MATH230: Tutorial Six [Solutions]

Computation and Church Encodings

Key ideas
- Practice β-reduction,
- Encode logic in λ-calculus,
- Encode natural numbers in λ-calculus.
- Encode compound data in λ-calculus.

Relevant topic: Untyped Lambda Calculus Slides
Relevant reading: Type Theory and Functional Programming, Simon Thompson

Hand in exercises: 1b, 2, 3b, 4c, 5b
Due Friday @ 5pm to the submission box on Learn.

Discussion Questions

- Compute the normal forms of the following λ-terms:

  $(\lambda x.\ x\ x)(\lambda y.\ \lambda z.\ y\ z\ z)$

$$
\begin{aligned}
&(\lambda x.\ x\ x)(\lambda y.\ \lambda z.\ y\ z\ z)\\
&=_\beta (\lambda y.\ \lambda z.\ y\ z\ z)(\lambda y.\ \lambda z.\ y\ z\ z)\\
&=_\alpha (\lambda y.\ \lambda w.\ y\ w\ w)(\lambda y.\ \lambda z.\ y\ z\ z)\\
&=_\beta \lambda w.\ (\lambda y.\ \lambda z.\ y\ z\ z)\ w\ w\\
&=_\beta \lambda w.\ (\lambda z.\ w\ z\ z)\ w\\
&=_\beta \lambda w.\ w\ w\ w
\end{aligned}
$$

  AND TRUE TRUE

$$
\begin{aligned}
&\text{AND TRUE TRUE}\\
&=_{\mathsf{DS}} (\lambda p.\ \lambda q.\ p\ q\ q)\ \text{TRUE TRUE}\\
&=_\beta (\lambda q.\ \text{TRUE}\ q\ q)\ \text{TRUE}\\
&=_\beta \text{TRUE TRUE TRUE}\\
&=_{\mathsf{DS}} (\lambda x.\ \lambda y.\ x)\ \text{TRUE TRUE}\\
&=_\beta (\lambda y.\ \text{TRUE})\ \text{TRUE}\\
&=_\beta \text{TRUE}
\end{aligned}
$$

Tutorial Exercises

1. Compute the normal form for each of the following λ-terms:

   (a) NOT FALSE

   Solution:

$$\begin{aligned}\text{NOT FALSE} &\equiv (\lambda p.\ p\ \text{FALSE TRUE})\ \text{FALSE}\\ &=_\beta \text{FALSE FALSE TRUE}\\ &= (\lambda p.\ \lambda q.\ q)\ \text{FALSE TRUE}\\ &=_\beta (\lambda q.\ q)\ \text{TRUE}\\ &=_\beta \text{TRUE}\end{aligned}$$

   (b) OR TRUE FALSE

   Solution:

$$\begin{aligned}\text{OR TRUE FALSE} &= (\lambda p.\ \lambda q.\ p\ p\ q)\ \text{TRUE FALSE}\\ &=_\beta \text{TRUE TRUE FALSE}\\ &= (\lambda p.\ \lambda q.\ p)\text{TRUEFALSE}\\ &=_\beta (\lambda q.\ \text{TRUE})\ \text{FALSE}\\ &=_\beta \text{TRUE}\end{aligned}$$

   (c) AND FALSE TRUE

   Solution:

$$\begin{aligned}\text{AND FALSE TRUE} &= (\lambda p.\ \lambda q.\ p\ q\ p)\ \text{FALSE TRUE}\\ &=_\beta \text{FALSE TRUE FALSE}\\ &= (\lambda p.\ \lambda q.\ q)\ \text{TRUE FALSE}\\ &=_\beta (\lambda q.\ q)\ \text{FALSE}\\ &=_\beta \text{FALSE}\end{aligned}$$

(d) IMPLIES FALSE TRUE

Solution:

$$
\begin{aligned}
\text{IMPLIES FALSE TRUE} &= (\lambda p.\ \lambda q.\ p\ q\ (\text{NOT}\ p))\ \text{FALSE TRUE} \\
&=_\beta \text{FALSE TRUE (NOT FALSE)} \\
&= (\lambda p.\ \lambda q.\ q)\ \text{TRUE (NOT FALSE)} \\
&=_\beta (\lambda q.\ q)\ (\text{NOTFALSE}) \\
&=_\beta \text{NOT FALSE} \\
&= (\lambda p.\ p\ \text{FALSE TRUE})\ \text{FALSE} \\
&=_\beta \text{FALSE FALSE TRUE} \\
&= (\lambda p.\ \lambda q.\ q)\ \text{FALSE TRUE} \\
&=_\beta (\lambda q.\ q)\ \text{TRUE} \\
&=_\beta \text{TRUE}
\end{aligned}
$$

2. Write down λ-expressions that represent the propositional binary connectives XOR, NAND, and NOR. Recall that these have the following truth tables.

P	Q	XOR(P,Q)
T	T	F
T	F	T
F	T	T
F	F	F

P	Q	NAND(P,Q)
T	T	F
T	F	T
F	T	T
F	F	T

P	Q	NOR(P,Q)
T	T	F
T	F	F
F	T	F
F	F	T

Solution:

TRUE and FALSE are defined as selectors of two inputs. For this reason it's helpful to think about what the binary-connective must return if the first input is TRUE, and then what it must return if the first input is FALSE. All binary connectives can be written in the following form where the first input is used as a selector:

$$\lambda p.\ \lambda q.\ p\ \text{«EXP-TRUE»}\ \text{«EXP-FALSE»}$$

If the first input is TRUE, then the binary connective will return EXP-TRUE, whereas if the first input is FALSE, then the binary connective will return EXP-FALSE.

XOR

If the first input is TRUE, then the truth table shows that the XOR connective should return the negation of the second input. Whereas, if the first input is FALSE, the XOR connective should return the second input.

$$\text{XOR} :\equiv \lambda p.\ \lambda q.\ p\ (\text{NOT}\ q)\ q$$

NAND

If the first input is TRUE, then the truth table shows that the NAND connective should return the negation of the second input. Whereas, if the first input is FALSE, then NAND connective should return TRUE. Equivalently, NAND should return the negation of the first input in this case.

$$\text{NAND} :\equiv \lambda p.\ \lambda q.\ p\ (\text{NOT}\ q)\ \text{TRUE}$$

NOR

Alternatively, one can write compound connectives using the λ-terms of their component parts. NOR is, by definition, the negation of OR. So we can define the λ-term:

$$\text{NOR} :\equiv \lambda p.\ \lambda q.\ \text{NOT}\ (\text{OR}\ p\ q)$$

It is interesting to note that these two approaches don't necessarily yield β-equivalent λ-terms for the propositional connectives. However, they are equivalent in the sense that they agree on Boolean inputs. That is, they are *extensionally* equivalent.

3. By substituting the explicit λ-expressions (as necessary) and performing β-reduction, determine the normal forms of the following λ-expressions.

   (a) SUCC ONE

   Solution:

$$\begin{aligned}
&\text{SUCC ONE}\\
&\quad = (\lambda n.\ \lambda u.\ \lambda v.\ u\ (n\ u\ v))\ \text{ONE}\\
&\quad =_\beta \lambda u.\ \lambda v.\ u\ (\text{ONE}\ u\ v)\\
&\quad = \lambda u.\ \lambda v.\ u\ ((\lambda s.\ \lambda x.\ s\ x)\ u\ v)\\
&\quad =_\beta \lambda u.\ \lambda v.\ u\ (u\ v)\\
&\quad =_\alpha\ \text{TWO}
\end{aligned}$$

   (b) SUM ONE TWO

   Solution:

$$\begin{aligned}
&\text{SUM ONE TWO}\\
&\quad = (\lambda m.\ \lambda n.\ \lambda u.\ \lambda v.\ m\ u\ (n\ u\ v))\ \text{ONE TWO}\\
&\quad =_\beta \lambda u.\ \lambda v.\ \text{ONE}\ u\ (\text{TWO}\ u\ v)\\
&\quad = \lambda u.\ \lambda v.\ \text{ONE}\ u\ ((\lambda s.\ \lambda x.\ s\ (s\ x))\ u\ v)\\
&\quad =_\beta \lambda u.\ \lambda v.\ \text{ONE}\ u\ (u\ (u\ v))\\
&\quad = \lambda u.\ \lambda v.\ (\lambda s.\ \lambda x.\ s\ x)\ u\ (u\ (u\ v))\\
&\quad =_\beta \lambda u.\ \lambda v.\ u\ (u\ (u\ v))\\
&\quad =_\alpha \text{THREE}
\end{aligned}$$

   (c) MULT TWO THREE

   Solution:

$$\begin{aligned}
&\text{MULT TWO THREE}\\
&\quad = (\lambda m.\ \lambda n.\ \lambda u.\ \lambda v.\ m\ (n\ u)\ v)\ \text{TWO THREE}\\
&\quad =_\beta \lambda u.\ \lambda v.\ \text{TWO}\ (\text{THREE}\ u)\ v\\
&\quad = \lambda u.\ \lambda v.\ (\lambda s.\ \lambda x.\ s\ (s\ x))\ (\text{THREE}\ u)\ v\\
&\quad =_\beta \lambda u.\ \lambda v.\ (\text{THREE}\ u)\ (\text{THREE}\ u\ v)\\
&\quad = \lambda u.\ \lambda v.\ (\text{THREE}\ u)\ ((\lambda s.\ \lambda x.\ s\ (s\ (s\ x)))\ u\ v)\\
&\quad =_\beta \lambda u.\ \lambda v.\ (\text{THREE}\ u)\ (u\ (u\ (u\ v)))\\
&\quad = \lambda u.\ \lambda v.\ ((\lambda s.\ \lambda x.\ s\ (s\ (s\ x)))\ u)\ (u\ (u\ (u\ v)))\\
&\quad =_\beta \lambda u.\ \lambda v.\ (\lambda x.\ u\ (u\ (u\ x)))\ (u\ (u\ (u\ v)))\\
&\quad =_\beta \lambda u.\ \lambda v.\ u\ (u\ (u\ (u\ (u\ v)))))\\
&\quad =_\alpha \text{SIX}
\end{aligned}$$

4. We have defined the following λ-expression to construct pairs of λ-expressions:

$$\mathsf{PAIR} = \lambda x.\ \lambda y.\ \lambda f.\ f\ x\ y$$

The third input is a built-in place ready to take a selector:

$$\mathsf{FirST} = \lambda x.\ \lambda y.\ x \qquad \mathsf{SecoND} = \lambda x.\ \lambda y.\ y$$

Reduce these to normal form

(a) PAIR a b fst

Solution:

$$\begin{aligned}
&\mathsf{PAIR}\ a\ b\ \mathsf{fst} \\
&\quad = (\lambda x.\ \lambda y.\ \lambda f.\ f\ x\ y)\ a\ b\ \mathsf{fst} \\
&\quad =_\beta (\lambda f.\ a\ b)\ \mathsf{fst} \\
&\quad =_\beta \ \mathsf{fst}\ \ a\ b \\
&\quad = (\lambda x.\ \lambda y.\ x)\ a\ b \\
&\quad =_\beta (\lambda y.\ a)\ b \\
&\quad =_\beta a
\end{aligned}$$

(b) PAIR a b snd

Solution:

$$\begin{aligned}
&\mathsf{PAIR}\ a\ b\ \mathsf{snd} \\
&\quad = (\lambda x.\ \lambda y.\ \lambda f.\ f\ x\ y)\ a\ b\ \mathsf{snd} \\
&\quad =_\beta (\lambda f.\ a\ b)\ \mathsf{snd} \\
&\quad =_\beta \ \mathsf{snd}\ \ a\ b \\
&\quad = (\lambda x.\ \lambda y.\ y)\ a\ b \\
&\quad =_\beta (\lambda y.\ y)\ b \\
&\quad =_\beta b
\end{aligned}$$

(c) PAIR (PAIR a b) (PAIR c d) snd

Solution:

$$\begin{aligned}
&\mathsf{PAIR}\ (\mathsf{PAIR}\ a\ b)\ (\mathsf{PAIR}\ c\ d)\ \mathsf{snd} \\
&\quad = (\lambda x.\ \lambda y.\ \lambda f.\ f\ x\ y)\ (\mathsf{PAIR}\ a\ b)\ (\mathsf{PAIR}\ c\ d)\ \mathsf{snd} \\
&\quad =_\beta (\lambda f.\ (\mathsf{PAIR}\ a\ b)\ (\mathsf{PAIR}\ c\ d))\ \mathsf{snd} \\
&\quad =_\beta \ \mathsf{snd}\ \ (\mathsf{PAIR}\ a\ b)\ (\mathsf{PAIR}\ c\ d) \\
&\quad = (\lambda x.\ \lambda y.\ y)\ (\mathsf{PAIR}\ a\ b)\ (\mathsf{PAIR}\ c\ d) \\
&\quad =_\beta (\lambda y.\ y)\ (\mathsf{PAIR}\ c\ d) \\
&\quad =_\beta (\mathsf{PAIR}\ c\ d)
\end{aligned}$$

5. Positive rational numbers are solutions to equations of the form $bx = a$, where $a, b : \mathbb{N}$. Use PAIR to represent positive rational numbers in the λ-calculus and write λ-expressions to compute rational number arithmetic.

   Note that a rational number such as $1/2$ is just a pair of natural numbers $(1, 2)$. Therefore arithmetic on rational numbers can be defined by arithmetic on pairs of natural numbers. Additional of rational numbers is done by cross multiplication: $(a, b) + (c, d) = (ad + bc, dc)$. Multiplication of rational numbers is done by multiplying numerators and denominators: $(a, b) \times (c, d) = (ac, bd)$. Taking the reciprocal amounts to swapping the terms in the pair: $(a, b) \mapsto (b, a)$. This implementation ignores the issue of not allowing 0 on the denominator.

   With this implementation of the rational numbers, we can write lambda expressions to compute the arithmetic of rational numbers.

   (a) RAT-SUM to calculate the sum of two rational numbers.

$$\begin{aligned}\text{RAT-SUM} :\equiv \text{PAIR } (\text{SUM } (\text{MULT } (\text{fst } \ x) \ (\text{snd } \ y)) \\ (\text{MULT } (\text{snd } \ x) \ (\text{fst } \ y))) \\ (\text{MULT } (\text{snd } \ x) \ (\text{snd } \ y))\end{aligned}$$

   (b) RAT-MULT to calculate the product of two rational numbers.

$$\begin{aligned}\text{RAT-MUL} :\equiv \lambda x.\ \lambda y.\ \text{PAIR } (\text{MULT } (\text{fst } \ x) \ (\text{fst } \ y)) \\ (\text{MULT } (\text{snd } \ x) \ (\text{snd } \ y))\end{aligned}$$

   (c) RAT-REC(iprocal) to calculate the reciprocal of an integer.

$$\text{RAT-REC} :\equiv \lambda x.\ \text{PAIR } (\text{snd } \ x) \ (\text{fst } \ x)$$