Lecture 27: λ-Encoding Propositional Logic

MATH230

Te Kura Pāngarau | School of Mathematics and Statistics
Te Whare Wānanga o Waitaha | University of Canterbury

Outline

Summary

In the previous lecture, motivated by conditional branching, we defined the following λ-expressions to mimic the Boolean values TRUE and FALSE:

$$\text{COND} = \lambda f.\ \lambda g.\ \lambda c.\ ((c\ f)\ g)$$
$$\text{TRUE} = \lambda x.\ \lambda y.\ x$$
$$\text{FALSE} = \lambda x.\ \lambda y.\ y$$
$$\text{NOT} = \text{COND FALSE TRUE}$$

Where COND is a λ-expression which reduces to f if $c = \mathit{TRUE}$ and reduces to g if $c = \mathit{FALSE}$.

You may expand and β-reduce the definition of NOT using the other three definitions above. However, you should perform any appropriate α-reduction to make sure there are no variable name clashes.

Examples

When computing β-reduction on Booleans these redex will arise regularly. So lets simplify them now.

TRUE TRUE

TRUE FALSE

FALSE TRUE

FALSE FALSE

Propositional Connectives

In this lecture we will think about how to define the following propositional connectives and binary/Boolean functions:

AND

OR

IMP(LICATION)

NAND

EQUIV(ALENT)

IsTRUE?

IsFALSE?

AND

AND is a binary function that returns a Boolean

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

TRUE $= \lambda x.\ \lambda y.\ x$

FALSE $= \lambda x.\ \lambda y.\ y$

AND $= \lambda p.\ \lambda q.$

Example

β-reduce the following λ-expression to normal form.

AND TRUE FALSE

OR

OR is a binary function that returns a Boolean

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

TRUE $= \lambda x.\ \lambda y.\ x$

FALSE $= \lambda x.\ \lambda y.\ y$

OR $= \lambda p.\ \lambda q.$

Example

β-reduce the following λ-expression to normal form.

OR FALSE TRUE

IMP

IMPlication is a binary function that returns a Boolean

P	Q	$P \to Q$
T	T	T
T	F	F
F	T	T
F	F	T

$\mathsf{TRUE} = \lambda x.\ \lambda y.\ x$

$\mathsf{FALSE} = \lambda x.\ \lambda y.\ y$

$\mathsf{IMP} = \lambda p.\ \lambda q.$

Example

β-reduce the following λ-expression to normal form.

IMP TRUE FALSE

NAND

NAND is a binary function that returns a Boolean

P	Q	NANDPQ
T	T	F
T	F	T
F	T	T
F	F	T

$\mathsf{TRUE} = \lambda x.\ \lambda y.\ x$

$\mathsf{FALSE} = \lambda x.\ \lambda y.\ y$

$\mathsf{NAND} = \lambda p.\ \lambda q.$

EQUIV

EQUIV is a binary function that returns a Boolean

P	Q	EQUIVPQ
T	T	T
T	F	F
F	T	F
F	F	T

TRUE $= \lambda x.\ \lambda y.\ x$

FALSE $= \lambda x.\ \lambda y.\ y$

EQUIV $= \lambda p.\ \lambda q.$

Example

β-reduce the following λ-expression to normal form.

EQUIV FALSE FALSE

IsTRUE?

IsTRUE? is a unary function that returns a Boolean.

A	IsTRUE?A
T	T
F	F

TRUE $= \lambda x.\ \lambda y.\ x$

FALSE $= \lambda x.\ \lambda y.\ y$

IsTRUE? $=$

IsFALSE?

IsFALSE? is a unary function that returns a Boolean.

A	IsFALSE?A
T	F
F	T

TRUE $= \lambda x.\ \lambda y.\ x$

FALSE $= \lambda x.\ \lambda y.\ y$

IsFALSE? $=$

Propositional Logic Summary

This is a list of encodings for propositional logic into λ-expressions. Note that there are many ways to encode these expressions.

$$\text{COND} = \lambda f.\ \lambda g.\ \lambda c.\ ((c\ f)\ g)$$

$$\text{TRUE} = \lambda x.\ \lambda y.\ x$$

$$\text{FALSE} = \lambda x.\ \lambda y.\ y$$

$$\text{NOT} = \text{COND FALSE TRUE}$$

$$\text{AND} = \lambda p.\ \lambda q.\ (p\ q)\ p$$

$$\text{OR} = \lambda p.\ \lambda q.\ (p\ p)\ q$$

$$\text{IMP} = \lambda p.\ \lambda q.\ \text{OR}\ (\text{NOT}\ p)\ q$$

$$\text{EQUIV} = \lambda p.\ \lambda q.\ (p\ q)(NOT\ q)$$

$$\text{IsTRUE?} = I$$

$$\text{IsFALSE?} = \text{NOT}$$

Further Reading

Here are some recommended reading to follow up on the lecture content. Some are freely available online.

Type Theory and Functional Programming, *Simon Thompson*

- Stanford Encyclopedia Articles:
  1. The Lambda Calculus
  2. Type Theory
  3. Church's Type Theory

Church's original papers are worth visiting, although more work than Turing's paper.
An Unsolvable Problem of Elementary Number Theory, *Alonso Church*

A Note on the Entscheidungsproblem, *Alonso Church*