MATH230: Tutorial Eleven

Solutions

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

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

Hand in exercises: 1c, 2, 3a, 5
Due following Friday @ 5pm.

Discussion Questions

- Logical connectives are encoded as functions in the λ-calculus. They're intended to take in the Boolean λ-expressions TRUE and FALSE. However, without any extra structure to the λ-calculus, there is nothing stopping us writing:

  NOT NOT NOT

  Perform β-reduction on the above until the expression is in normal form. The strange answer you get suggests that we should not do this! To avoid this, extra *type* structure is added to the λ-calculus; this amounts to saying only certain λ-expressions can be applied to others. If the language were typed sensibly, we would get a type error when trying to evaluate NOT NOT NOT.

  Solution:

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

Tutorial Exercises

1. By substituting the explicit λ-expressions (as necessary) and performing β-reduction, show that the expressions below are β-equivalent to the Boolean values we should expect from the logical connectives involved.

   Hint: You do not need to desugar each term at the start. Only as you need them.

   (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 following propositional binary connectives.

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.

   Only expand those expressions necessary for each step.

   (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:

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

Compound Data with PAIR

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. Integers are solutions to equations $x+b=a$. We can represent natural number solutions using Church numerals. However, further abstractions are required to represent the negative solutions to such equations.

   Example The equation $x+1=0$ has solution $x=-1$. One way to represent this in the λ-calculus is to use the pair $(0,1)$ which we interpret as $-1=(0,1)$.

   More generally $k=(a,b)=a-b$ represents the solution to $x+b=a$.

   Such representations are not unique! e.g. $0=(0,0)=(1,1)=\ldots$

   Write λ-expressions for arithmetic on integers as pairs of Church numerals.

   INT-SUM to calculate the sum of two integers.

   Solution: As we are representing integers as pairs of natural numbers we write an abstraction with the understanding that pairs will be used as input. The following lambda expression constructs the pair representing the sum of two integers.

   INT-SUM $\equiv \lambda m.\ \lambda n.$ PAIR (SUM (FST m) (FST n)) (SUM (SND m) (SND n))

   INT-MULT to calculate the product of two integers.

   Solution: If $a-b\equiv(a,b)$ and $(c-d)\equiv(c,d)$, then in order to find the correct expression for multiplication of pairs we can calculate $(a-b)(c-d)$ and collect the positive and negative parts:

   $$(a-b)(c-d)=(ac+bd)-(ad+bc)$$

   It follows that when integers are represented as pairs of natural numbers we can define integer multiplication as:

   $$(a,b)(c,d)=(ac+bd,ad+bc)$$

   INT-MULT $\equiv \lambda m.\ \lambda n.$ PAIR
   (SUM (MULT (FST m) (FSTn)) (MULT (SND m) (SND n)))
   (SUM (MULT (FST m) (SNDn)) (MULT (SND m) (FST n)))

   INT-NEG(ative) to calculate the negative of an integer.

   Solution:

   INT-NEG $\equiv \lambda m.$ PAIR (SND m) (FST m)

6. Rational numbers are solutions to equations of the form $bx = a$. Use PAIR to represent rational numbers in the λ-calculus and write λ-expressions to compute rational number arithmetic.

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

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

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

7. Imaginary numbers can be represented as PAIRs of rational numbers.

8. Real numbers are a more complicated issue.