PropositionsAsTypes

About me

PhD Candidate
Computing Foundations
School of Computing

Visiting Scholar
Trusted Systems Lab
IIIS

ASD Co-Lab Scholar

Proving Theorems $\cong$ Writing Programs - Overview

The PAT (Propositions as Types) Interpretation

- Types are logical propositions or theorems.
- An inhabitant or element of the type is the proof.
- Strictly speaking an isomorphism that holds under specific conditions.

Uses for the PAT Interpretation

- Used by mathematicians and logicians to *formalise* or *mechanise* mathematics and logic.
- Used by software developers to provide a stronger guarantee of functional correctness than testing.

Proving Theorems $\cong$ Writing Programs - Applications

Applications

- In defense/security: guarantees for software security.
- In aeronautics and naval agencies: guarantees for software properties that control hardware.
- In fin-tech: guarnatees for software properties that involve transactions.
- Suitable for any application where the cost of a failed test after deployment is just too high.

Propostions - Refresher

What is a proposition?

A statement or assertion that expresses a judgement, usually readily apparent or verifiable.

Examples of a proposition

- "It is raining outside."
- "Terry is the child of Leigh."
- Sometimes formally denoted:
  Let p denote "It is raining outside."

Theorems - Refresher

What is a theorem?

A more general proposition that is less readily-apparent, usually requires a chain of reasoning to be accepted, or 'proven'.

Examples of Theorems

- "If Terry has a child r, and Terry is the child of Leigh, then r is the grandchild of Leigh."
- "For all natural numbers x and y, and given the addition operation $(+)$, then $x + y$ is equal to $y + x$."
- "There is at least one weekday occuring in the future."

Proofs - Refresher

What is a poof?

The chain of formal reasoning that, when followed, always verifies a theorem.

Examples of Proof sketches

- Grandchild is defined as "The child of a person's child." Thus, under this definition, this holds.
- Use a lemma (a smaller proof) to show that $x + 0 = 0 + x$ by definition of $+$. Then apply the induction hypothesis for all other cases.
- This is more complicated. We could use an inductive, discrete definition of "future", and show that after so many iterations a new day occurs.

Types - Refresher

What is a type?

We often think of types in two forms, one computational, and one mathematical.

- A type is a way to interpret the series of bits that represent a value to distinguish it from other values that are encoded with the same bit representation.
- A type is a "set" of elements that inhabit the type.

Examples of Types

- Take the binary number 1100001. Does this represent the positive integer 1,100,001? The letter 'a' on the keyboard? Or the decimal value of 97?
- $\mathbb{B}$ denotes the set of Boolean values: {True, False}.
- $\mathbb{Z}$ denotes the set of Integer values: {..., -2, -1, 0, 1, 2, ...}.

Elements/Inhabitants - Refresher

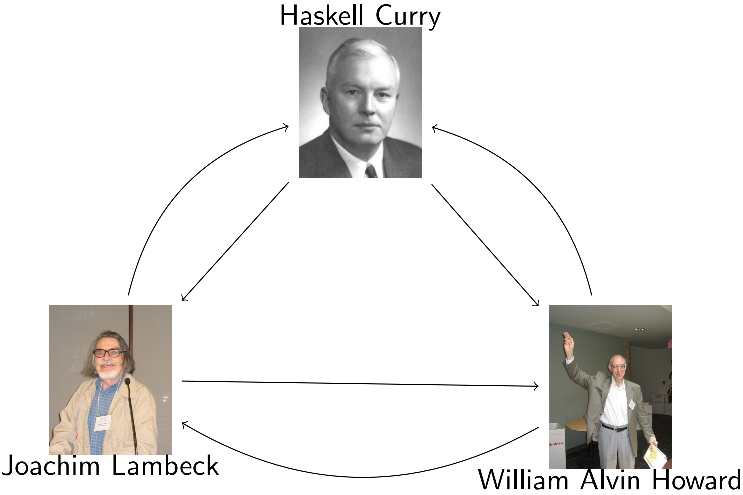
What is an element or inhabitant of a type?

One of the members of the set that characterises a type.

Examples of elements of types.

- Ordinary values: Such as {True,False} from $\mathbb{B}$.
- Data Types:
  {("Nil", $* \to$ List A), ("Cons", A $\to$ List A $\to$ List A)}.
- Codata Types: {("hd", List A $\to$ A), ("tl", List A $\to$ List A)}.

Curry, Howard, And Lambeck - Computational Trinitarianism



Isomorphism

What is an isomorphism?

Often denoted with $\cong$, an isomorphism is a *structure preserving map* between two algebraic structures. Generally this means there is a pair of functions such that their composition is the identity.

An Example of Isomorphism

$\mathbb{B} = \{True, False\}$.
$Bin = \{0, 1\}$.

$f : \mathbb{B} \rightarrow Bin$
$f(True) = 1$
$f(False) = 0$

$g : Bin \rightarrow \mathbb{B}$
$g(0) = False$
$g(1) = True$

Types and Props - Implication

Implication: English and Formalisms

"If p then q" Or "p implies q", $p \to q$, $p \subset q$.

An Example of Implication in Haskell

```haskell
module Implication where
-- Strings imply Integers
f :: String -> Integer
f "zero" = 0
f "one"  = 1
f "two"  = 2
f _      = -1
```

Types and Props - Conjuction

Conjunction: English and Formalisms

"p and q", $p \wedge q$, $p \times$q.

An Example of Implication in Haskell

```haskell
module Conjunction where
type Name = String
type Age = Integer
data Person = Person Name Age
-- Name and Age Implies Personhood
f :: (Name, Age) -> Person
f (s, k) = Person s k
```

Types and Props - Disjunction

Disjunction: English and Formalisms

"p or q", $p \vee q$, $p + q$.

An Example of Implication in Haskell

```
module Disjunction where
type Name = String
type Age = Integer
type Error = String
data Person = Person Name Age
-- Name and Age implies Personhood or Errors.
f :: (Name, Age) -> Either Person Error
f (s, k) =
  case (s == "" || k < 1) of
    True -> Right "Error"
    False -> Left (Person s k)
```

Types and Props - Negation

Types and Props - Quantifiers?

Types and Props - Quantifiers (and Language)

Examples - Linked List Invariants and Vectors

References