Exercise 1

<table>
<thead>
<tr>
<th>Subformula</th>
<th>Free variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (((Qx) \lor \exists y.((P z x) \land (Q a))) \lor \forall z.(R x z x)))</td>
<td>{x, z}</td>
</tr>
<tr>
<td>2 (((Qx) \lor \exists y.((P z x) \land (Q a))))</td>
<td>{x, z}</td>
</tr>
<tr>
<td>3 ((Qx))</td>
<td>{x}</td>
</tr>
<tr>
<td>4 \exists x.\forall y.((P z x) \land (Q a))</td>
<td>{z}</td>
</tr>
<tr>
<td>5 \forall y.((P z x) \land (Q a))</td>
<td>{z, x}</td>
</tr>
<tr>
<td>6 ((P z x) \land (Q a))</td>
<td>{z, x}</td>
</tr>
<tr>
<td>7 ((P z x))</td>
<td>{z, x}</td>
</tr>
<tr>
<td>8 ((Q a))</td>
<td>{}</td>
</tr>
<tr>
<td>9 \forall z.(R x z x)</td>
<td>{x}</td>
</tr>
<tr>
<td>10 ((R x z x))</td>
<td>{x, z}</td>
</tr>
</tbody>
</table>

Points to note:

- Every formula is a subformula of itself.
- If \(Q\) is a quantifier, then \(\phi\) is a subformula of \(Q x. \phi\); however, \(Qx\) is not a subformula of \(Q x. \phi\) so the question of \(x\) being free in \(Q x. \phi\) does not arise.

Exercise 2

We had the following formulas (for better readability, expressed in relational rather than functional form):

1. \(\forall x.\exists y.\text{neighbour}(x, y)\)
2. \(\forall x.\neg\text{neighbour}(x, x)\)
3. \(\forall x.\forall y.\forall z. (\text{neighbour}(x, y) \land \text{neighbour}(y, z)) \rightarrow \text{neighbour}(x, z)\)

Although it can be helpful to use the NLTK style models for exploring truth conditions, we can also look at the argument more abstractly.

Let’s assume that there are exactly two individuals in the model, named \(a\) and \(b\). From (1) we can infer (i) and (ii) in the table below. From (2) we can infer (iii) and (iv). Given that we have only two individuals, in (3) we replace \(x\) by \(a\), \(y\) by \(b\) and \(z\) by \(a\) again. Now the antecedent of the conditional is true, but the consequent contradicts (iii).
(i) \( \text{neighbour}(a, b) \)
(ii) \( \text{neighbour}(b, a) \)
(iii) \( \neg\text{neighbour}(a, a) \)
(iv) \( \neg\text{neighbour}(b, b) \)
(v) \( \text{neighbour}(a, b) \land \text{neighbour}(b, a) \rightarrow \text{neighbour}(a, a) \)

More generally, assume we have a finite, non-empty domain \( D = \{d_1, \ldots, d_n\} \), and draw a graph in which edges correspond to the \( \text{neighbour} \) relation. At some point, we add the edge \( (d_{n-1}, d_n) \) to the graph, where \( d_n \) is the last element of \( D \). By virtue of (1), we need to add \( (d_n, d_i) \), for some \( d_i, i \leq n \) which is already in the graph. So we have a loop. Now by repeated appeal to (3), we can conclude that \( (d_i, d_n) \) is in the graph, so again by (3), \( (d_i, d_i) \) is in the graph, contradicting (2).

**Exercise 3**

The two formulae:

(4) a. \( \forall x.\forall y.((P y x) \rightarrow (P x y)) \)
    b. \( \forall x.\forall y.((Q_1 y x) \rightarrow (Q_2 x y)) \)

**Model:**

\[
\begin{align*}
P &= \{(a, b), (b, a)\} \\
Q_1 &= \{(c, d)\} \\
Q_2 &= \{(d, c)\}
\end{align*}
\]

**NL Relations:** We have to find a symmetric relation for \( P \), and instances of \( Q_1, Q_2 \) which are converses of each other. For example, \( P \Rightarrow \text{‘is married to’}, Q_1 \Rightarrow \text{‘mother of’}, Q_2 \Rightarrow \text{‘child of’} \).

**Exercise 4**

**Predicates**

<table>
<thead>
<tr>
<th>Type</th>
<th>Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{IND} \rightarrow \text{BOOL} )</td>
<td>block, cube, pyramid, red, green, yellow, table</td>
</tr>
<tr>
<td>( \text{IND} \rightarrow (\text{IND} \rightarrow \text{BOOL}) )</td>
<td>larger, same-size, smaller, on</td>
</tr>
</tbody>
</table>

**Model**

\( M = (D, V) \) where:

- \( D = \{c_1, c_2, p_1, p_2, t\} \)
Suppose, instead, that it is interpreted as a set, say smaller understanding of the natural language expressions. In particular, Note that we have tried to construct a model which corresponds fairly intuitively to our un-

\[ V(block) = \{c_1, c_2, p_1, p_2\} \]
\[ V(cube) = \{c_1, c_2\} \]
\[ V(pyramid) = \{p_1, p_2\} \]
\[ V(red) = \{c_1, c_2\} \]
\[ V(green) = \{p_1\} \]
\[ V(yellow) = \{p_2\} \]
\[ V(table) = \{t\} \]
\[ V(larger) = \{(p_1, p_2)\} \]
\[ V(same-size) = \{(\langle c_1, c_2, c_1, c_1, c_1, c_1, c_2, c_2, p_1, p_1, p_1, p_1, p_2, p_2, p_2, p_2, t, t\rangle\} \]
\[ V(smaller) = \{(p_2, p_1)\} \]
\[ V(on) = \{(c_1, t), (c_2, t), (p_1, t), (p_2, c_1)\} \]

Note that we have tried to construct a model which corresponds fairly intuitively to our understanding of the natural language expressions. In particular, smaller is a relation, not a set. Suppose, instead, that it is interpreted as a set, say \( S = \{p_2\} \), with the assumption that this means that \( p_2 \) is smaller than \( p_1 \), which is not in the set \( S \). The problem is that this won’t generalize to three elements. Suppose \( p_3 \) is smaller than \( p_2 \), which in turn is smaller than \( p_1 \). So \( p_2 \) is in \( S \) because it is smaller than \( p_1 \) but it is also not in \( S \) because it is bigger than \( p_3 \).

**Formulae**

Now here is a possible description of the scenario:

\[
\exists x_0 \exists x_1 \exists x_2 \exists x_3 \exists x_4 \left( \text{block}(x_0) \land \text{block}(x_1) \land \text{block}(x_2) \land \text{block}(x_3) \land \text{table}(x_4) \land \\
 x_0 \neq x_1 \land x_0 \neq x_2 \land x_0 \neq x_3 \land \\
x_1 \neq x_2 \land x_1 \neq x_3 \land \\
x_2 \neq x_3 \land \\
\forall y(\text{block}(y) \rightarrow (y = x_0 \lor y = x_1 \lor y = x_2 \lor y = x_3)) \land \\
\text{cube}(x_0) \land \text{cube}(x_1) \land \text{pyramid}(x_2) \land \text{pyramid}(x_3) \land \\
\text{same-size}(x_0, x_1) \land \text{larger}(x_2, x_3) \land \\
\text{red}(x_0) \land \text{red}(x_1) \land \text{green}(x_2) \land \text{yellow}(x_3) \land \\
\text{on}(x_0, x_4) \land \text{on}(x_1, x_4) \land \text{on}(x_2, x_4) \land \text{on}(x_3, x_0) \lor \text{on}(x_3, x_1) \right)
\]

We can be somewhat more succinct by making it one big conjunction, but it can also be split up into smaller formulae.

However, this description doesn’t do justice to our intuitive understanding of the sets and relations in the scenario. So we should look at adding some ‘background knowledge’ to supplement the description:

\[
\forall x \forall y(\text{larger}(x, y) \leftrightarrow \text{smaller}(y, x)) \\
\forall x \forall y((\text{larger}(x, y) \lor \text{smaller}(x, y)) \leftrightarrow \neg \text{same-size}(x, y)) \\
\forall x(\text{same-size}(x, x)) \\
\forall x \forall y(\text{same-size}(x, y) \rightarrow \text{same-size}(y, x)) \\
\forall x(\text{block}(x) \rightarrow (\text{cube}(x) \leftrightarrow \neg \text{pyramid}(x))) \\
\forall x(\text{block}(x) \leftrightarrow \neg \text{table}(x)) \\
\forall x \forall y(\text{on}(x, y) \leftrightarrow \neg \text{on}(y, x)) \\
\forall x(\neg \text{on}(x, x))
\]

**Alternative Models**

An alternative vocabulary would not include same-size, instead defining it in terms of larger and smaller. Similarly, one could remove smaller, and replacing it with ‘\( \neg \text{larger} \)’, or vice versa. The above model respects the background knowledge given above, but if the model didn’t have
to satisfy this background knowledge, then the extensions of larger and smaller need not be anti-
symmetric, and the extension of same-size need not be an equivalence relation. We could also
increase the extensions of the colour predicates (making some objects both red and yellow, for
instance). We could remove \((p_2, c_1)\) from \(V\) and replace it with \((p_2, c_2)\). And finally, we could
add further individuals to the model that aren’t blocks or tables.

**Exercise 5**

Let’s assume that (5-a) is represented as (5-b) in standard FOL, or equivalently as (5-b) in NLTK-
style LF.

(5) a. Suzie shows John a dog.
   b. \(\exists x (\text{dog}(x) \land \text{show}(\text{suzie}, x, \text{john}))\)
   c. some x.((\text{dog} x) and (\text{show } j x \text{ suzie}))

Let’s also assume that the LF for shows John a dog is derived by combining shows John with a
dog.

Now, let’s figure out the lambda terms by working top-down from the root of the sentence. On the
left-hand side of the following tables we show the (type-raised) functor NP, and on the right-hand
side, the argument. So here’s how the subject combines with the VP:

<table>
<thead>
<tr>
<th>Suzie</th>
<th>shows John a dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\P.(\P \text{suzie}))</td>
<td>(\y.\text{some } x.((\text{dog} \ x) \text{ and } (\text{show } j x \ y)))</td>
</tr>
</tbody>
</table>

Down at the VP level, we need to combine a dog with shows John. So we need to pull out the
quantifier phrase and make it a functor:

<table>
<thead>
<tr>
<th>a dog</th>
<th>shows John</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\P.\text{some } x.((\text{dog} \ x) \text{ and } (\P \ x)))</td>
<td>(\x y.(\x. (\text{show } j x \ y)))</td>
</tr>
</tbody>
</table>

And down another level, we pull out the NP argument John, and make it a functor. So the
expression on the right-hand side below is the representation we want for the verb shows:

<table>
<thead>
<tr>
<th>John</th>
<th>shows</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\P.\text{(P j)})</td>
<td>(\Z X y. (X \x. (\Z z. (\text{show } z x y))))</td>
</tr>
</tbody>
</table>

There are two main ways of expressing the syntactic rule for ditransitives. If we stick to using
app\((a, b)\) as the semantic form of the mother, then we can only use binary syntactic rules:

# binary branching ditransitive VP
\(\text{VP}[\text{num}=?n, \text{sem}=<\text{app}(?\text{dtvp},?\text{obj})>] \to \text{DTV}[\text{num}=?n, \text{sem}=?\text{dtvp}] \text{NP}[\text{sem}=?\text{obj}]\)
\(\text{DTV}[\text{num}=?n, \text{sem}=<\text{app}(?\text{v},?\text{obj})>] \to \text{DTV}[\text{num}=?n, \text{sem}=?\text{v}] \text{NP}[\text{sem}=?\text{obj}]\)
\(\text{DTV}[\text{num}=\text{sg}, \text{sem}=<\Z X y. (X \x. (\Z z. (\text{show } z x y))), \text{tns}=\text{pres}] \to \text{’shows’}\)
\(\text{DTV}[\text{num}=\text{pl}, \text{sem}=<\Z X y. (X \x. (\Z z. (\text{show } z x y))), \text{tns}=\text{pres}] \to \text{’show’}\)

However, we are allowed to use more complex values in the semantics of the mother, such as the
following:

# ternary branching ditransitive VP
\(\text{VP}[\text{num}=?n, \text{sem}=<((?\text{dtvp} ?\text{obj1}) ?\text{obj2})>] \to \text{DTV}[\text{num}=?n, \text{sem}=?\text{dtvp}] \text{NP}[\text{sem}=?\text{obj1}] \text{NP}[\text{sem}=?\text{obj2}]\)

Here is a model, with an evaluation of the sentence:
from nltk.sem import *
val = Valuation({
    'suzie': 's',
    'john': 'j',
    'dog': {'f': True},
    'show': {'j': {'f': {'s': True}}}
})
dom = val.domain
m = Model(dom, val)
g = Assignment(dom)
e = 'some x.((dog x) and ((show john x) suzie))'
print m.evaluate(e, g)
# True

Exercise 6

Here are LFs for the two readings:

(6) Every person needs a doctor.
    a. $\exists y (\text{doctor}(y) \land \forall x (\text{person}(x) \rightarrow \text{need}(x, y)))$
    b. $\forall x (\text{person}(x) \rightarrow \exists y (\text{doctor}(y) \land \text{need}(x, y)))$

Let’s consider the models $M_1$, $M_2$ defined as follows: $M_1 = \langle D, V_1 \rangle$ where:

- $D = \{p_1, p_2, d_1, d_2\}$
- $V_1(\text{person}) = \{p_1, p_2\}$
- $V_1(\text{doctor}) = \{d_1, d_2\}$
- $V_1(\text{need}) = \{(p_1, d_1), (p_2, d_1)\}$

$M_2 = \langle D, V_2 \rangle$ where:

- $D = \{p_1, p_2, d_1, d_2\}$
- $V_2(\text{person}) = \{p_1, p_2\}$
- $V_2(\text{doctor}) = \{d_1, d_2\}$
- $V_2(\text{need}) = \{(p_1, d_1), (p_2, d_2)\}$

(We have assumed, for simplicity, that doctors and people are disjoint!)

Now $M_1 \models (6\text{-a})$, since $d_1$ is a doctor such that every person (i.e., $p_1$, $p_2$) needs him/her. Moreover $M_1 \models (6\text{-b})$. By contrast $M_2 \models (6\text{-b})$ but $M_2 \not\models (6\text{-a})$. So $M_1$ satisfies both readings, and $M_2$ satisfies only one reading, $(6\text{-b})$. So $\forall M$, if $M \models (6\text{-a})$ then $M \models (6\text{-b})$, but not conversely. So we can conclude that $(6\text{-a})$ entails $(6\text{-b})$.

Exercise 7

Tableaux expansion rules:
\[
T_{\text{nor}}: \quad \frac{T(\phi \text{ nor } \psi)}{T\phi} \quad \frac{F\phi}{F\psi}
\]

\[
F_{\text{nor}}: \quad \frac{F(\phi \text{ nor } \psi)}{T\phi} \quad \frac{F\psi}{T\psi}
\]

Proof of \(\neg(p \lor q) \leftrightarrow (p \text{ nor } q)\). We do it in two stages.

→ direction:

1. \(F(\neg(p \lor r) \rightarrow (p \text{ nor } q)) \quad \checkmark\)
2. \(T(p \text{ nor } q)) \quad 1, F_{\text{nor}}, \checkmark\)
3. \(F(p \text{ nor } q)) \quad 1, F_{\text{nor}}, \checkmark\)
4. \(F(p \lor r)) \quad 2, T_{\text{nor}}, \checkmark\)
5. \(Fp \quad 4, F_{\text{nor}}\)
6. \(Fq \quad 4, F_{\text{nor}}\)
7. \(Tp \quad 3, F_{\text{nor}}\)

← direction:

1. \(F(p \text{ nor } q) \rightarrow \neg(p \lor r) \quad \checkmark\)
2. \(T(p \text{ nor } q)) \quad 1, F_{\text{nor}}, \checkmark\)
3. \(F\neg(p \lor r) \quad 1, F_{\text{nor}}, \checkmark\)
4. \(Fp \quad 2, T_{\text{nor}}\)
5. \(Fq \quad 2, T_{\text{nor}}\)
6. \(T(p \lor r)) \quad 3, F_{\text{nor}}, \checkmark\)
7. \(Tp \quad 6, T_{\text{nor}}\)

You can combine these two conditionals into a single tableau by using the following rule for biconditionals:

\[
F_{\text{nor}}: \quad \frac{F(\phi \text{ nor } \psi)}{T\phi} \quad \frac{F\phi}{T\psi}
\]

Exercise 8

In the following grammar, the event variable \(e\) gets passed through the rules by successive \(\lambda\)-abstraction and \(\beta\)-conversion. In general — if we ignore type-raising over \(NP\) arguments — the basic type of intransitive verbs become \(\text{IND} \rightarrow \text{IND} \rightarrow \text{BOOL}\), i.e., binary relations over individuals and events. As a consequence, we change the type of \(NPs\) from \((\text{IND} \rightarrow \text{BOOL}) \rightarrow \text{BOOL}\) to \((\text{IND} \rightarrow \text{BOOL}) \rightarrow \text{BOOL}\), i.e., a function from binary relations to sets. This means that the event variable of a \(VP\) that contains a quantified \(NP\) object will still be available for adverbial modification. Finally, the semantic representation of an \(S\), e.g., the value of \(?\text{subj } ?\text{vp}\), will be of the form \(\\e. \phi\). Existential quantification of the event variable takes place as a supplementary step in the semantics for building \(S\) from \(NP\) and \(VP\).

% start S
\[
S[\text{sem} = \langle\text{some } e.((?\text{subj } ?\text{vp}) e)\rangle] \rightarrow \text{NP}[\text{num}=?n, \text{sem}=?\text{subj}] \ \text{VP}[\text{num}=?n, \text{sem}=?\text{vp}]
\]
\[
\text{NP}[\text{num}=?n, \text{sem}=\text{app}(?\text{det}, ?\text{nom})] \rightarrow \text{Det}[\text{num}=?n, \text{sem}=?\text{det}] \ \text{Nom}[\text{num}=?n, \text{sem}=?\text{nom}]
\]
\[
\text{NP}[\text{loc}=?l, \text{num}=?n, \text{sem}=?\text{np}] \rightarrow \text{PropN}[\text{loc}=?l, \text{num}=?n, \text{sem}=?\text{np}]
\]

6


PropN[-loc,num=sg,sem=<\R e. (R john e)>] -> 'John'
PropN[-loc,num=sg,sem=<\R e. (R mary e)>] -> 'Mary'
PropN[-loc,num=sg,sem=<\R e. (R suzie e)>] -> 'Suzie'
PropN[-loc,num=sg,sem=<\R e. (R fido e)>] -> 'Fido'
PropN[-loc, num=sg,sem=<\P.(P noosa)>>] -> 'Noosa'

NP[-loc, num=sg,sem=<\P.\x.(P x)>>] -> 'who'

Det[num=sg,sem=<\P R e. all x. ((P x) implies (R x e))>>] -> 'every'
Det[num=pl,sem=<\P R e. all x. ((P x) implies (R x e))>>] -> 'all'
Det[sem=<\P R e. some x. ((P x) and (R x e))>>] -> 'some'
Det[num=sg,sem=<\P R e. some x. ((P x) and (R x e))>>] -> 'a'

N[num=sg,sem=<boy>] -> 'boy'
N[num=pl,sem=<boy>] -> 'boys'
N[num=sg,sem=<girl>] -> 'girl'
N[num=pl,sem=<girl>] -> 'girls'
N[num=sg,sem=<dog>] -> 'dog'
N[num=pl,sem=<dog>] -> 'dogs'

TV[num=sg,sem=<\X y. (X \x e. ((agent y e) and ((chase e) and (patient x e)))))> , tns=pres] -> 'chases'
TV[num=pl,sem=<\X y. (X \x e. ((agent y e) and ((chase e) and (patient x e)))))> , tns=pres] -> 'chase'
TV[num=sg,sem=<\X y. (X \x e. ((agent y e) and ((see e) and (patient x e)))))> , tns=pres] -> 'sees'
TV[num=pl,sem=<\X y. (X \x e. ((agent y e) and ((see e) and (patient x e)))))> , tns=pres] -> 'see'
IV[num=sg,sem=<\x e. ((agent x e) and (bark e)))> , tns=pres] -> 'barks'
IV[num=pl,sem=<\x e. ((agent x e) and (bark e)))> , tns=pres] -> 'bark'
IV[num=sg,sem=<\x e. ((agent x e) and (walk e)))> , tns=pres] -> 'walks'
IV[num=pl,sem=<\x e. ((agent x e) and (walk e)))> , tns=pres] -> 'walk'

Adv[sem=<\R x e. ((slow e) and (R x e))>>] -> 'slowly'
Adv[sem=<\R x e. ((thoughtful e) and (R x e))>>] -> 'thoughtfully'

Since the NLTK LogicParser is set up to only recognize x, y, z as individual variables, we can modify the S rule along the following lines to ensure that the resulting LFs can be parsed:

S[sem = <some x.((event x) and (((?subj ?vp) e)))> ] -> NP[num=?n,sem=?subj] VP[num=?n,sem=?vp]

Sample model:

from nltk.sem import *
v = ***
suzie => s
john => j
dog => {f}
girl => {s}
boy => {j}
event => {e1, e2, e3, e4, e5, e6}
walk => {e1, e2, e3}
slow => {e2}
see => {e4, e5, e6}
agent => {((e1, f), (e2, j), (e3, s), (e4, s), (e5, j), (e6, f))
patient => {((e4, j), (e5, f), (e6, s))}

val = parse_valuation(v)
dom = val.domain
m = Model(dom, val)
g = Assignment(dom)

expr = '\ z. some x. ((event z) and ((boy x) and ((agent suzie z) and ((see z) and (patient x z)))

print m.evaluate(expr, g)
#True