Informatics 2A: Tutorial Sheet 8 Solutions

Shay Cohen[1]

1. The question is somewhat open-ended, but one solution is as follows. We consider the following three attributes:

Person:	values 1,2,3, ranged over by x
Number:	values s,p, ranged over by y
Gender:	values m,f,n, ranged over by z

The following parameterized rules suffice:

S	$\rightarrow$	SPro[x,y,z] Vstem Vsuf[x,y] ReflPro[x,y,z]
Vstem	$\rightarrow$	*love* | *prepare* | *congratulate*
SPro[1,s,z]	$\rightarrow$	*I*
SPro[1,p,z]	$\rightarrow$	*We*
SPro[2,y,z]	$\rightarrow$	*You*
SPro[3,s,m]	$\rightarrow$	*He*
SPro[3,s,f]	$\rightarrow$	*She*
SPro[3,s,n]	$\rightarrow$	*It*
SPro[3,p,z]	$\rightarrow$	*They*
Vsuf[3,s]	$\rightarrow$	*-s*
Vsuf[1,s]	$\rightarrow$	ϵ, etc.
ReflPro[1,s,z]	$\rightarrow$	*myself*
ReflPro[2,s,z]	$\rightarrow$	*yourself*
ReflPro[3,s,m]	$\rightarrow$	*himself*
ReflPro[3,s,f]	$\rightarrow$	*herself*
ReflPro[3,s,n]	$\rightarrow$	*itself*
ReflPro[1,p,z]	$\rightarrow$	*ourselves*
ReflPro[2,p,z]	$\rightarrow$	*yourselves*
ReflPro[3,p,z]	$\rightarrow$	*themselves*

2. We want a single constant, Jumbo, and predicates with arities as follows:

elephant/1 mammal/1 owns/2 song/1 sings_to/3 danced/1

We also have a binary equality predicate as a standard part of our FOPL machinery.

The given sentences may be translated into FOPL as follows:

- *Jumbo is an elephant*: elephant(Jumbo)
- *An elephant is a mammal*: this could arguably have either of the following meanings.

$$\forall x.\,\mathrm{elephant}(x) \Rightarrow \mathrm{mammal}(x)$$

$$\exists x.\,\mathrm{elephant}(x) \wedge \mathrm{mammal}(x)$$

[1] Thanks to Michael Herrmann for providing the solution for Problem 3.

- *Every elephant has an owner*:

$$\forall x.\, \text{elephant}(x) \Rightarrow \exists y.\, \text{owns}(y, x)$$

or possibly

$$\exists y. \forall x.\, \text{elephant}(x) \Rightarrow \text{owns}(y, x)$$

- *Everyone who owns an elephant sings it a song*:

$$\forall x. \forall y.\, (\text{owns}(x, y) \wedge \text{elephant}(y)) \Rightarrow \exists z.\, (\text{song}(z) \wedge \text{sings\_to}(x, z, y))$$

or possibly

$$\forall x. \forall y. \exists z.\, (\text{owns}(x, y) \wedge \text{elephant}(y)) \Rightarrow (\text{song}(z) \wedge \text{sings\_to}(x, z, y))$$

- *Only one elephant danced.*

$$\exists x.\, \text{elephant}(x) \wedge \text{danced}(x) \wedge (\forall y.\, (\text{elephant}(y) \wedge \text{danced}(y)) \Rightarrow y = x)$$

- *Every elephant did not dance*: there is a scoping ambiguity here (cf. the expression *All is not lost.*) Either

$$\neg(\forall x.\, \text{elephant}(x) \Rightarrow \text{danced}(x))$$

or

$$\forall x.\, \text{elephant}(x) \Rightarrow \neg\text{danced}(x)$$

3. The rules that take as an argument variables called P or Q are type-raising rules. Type raising refers to the case in which functions themselves are fed as an argument to a higher-order function.

   The semantics of the given phrases are as follows. For the first two examples, we show the β-reductions steps; for the others, we show only the result after β-reduction.

   - John runs:

   $$(\lambda\text{P.P(John)})(\lambda\text{x.run(x)}) \rightarrow_\beta (\lambda\text{x.run(x)})(\text{John}) \rightarrow_\beta \text{run(John)}$$

   - likes ice-cream:

   $$\begin{gathered}\lambda\text{x.}(\lambda\text{P.P(Ice-cream)})((\lambda\text{x.}\lambda\text{y.like(x,y)})(\text{x})) \\ \rightarrow_\beta \lambda\text{x.}(\lambda\text{P.P(Ice-cream)})(\lambda\text{y.like(x,y)}) \\ \rightarrow_\beta \lambda\text{x.}(\lambda\text{y.like(x,y)})(\text{Ice-cream}) \rightarrow_\beta \lambda\text{x. like(x,Ice-cream)}\end{gathered}$$

   - John likes ice-cream: like(John,Ice-cream)
   - an ice-cream: λP.$\exists$x. ice-cream(x) $\wedge$ P(x)
   - likes an ice-cream: λy.$\exists$x. ice-cream(x) $\wedge$ like(y,x)
   - John likes an ice-cream: $\exists$x. ice-cream(x) $\wedge$ like(John,x)
   - every cat: λP.$\forall$x. cat(x) $\Rightarrow$ P(x)
   - every cat likes ice-cream: $\forall$x. cat(x) $\Rightarrow$ like(cat,Ice-cream)
   - every cat likes an ice-cream: $\forall$x. cat(x) $\Rightarrow$ $\exists$y.ice-cream(y)$\wedge$like(x,y)

Informatics 2A: Tutorial Sheet 8 Solution of problem 3

S → NP VP	{ NP.Sem(VP.Sem) }	t
VP → IV	{ IV.Sem }	$< e, t >$
VP → TV NP	{λx.NP.Sem(TV.Sem(x)) }	$< e, t >$
NP → Det N	{ Det.Sem(N.Sem) }	$<< e, t >, t >$
NP → John	{λP.P(John) }	$<< e, t >, t >$
NP → ice-cream	{λP.P(Ice-cream) }	$<< e, t >, t >$
Det → a | an	{λQ.λP.∃x.Q(x) ∧ P(x) }	$<< e, t >,<< e, t >, t >>$
Det → every	{λQλP.∀x.Q(x) ⇒ P(x) }	$<< e, t >,<< e, t >, t >>$
N → cat	{λx.cat(x) }	$< e, t >$
N → ice-cream	{λx.ice-cream(x) }	$< e, t >$
IV → runs	{λx.run(x) }	$< e, t >$
TV → likes	{λx.λy.like(x,y) }	$< e,< e, t >>$

Remember β reduction: (λx.t) (s) reduces to the term t[x := s], i.e. any occurance of x in t is substituted by s.

---

a) phrase: John runs

parse tree: (S(NP John)(VP(IV runs)))

semantics: NP.Sem(VP.Sem) → (λP.P(John))(λx.run(x)) (note that there is a different arrow now)

$\rightarrow_\beta$ (λx.run(x))(John) $\rightarrow_\beta$ run(John)

---

b) phrase: likes ice-cream

parse tree: VP((TV likes)(NP ice-cream))

semantics: The semantics of VP tells us how to proceed:

λx.NP.Sem(TV.Sem(x))

We can replace the x in TV.Sem by a z to avoid a clash of variable and insert the semantics (for simplicity this is not always done in the other examples below):

→λx.(λP.P(Ice-cream))((λz.λy.like(z,y))(x))

Startig from left λz is applied :

$\rightarrow_\beta$ λx.(λP.P(Ice-cream))(λy.like(x,y))

And then, just as in the previous example

$\rightarrow_\beta$ λx.(λy.like(x,y))(Ice-cream) $\rightarrow_\beta$ λx.(like(x.Ice-cream)

c) phrase: John likes ice-cream

parse tree: (S(NP John)(VP(TV likes)(NP ice-cream)))

semantics: NP.Sem(VP.Sem) $\rightarrow$ (λP.P(John))(λx.NP.Sem(TV.Sem(x)))

$\rightarrow$ λP.P(John))(λx.(λP.P(Ice-cream))((λz.λy.like(z,y))(x))

$\rightarrow$ λP.P(John))((λP.P(Ice-cream))((λx.λy.like(x,y)))

$\rightarrow_\beta$ (λP.P(Ice-cream))(λx.λy.like(x,y)))(John)

$\rightarrow_\beta$ (λP.P(Ice-cream))(λy.like(John,y)

$\rightarrow_\beta$ λy.like(John,y)(Ice-cream)

$\rightarrow_\beta$ like(John,Ice-cream)

---

d) phrase: an ice-cream

parse tree: (NP(Det an)(N ice-cream))

semantics: Det.Sem(N.Sem) $\rightarrow$(λQ.λP.$\exists$x.Q(x) $\wedge$ P(x))(λx.ice-cream(x))

$\rightarrow_\beta$ λP.$\exists$x.λx.ice-cream(x) $\wedge$ P(x)

$\rightarrow_\beta$ λP.$\exists$x.ice-cream(x) $\wedge$ P(x)

---

e) phrase: likes an ice-cream

parse tree: (VP(TV likes)(NP(Det an)(N ice-cream)))

semantics: λx.NP.Sem(TV.Sem(x)) $\rightarrow$ λx.(Det.Sem(N.Sem))(λx.λy.like(x,y)(x))

$\rightarrow$$\lambda$x.(($\lambda$Q.$\lambda$P.$\exists$s.Q(s) $\wedge$ P(s))(λw.ice-cream(w)))(λz.λy.like(z,y)(x))

$\rightarrow_\beta$ λx.((λP.$\exists$s.(λw.ice-cream(w)(s)) $\wedge$ P(s)))(λz.λy.like(z,y)(x))

$\rightarrow_\beta$ λx.((λP.$\exists$s.ice-cream(s) $\wedge$ P(s)))(λz.λy.like(z,y)(x))

$\rightarrow_\beta$ λx.(λP.$\exists$s.ice-cream(s) $\wedge$ P(s))(λy.like(x,y))

$\rightarrow_\beta$ λx.($\exists$s.ice-cream(s) $\wedge$ (λy.like(x,y))(s))

λx.($\exists$s.ice-cream(s) $\wedge$ like(x,s))

[note again that disambiguation of the variables is important]

---

f) phrase: John likes an ice-cream

parse tree: (S(NP John)(VP(TV likes)(NP(Det an)(N ice-cream))))

semantics: NP.Sem(VP.Sem) $\rightarrow$ λP.P(John)(λz.NP.Sem(TV.Sem(z)))

$\rightarrow$ λP.P(John)(λz.(Det.Sem(N.Sem))(λx.λy.like(x,y)(z)))

$\rightarrow$ λP.P(John)(λz.((λQ.λP.$\exists$x.Q(x) $\wedge$ P(x))(λw.ice-cream(w)))(λx.λy.like(x,y)(z)))

$\rightarrow_\beta$ (λz.((λQ.λP.$\exists$x.Q(x) $\wedge$ P(x))(λw.ice-cream(w)))(λx.λy.like(x,y)(z)))(John)

$\rightarrow_\beta$ ((λQ.λP.$\exists$x.Q(x) $\wedge$ P(x))(λw.ice-cream(w)))(λx.λy.like(x,y)(John))

$\rightarrow_\beta$ ((λQ.λP.$\exists$x.Q(x) $\wedge$ P(x))(λw.ice-cream(w)))(λy.like(John,y))

$\rightarrow_\beta$ ((λP.$\exists$x.(λw.ice-cream(w))(x) $\wedge$ P(x))(λy.like(John,y))

$\rightarrow_\beta$ ((λP.$\exists$x.ice-cream(x) $\wedge$ P(x))(λy.like(John,y))

$\rightarrow_\beta$ $\exists$x.ice-cream(x) $\wedge$ (λy.like(John,y))(x)

$\rightarrow_\beta$ $\exists$x.ice-cream(x) $\wedge$ like(John,x)

---

g) phrase: every cat

parse tree: (NP(Det every)(N cat))

semantics: Det.Sem(N.Sem) $\rightarrow$(λQλP.$\forall$x.Q(x)) $\Rightarrow$ P(x))(λx.cat(x))

$\rightarrow_\beta$ λP.$\forall$x.(λx.cat(x)) $\Rightarrow$ P(x)

$\rightarrow_\beta$ λP.$\forall$x.cat(x) $\Rightarrow$ P(x)

---

h) phrase: every cat likes ice-cream

parsing tree: (S(NP(Det every)(N cat))(VP(TV likes)(NP Ice-cream)))

semantics: from the first level of the semantic tree and using (g) and (b) we have

NP.Sem(VP.Sem) $\rightarrow$ ((λP.$\forall$x.cat(x)) $\Rightarrow$ P(x))(λx.(like(x.Ice-cream)))

$\rightarrow_\beta$ $\forall$x.cat(x) $\Rightarrow$ (λx.like(x,Ice-cream)(x))

$\rightarrow_\beta$ $\forall$x. cat(x) $\Rightarrow$ like(x,Ice-cream) [note that this is not exaclty the given solution, but equivalent]

---

i) phrase: every cat likes an ice-cream

parse tree: (S(NP(Det every)(N cat))(VP(TV likes)(NP(Det an)(N ice-cream))))

semantics: NP.Sem(VP.Sem) $\rightarrow$ (Det.Sem(N.Sem))(λz.NP.Sem(TV.Sem(z)))

$\rightarrow$ (Det.Sem(N.Sem))(λz.(Det.Sem(N.Sem))(λx.λy.like(x,y)(z)))

$\rightarrow$ ((λQλP.$\forall$x.Q(x)) $\Rightarrow$ P(x))(λx.cat(x))(λz.((λQ.λP.$\exists$y.Q(y)$\wedge$P(y))(λw.ice-cream(w)))(λr.λs.like(r,s)(z)))

$\rightarrow_\beta$ (λP.$\forall$x.(λx.cat(x))(x) $\Rightarrow$ P(x))(λz.((λQ.λP.$\exists$y.Q(y)$\wedge$P(y))(λw.ice-cream(w)))(λr.λs.like(r,s)(z)))

$\rightarrow_\beta$ (λP.$\forall$x.cat(x) $\Rightarrow$ P(x))(λz.((λQ.λP.$\exists$y.Q(y)$\wedge$P(y))(λw.ice-cream(w)))(λr.λs.like(r,s)(z)))

$\rightarrow_\beta$ (λP.$\forall$x.cat(x) $\Rightarrow$ P(x))(λz.((λP.$\exists$y.(λw.ice-cream(w))(y))$\wedge$P(y))(λr.λs.like(r,s)(z)))

$\rightarrow_\beta$ (λP.$\forall$x.cat(x) $\Rightarrow$ P(x))(λz.((λP.$\exists$y.ice-cream(y))$\wedge$P(y)))(λr.λs.like(r,s)(z))

$\rightarrow_\beta$ (λP.$\forall$x.cat(x) $\Rightarrow$ P(x))(λz.(($\exists$y.ice-cream(y))$\wedge$(λr.λs.like(r,s)(z)(y))

$\rightarrow_\beta$ $\forall$x.cat(x) $\Rightarrow$ (λz.(($\exists$y.ice-cream(y))$\wedge$(λr.λs.like(r,s)(z))(y))(x)

$\rightarrow_\beta$ $\forall$x.cat(x) $\Rightarrow$ λz.(($\exists$y.ice-cream(y))$\wedge$(λs.like(z,s))(y))(x)

$\rightarrow_\beta$ $\forall$x.cat(x) $\Rightarrow$ λz.(($\exists$y.ice-cream(y))$\wedge$like(z,y))(x)

$\rightarrow_\beta$ $\forall$x.cat(x) $\Rightarrow$ $\exists$y.ice-cream(y)$\wedge$like(x,y)

... seems quite a lot of work, but most is just cut&paste&edit, you don't need to do every step separately and large parts repeat across the examples which I did not use (except in h). And, sorry for any missing or superfluous brackets, for typos etc.