# Exercises in Logic and Knowldege Representation

Brandon Bennett

October 6, 2014

## Contents

# 1 Translating from English into First-Order Logic

## Batch 1

Formulate the following English sentences as formulae in classical 1st-order logic (answers are given at the end of this document).

1. All purple mushrooms are poisonous.
2. No student likes every lecture.
3. Everest is the highest mountain on Earth.
4. There are at least two apples in a barrel.
5. There are at least two apples in every barrel.
6. $x$ is part of $y$ just in case everything that is connected to $x$ is also connected to $y$ (this is a fundamental definition used in certain theories of spatial relations).
7. No region is part of each of two disjoint regions.
8. Nothing can be inside two different boxes unless one box is inside the other. (Quite hard)

## Batch 2

1. Jane ate a mushroom that she had picked herself.
2. No yellow frogs are edible.
3. All students that had missed a lecture answered at least one question incorrectly.
4. Young creatures who go up in balloons are liable to giddiness.
5. Every bag contains at least one coin.
6. The father of a mother or father is a grandfather.
7. Tom's sister knows Mary's brother.
8. A careless soldier killed himself.
9. No happy girl despises all animals.
10. Every boy baked at least two cakes.
11. Two regions overlap just in case they share a common part.
12. If three convex regions all overlap each other they share a common part.
13. If a cake is from France then it has more sugar if it is made with chocolate than when it is made with cream but if a cake is from Italy then it has more sugar if it is made with cream than if it is made of chocolate".

## Batch 3

1. Every person loves themself.
2. Everyone loves some person who loves themself.
3. All gardeners like the sun.
4. Some person loves no one except themself.
5. Toby loves everyone who loves him.
6. Love is never requited. (Expand out the meaning of requited.)
7. Brown frogs are bigger than green frogs.
8. No natural food is blue.
9. Mary gave an apple to Tom.
10. One of the apples that Mary gave to Tom is Rotten.

## Sentences that are Difficult to Represent

Some sentences whose logical form cannot easily be captured in 1st-order logic:

1. The exam was hard but not very hard.
   (How should one represent the modifier 'very'?)
2. You only live twice. (Adverbs are not easily formalised)
3. By this time tomorrow I shall have finished my exams.
   (Complex temporal expressions hard to formalise.)

# 2 Translation into Logic — Answers

Here are some possible answers for some, but not all of the translation examples. There will generally be many acceptable translations of any given English sentence. The main reasons for this are the following:

## Batch 1

1. $\forall x[(\mathsf{Mushroom}(x) \wedge \mathsf{Purple}(x)) \rightarrow \mathsf{Poisonous}(x)]$

2. $\neg\exists x[\mathsf{Student}(x) \wedge \forall y[\mathsf{Lecture}(y) \rightarrow \mathsf{Likes}(x, y)]]$

3. $\mathsf{Mountain}(\mathsf{everest}) \wedge \neg\exists x[\mathsf{Mountain}(x) \wedge \mathsf{Higher}(x, \mathsf{everest}) \wedge \mathsf{On}(x, \mathsf{earth})]$

4. $\exists x\exists y\exists z[\mathsf{Apple}(x) \wedge \mathsf{Apple}(y) \wedge \neg(x = y) \wedge \mathsf{Barrel}(z) \wedge \mathsf{In}(x, z) \wedge \mathsf{In}(y, z)]$

5. $\forall z\exists x\exists y[\mathsf{Barrel}(z) \rightarrow (\mathsf{Apple}(x) \wedge \mathsf{Apple}(y) \wedge \neg(x = y) \wedge \mathsf{In}(x, z) \wedge \mathsf{In}(y, z))]$

   (Note that the universal quantification over barrels comes before the existential quantifiers over apples. If we had $\exists x\exists y\forall z[\ldots]$ this would mean that the same two apples were present in every barrel.

6. $\mathsf{P}(x, y) \leftrightarrow \forall z[\mathsf{C}(z, x) \rightarrow \mathsf{C}(z, y)]$
   (In this formula, the connective $\alpha \leftrightarrow \beta$ is logical equivalence and means the same as $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.)

7. $\neg\exists x\exists y\exists z[\mathsf{Reg}(x) \wedge \mathsf{Reg}(y) \wedge \mathsf{Reg}(z) \wedge \mathsf{P}(x, y) \wedge \mathsf{P}(x, z) \wedge \mathsf{Disjoint}(y, z)]$

8. $\forall x\forall b_1\forall b_2[(\mathsf{Box}(b_1) \wedge \mathsf{Box}(b_2) \wedge \neg(b_1 = b_2) \wedge \mathsf{In}(x, b_1) \wedge \mathsf{In}(x, b_2)) \rightarrow (\mathsf{In}(b_1, b_2) \vee \mathsf{In}(b_2, b_1))]$
   (There are many equivalent alternative formulations; but there are also many similar formulae which are incorrect.)

## Batch 2

1. Jane ate a mushroom that she had picked herself.
   $\exists x[\mathsf{Mushroom}(x) \wedge \mathsf{Ate}(\mathsf{jane}, x) \wedge \mathsf{Picked}(\mathsf{jane}, x)]$

2. No yellow frogs are edible.
   $\neg\exists x[\mathsf{Frog}(x) \wedge \mathsf{Yellow}(x) \wedge \mathsf{Edible}(x)]$

3. All students that had missed a lecture answered at least one question incorrectly.

4. Young creatures who go up in balloons are liable to giddiness.
   $\forall x[(Yy \wedge Cx \wedge Bx) \rightarrow Gx]$
   (Illustrates a concise representation for the logical form of the sentence.)

5. Every bag contains at least one coin.
   $\forall x[\mathsf{Bag}(x) \rightarrow \exists y[\mathsf{Coin}(y) \wedge \mathsf{Contains}(x, y)]]$

6. The father of a mother or father is a grandfather.
   This is quite tricky and it is debatable what is the best representation.
   One possibility is to represent all the family terms using relations:
   $\forall x \forall y[\ \exists z[\mathsf{IsFatherOf}(x,z) \wedge (\mathsf{IsMotherOf}(z,y) \vee \mathsf{IsFatherOf}(z,y))] \leftrightarrow \mathsf{IsGrandfatherOf}(x,y)]$
   Also we could treat some of the terms as single-argument properties:
   $\forall x \exists y[\mathsf{IsFatherOf}(x,y) \wedge (\mathsf{Father}(y) \vee \mathsf{Mother}(y))] \leftrightarrow \mathsf{Grandfather}(x)]$
   But then the connection between the Father property and the IsFatherOf relation is not made explicit. We could make this explicit with an additional formula:
   $\forall x[\mathsf{Father}(x) \leftrightarrow \exists y[\mathsf{IsFatherOf}(x,y)]]$
   (Similar definitions could be given for the Mother and Grandfather properties.)

7. Tom's sister knows Mary's brother.
   $\exists x \exists y[\mathsf{IsSisterOf}(x,\mathsf{tom}) \wedge \mathsf{IsBrotherOf}(y,\mathsf{mary}) \wedge \mathsf{Knows}(x,y)]$

8. A careless soldier killed himself.
   $\exists x[\mathsf{Soldier}(x) \wedge \mathsf{Careless}(x) \wedge \mathsf{Killed}(x,x)]$

9. No happy girl despises all animals.
   $\neg \exists x[\mathsf{Happy}(x) \wedge \mathsf{Girl}(x) \wedge \forall y[\mathsf{Animal}(y) \rightarrow \mathsf{Dispises}(x,y)]]$

10. Every boy baked at least two cakes.
    $\forall x[\mathsf{Boy}(x) \rightarrow \exists y \exists z[\neg(y=z) \wedge \mathsf{Baked}(x,y) \wedge \mathsf{Baked}(x,z)]]$

11. Two regions overlap just in case they share a common part.
    $\forall x \forall y[(\mathsf{Reg}(x) \wedge \mathsf{Reg}(y)) \rightarrow (\mathsf{Overlap}(x,y) \leftrightarrow \exists z[\mathsf{Part}(z,x) \wedge \mathsf{Part}(z,y)])]$

12. If three convex regions all overlap each other they share a common part.
    ????

13. If a cake is from France then it has more sugar if it is made with chocolate than when it is made with cream but if a cake is from Italy then it has more sugar if it is made with cream than if it is made of chocolate".
    ????

## Batch 3

1. Every person loves themself.
   $\forall x[\mathsf{Person}(x) \rightarrow \mathsf{Loves}(x,x)]$

2. Everyone loves some person who loves themself.
   $\forall x[\mathsf{Person}(x) \rightarrow \exists y[\mathsf{Person}(y) \wedge \mathsf{Loves}(x,y) \wedge \mathsf{Loves}(x,x)]]$

3. All gardeners love the sun.
   $\forall x[\mathsf{Gardener}(x) \rightarrow \mathsf{Loves}(x,\mathsf{sun})]$
   (Since 'sun' refers to a unique object, it is most straightforwardly represented by a constant (sun).)

4. ...

# 3 Questions on First-Order Models

1. Consider a fist order language whose vocabulary consists of a single binary relation, $R$, and four models: $\mathcal{M}_1 = \langle \mathcal{D}, \delta_1 \rangle$, $\mathcal{M}_2 = \langle \mathcal{D}, \delta_2 \rangle$, $\mathcal{M}_3 = \langle \mathcal{D}, \delta_3 \rangle$ and $\mathcal{M}_4 = \langle \mathcal{D}, \delta_4 \rangle$, which all have the same domain of individuals, $\mathcal{D} = \{a, b, c\}$). The denotation functions for each of the models are as follows:

   $\mathcal{M}_1$: $\delta_1(R) = \{\langle a, a\rangle,\ \langle a, b\rangle,\ \langle a, c\rangle,\ \langle b, a\rangle,\ \langle b, b\rangle,\ \langle b, c\rangle,\ \langle c, a\rangle,\ \langle c, b\rangle,\ \langle c, c\rangle\}$
   $\mathcal{M}_2$: $\delta_2(R) = \{\langle b, a\rangle,\ \langle b, b\rangle,\ \langle b, c\rangle\}$
   $\mathcal{M}_3$: $\delta_2(R) = \{\langle a, b\rangle,\ \langle b, a\rangle,\ \langle c, a\rangle\}$
   $\mathcal{M}_4$: $\delta_3(R) = \{\langle a, a\rangle\}$

   For each of the following formulae, state which of the models $\mathcal{M}_1$ – $\mathcal{M}_4$ satisfy it, if any:

   (a) $\exists x[R(x, x)]$
   (b) $\forall x[R(x, x)]$
   (c) $\forall x \forall y[R(x, y)]$
   (d) $\forall x \exists y[R(x, y)]$
   (e) $\exists x \forall y[R(x, y)]$
   (f) $\exists x \exists y[R(x, y)]$

2. $\mathcal{M} = \langle \mathcal{D}, \delta \rangle$ is a model for a first-order language with a unary predicate $P$ and a binary relation $T$. The domain of $\mathcal{M}$ is the set $\{a, b, c, d\}$; and the denotations of $P$ and $T$ are as follows:

   - $\delta(P) = \{a, c\}$
   - $\delta(T) = \{\langle a, b\rangle, \langle b, c\rangle, \langle c, d\rangle, \langle d, a\rangle\}$

   Which of the following formulae are satisfied by this model:

   (a) $\exists x[T(x, x)]$
   (b) $\exists x \forall y[T(x, y)]$
   (c) $\forall x \exists y[T(x, y)]$
   (d) $\forall x \forall y \forall z[(T(x, y) \wedge T(y, z)) \rightarrow T(x, z)]$
   (e) $\forall x \forall y[(P(x) \wedge T(x, y)) \rightarrow \neg P(y)]$

# 4 Automated Reasoning Exercises

## 4.1 Using the *Prover9* Theorem Prover

A wide variety of automated theorem provers have been developed since their origin in the 1960's. However, this kind of program is difficult to implement and often difficult to use. Thus, very few theorem provers have gained many users beyond the researchers who developed them. One of the more popular was *Otter*, which at one time was arguably the most powerful prover around, and was able to verify a number of theorems that mathematicians had hypothesised but not been able to prove. *Prover9* is an improved version of the original *Otter*. Although it is not usually regarded as one of the most powerful, it does have the advantage of being relatively simple compared to most of the more modern provers.
Documentation and downloads can be found on the *Prover9* home page at:

http://www.cs.unm.edu/~mccune/prover9/

### Running Prover9 Executables on SoC Linux Machines

The Prover9 Linux executable can be found at

`/usr/local/prover9/bin/prover9`

The basic mode of operation of Prover9 is to read input from a file, containing formulae, which of course need to be written in a particular format. To run Prover9 on the file `myinput.p9` you would give the following command:[1]

```
/usr/local/prover9/bin/prover9 -f myinput.p9
```

To limit the proof search to a fixed time, say 10 seconds, use the form:

```
/usr/local/prover9/bin/prover9 -t 10 -f myinput.p9
```

This will write its output to standard output (usually the terminal). However, it is often convenient to redirect the output to a file. To put the output in the file `myoutput.p9` you would use the command:

```
/usr/local/prover9/bin/prover9 < myinput.p9 > myoutput.p9out
```

More details on the command line syntax can be found in the online Prover9 manual.

There is also a GUI front-end for Prover9. It is probably best to start off using the GUI, although once you get used to the syntax you may prefer to code the formulae using your favourite editor and then run the command line version.

The GUI version can be run using the path:

`/usr/local/prover9/prover9-mace4.py`

[1] Or equivalently `/usr/local/prover9/bin/prover9 < myinput.p9`

## Prover9's Proof Method

The primary mode of inference used by Prover9 is *resolution*. It repeatedly makes resolution inferences with the aim of detecting inconsistency by deriving a contradiction. As discussed in the lecture notes, resolution provides a complete proof procedure for detecting *inconsistency* of formulae that are expressed in *clausal form*.
Prover9 will first do some preprocessing on the input file to convert it into the form it uses for inferencing. First it negates the formula given as a goal.[2] It then translates all formulae into clausal form. (In some cases it will do some further pre-processing, but you do not need to worry about this.) Then it will compute inferences and by default write these standard output. Unless the input is very simple it will often generate a large number of inferences. If it detects an inconsistency it will stop and print out a proof consisting of the sequence of resolution rules that generated the inconsistency. It will also print out various statistics associated with the proof.

## Simple Examples

Here is an example of a very simple Prover9 file:

```
formulas(assumptions).
p & s.                  %  ‘&’ symbol is for conjunction ‘and’
p -> q.                 % ‘->’ symbol is for implication ‘implies’
q -> r.
end_of_list.

formulas(goals).
r | t.                  % ‘|’ symbol is for disjunction ‘or’
end_of_list.
```

If you are using the GUI version you don't need the lines declaring the formula list and goal list. You just enter the formulae into the appropriate boxes in the interface. Here is a slightly more complex example using quantifiers (borrowed from the introductory material on the Prover9 web site):

```
formulas(assumptions).
  all x all y (subset(x,y) <-> (all z (member(z,x) -> member(z,y)))).
end_of_list.

formulas(goals).
  all x all y all z (subset(x,y) & subset(y,z) -> subset(x,z)).
end_of_list.
```

[2] Actually Prover9 allows multiple goal formulae, but it is best to use just one, since the semantics of multiple goals is potentially confusing.

## A More Complex Example

Here is a much more complex example devised by Len Schubert. It was originally known as 'The Steamroller' because at the time (in the 80s) it was very difficult for automated theorem provers to solve (it would flatten them). However, it presents no problem for a modern theorem prover such as Prover9 running on a modern computer.

```
%               Schubert's "Steamroller" Problem
%
% Wolves, foxes, birds, caterpillars, and snails are animals,
% and there are some of each of them.
%
% Also there are some grains, and grains are plants.
%
% Every animal either likes to eat all plants or all animals much
% smaller than itself that like to eat some plants.
%
% Caterpillars and snails are much smaller than birds, which are much
% smaller than foxes, which are in turn much smaller than wolves.
%
% Wolves do not like to eat foxes or grains, while birds like to eat
% caterpillars but not snails.
%
% Caterpillars and snails like to eat some plants.
%
% Prove there is an animal that likes to eat a grain-eating animal.
% (where a grain eating animal is one that eats all grains)

formulas(assumptions).
all x (Wolf(x) -> animal(x)).
all x (Fox(x) -> animal(x)).
all x (Bird(x) -> animal(x)).
all x (Caterpillar(x) -> animal(x)).
all x (Snail(x) -> animal(x)).
all x (Grain(x) -> plant(x)).

exists x Wolf(x).
exists x Fox(x).
exists x Bird(x).
exists x Caterpillar(x).
exists x Snail(x).
exists x Grain(x).
```

Continued Overleaf.

```
% All animals either eat all plants or eat all smaller animals that eat some plants.

all x (animal(x) -> (all y (plant(y)->eats(x,y)))
                    |
                    (all z ( animal(z) &
                             Smaller(z,x) &
                             (exists u (plant(u)&eats(z,u)))
                            ->
                             eats(x,z)))).

all x all y (Caterpillar(x) & Bird(y) -> Smaller(x,y)).
all x all y (Snail(x) & Bird(y) -> Smaller(x,y)).
all x all y (Bird(x) & Fox(y) -> Smaller(x,y)).
all x all y (Fox(x) & Wolf(y) -> Smaller(x,y)).
all x all y (Bird(x) & Caterpillar(y) -> eats(x,y)).

all x (Caterpillar(x) -> (exists y (plant(y) & eats(x,y)))).
all x (Snail(x)       -> (exists y (plant(y) & eats(x,y)))).

all x all y (Wolf(x) & Fox(y) -> -eats(x,y)).
all x all y (Wolf(x) & Grain(y) -> -eats(x,y)).
all x all y (Bird(x) & Snail(y) -> -eats(x,y)).
end_of_list.

formulas(goals).
% "there is an animal that eats {an animal that eats all grains}".
exists x exists y ( animal(x) & animal(y) & eats(x,y) &
                    (all z (Grain(z) -> eats(y,z)))
                  ).
end_of_list.
```

## 4.2 Propositional Logic Examples

Enter *propositional* logic translations of the following arguments into *Prover9* (using Prover9's ASCII representation for the formulae). Then use Prover9's theorem proving capabilities to show that the arguments are valid:
Remember that when translating to propositional logic you often have to simplify and/or alter the structure of sentences. You need to look for underlying propositions (i.e. statements that are true or false) and recast each sentence in terms of these propositions combined by means of the truth-functional operators.

A)
1. If Alan did not lock the door he either forgot or lost the key.
2. If he lost the key he will be in trouble.
3. If he has a good memory he did not forget.
4. Alan has a good memory.

$\therefore$ Either Alan locked the door or he will be in trouble.

B)
1. Either the picture is valuable or it is a fake.
2. If it is valuable it is by Rembrandt.
3. If it is a fake there will be a police investigation.

$\therefore$ Either the picture is by Rembrandt or there will be a police investigation.

C)
1. Sunday is a Holiday.
2. Saturday and Sunday is the weekend.
3. On holidays the University is closed.
4. It is the weekend.
5. It is not Saturday

$\therefore$ The University is closed.

Note that in some contexts we can translate names of days or time periods into propositions. Thus, sometimes 'Saturday' can be translated as a proposition, i.e. as 'It is Saturday'.

D)
1. If Tarquin learns philosophy then he acquires wisdom and if he learns science he will advance human knowledge.
2. If he acquires wisdom, Tarquin will not die in misery.
3. If he advances human knowledge, his aspirations will be satisfied.
4. If Tarquin's aspirations are not satisfied he will die in misery.
5. Alas, Tarquin's aspirations will never be satisfied.

$\therefore$ Thus, he learns neither philosophy nor science.

## 4.3 Some Simple Syllogisms

The *Syllogisms* were probably the first forms of logical inference that were systematically investigated by humans. The Greeks originally studied arguments in the context of legal disputes and recognised that some (but not all) arguments are based on the *form* of sentences with which they are expressed. A syllogism is a basic form of argument in which the meanings of two assertions are used to derive a consequence. More specifically, the term 'syllogism' is usually applied to a class of inferences involving sentences constructed from two property terms (predicates) linked by quantifiers and possibly negations.
The Medieval logicians continued the study of syllogisms and gave them all names based on a coding of the quantifiers and negations into letters. To find out more about the valid syllogisms and the coding used to name them take a look at the Wikipedia article http://en.wikipedia.org/wiki/Syllogism.

### 4.3.1 Barbara

1. All Greeks are men.
2. All men are mortal.

$\therefore$ All Greeks are mortal.

### 4.3.2 Celarent

1. No reptiles have fur.
2. All snakes are reptiles.

$\therefore$ No snakes have fur.

### 4.3.3 Ferio

1. No homework is fun.
2. Some reading is homework.

$\therefore$ Some reading is not fun.

### 4.3.4 Baroco

1. All informative things are useful.
2. Some web sites are not useful.

$\therefore$ Some web sites are not informative.

## 4.4 Extended Syllogistic Reasoning

The basic syllogisms involve exactly two predicates in each sentence. But we can use similar reasoning principles when this is not the case.

A) 1. No old rabbits are greedy.
   2. All black rabbits are greedy.
   $\therefore$ No old rabbits are black.

B) 1. Young rabbits are greedy.
   2. Greedy animals are fat.
   3. Rabbits are animals.
   $\therefore$ All young rabbits are fat.

## 4.5 Invalid and Unsound Syllogisms

Here are a couple of Syllogisms that are *not* valid. Try proving them and see what happens.

A) 1. Some rhombuses are equilateral.
2. Some equilaterals are triangular.
$\therefore$ Some rhombuses are triangular.

B) 1. All polygons are shapes.
2. Some shapes are triangular.
$\therefore$ All polygons are triangular.

C) Note that an argument can be logically *invalid* even if the conclusion is true. The conclusion may happen to be true even though this is not a consequence of the premisses.

1. Some birds fly.
2. Some flying things eat nuts.
$\therefore$ Some birds eat nuts.

D) Conversely an argument can be logically *valid* even if the conclusion is false.

1. Some fish eat bananas.
2. All banana eaters enjoy backgammon.
$\therefore$ Some fish enjoy backgammon.

If an argument is valid but its conclusion is false, then at least one of its premisses must be false.
An argument with one or more false premisses is called *unsound*.

## 4.6 Reasoning with Properties and Individuals

A) 1. Socrates is Greek.
2. All Greeks are men.
3. All men are mortal.
$\therefore$ Socrates is mortal.

B) 1. Porky is a pig.
2. Pigs are animals.
3. All pigs are fat.
4. No fat animals run fast.
$\therefore$ Porky does not run fast.

C) 1. The Jabberwock has eyes of flame.
2. All creatures with eyes of flame whiffle and burble.
3. All creatures that whiffle or burble have claws.
$\therefore$ The Jabberwock has claws.

## 4.7 A Murder Mystery Problem

Translate the following sentences into 1st-order logic and put them in a file using Prover9 syntax.

1. Someone who lives in Dreadbury Mansion killed Aunt Agatha.
2. Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein.
3. A killer always hates his victim, and is never richer than his victim.
4. Charles hates no one that Aunt Agatha hates.
5. Agatha hates everyone except the butler.
6. The butler hates everyone not richer than Aunt Agatha.
7. The butler hates everyone Aunt Agatha hates.
8. No one hates everyone.
9. Agatha is not the butler.

Now use the Prover9 theorem prover to deduce who killed Aunt Agatha.
(Hint: try for each of the possibilities.)

## 4.8 Ballooning Pigs

Enter formulae into *Prover9* corresponding to the following sentences:[3]

1. All, who neither dance on tightropes nor eat penny-buns, are old.
2. Pigs, that are liable to giddiness, are treated with respect.
3. A wise balloonist takes an umbrella with him.
4. No one ought to lunch in public who looks ridiculous and eats penny-buns.
5. Young creatures, who go up in balloons, are liable to giddiness.
6. Fat creatures, who look ridiculous, may lunch in public, provided that they do not dance on tightropes.
7. No wise creatures dance on tightropes, if liable to giddiness.
8. A pig looks ridiculous, carrying an umbrella.
9. All, who do not dance on tightropes, and who are treated with respect are fat.

Use *Prover9* to determine that, from the propositions given in part (b), the following conclusion can be inferred:

*No wise young pigs go up in balloons.*

In order to prove this you will have to add two further formula to the assumptions. Each of these additional formula is a general axiom which captures a *necessary* inter-dependence between two of the concepts that occur in propositions 1-9. One of them is a general fact about pigs and the other is about age.

[3]This exercise is based on a problem set by Lewis Carroll (Author of *Alice in Wonderland*) in his book *Symbolic Logic*.

## 4.9 Illegible Letters

Encode and prove the following argument. You may assume that the *domain of discourse* consists of all the letters in the room.

1. All the dated letters in this room are written on blue paper.
2. None of them are in black ink except those that are written in the third person.
3. AI have not filed any of them that I can read.
4. None of them that are written on one sheet are undated.
5. All of them that are not crossed are in black ink. :
6. All of them written by Brown begin with "Dear Sir" :
7. All of them written on blue paper are filed. :
8. None of them written on more than one sheet are crossed.
9. None of them that begin with "Dear Sir" are written in third person.

$\therefore$ The letters written by Brown cannot be read.

## 4.10 Email Logic Clique

> Anna, Barry, Charles and Deborah are logic students. Each logic student who has received an email has sent an email to every logic student. Every email that was sent was received (by the correct recipient). Anna has sent an email to Barry.

Encode the preceeding statements in first-order logic and use an automated theorem prover to prove that:

Charles has received an email from Deborah.

## 4.11 Qualitative Spatial Reasoning (QSR)

In Qualitative Spatial Reasoning the spatial part relation is defined in terms of the connection relation as follows:

$$\forall x \forall y [\mathsf{P}(x,y) \leftrightarrow \forall z [\mathsf{C}(z,x) \rightarrow \mathsf{C}(z,y)]] \ .$$

Use a theorem prover to demonstrate that the part relation is *transitive*.

## 4.12 Bovinophobia

a) Encode the following argument in Prover9 syntax:

1. Cows are animals that eat grass.
2. I don't eat grass.
3. Apart from me, the only animals in my house are feline.
4. No animals ever take to me, except those in my house.
5. I detest animals that do not take to me.
6. When I detest an animal, I avoid it.
7. Every animal that loves to gaze at the moon is suitable for a pet.
8. No animals are carnivorous, unless they prowl at night.
9. Every feline eats some animal smaller than itself.
10. I am not the smallest animal in my house.
11. Only carnivores eat animals.
12. Cows are not suitable for pets.
13. Animals that prowl at night always love to gaze at the moon.

$\therefore$ I avoid cows.

b) Use a theorem prover to show that proposition 14 is a logical consequence of the other propositions.

c) Which premiss is redundant to the proof? (Hints: Take a look through the prover's output file. Test your answer by commenting out this proposition.)

d) Give a formula stating a property of cows, which when added to propositions 1–6 entails proposition 14, without the need for propositions 7–13. Test your answer.

## 4.13 Favourite Colours

a) Translate the following sentences into an appropriate encoding for the first-order logic theorem prover that you are using:

1. There are three children called Alfred, Betty and Charles.
   (Hint: you must explicitly ensure that different names refer to distinct individuals).
2. Every child is wearing either a hat, coat or a scarf.
3. Alfred is wearing a red hat.
4. Betty is wearing a green coat.
5. Charles is wearing a blue scarf.
6. Each child has a different favourite colour, which is either red, green, or blue.
7. No child is wearing anything that has their favourite colour.
8. Betty's favourite colour is red.

b) Use the theorem prover to determine what are the favourite colours of Alfred and Charles.

c) Which proposition is redundant to the proof?

d) By examining the theorem prover's output, give relevant information that for comparing the behaviour of the prover (e.g. run-time, inferences generated etc.) when generating a proof, with and without the redundant proposition.

## 4.14 A Geographic Example

### Exercise Specification

In this exercise you will analyse a set of natural language sentences expressing a variety of forms of geographic information and translate the core meaning into first-order logic.
You will then look at a further set of sentences, which intuitively follow as consequences of the first set. Your aim will be to demonstrate that given a suitable logical representation of the sentences, together with additional axioms specifying interconnections between the concepts that are involved, each of these consequences can be proved using a first-order logic theorem prover (i.e. *Prover9*).

### Step 1

Translate the following sentences into first-order logic and enter their representations into Prover9 (in the *assumptions* box).
Given information:

1. Leeds is a city located in Yorkshire.
2. Sheffield is a city located in Yorkshire.
3. Lancaster is a city located in Lancashire.
4. Lancaster has a port.
5. Leeds is the largest city in Yorkshire.
6. Yorkshire is a county of the United Kingdom.
7. Lancashire is a county of the United Kingdom.
8. Yorkshire borders Lancashire.

### Step 2

Think about why the following formulae are consequences of the ones you have just analysed. Then translate them into first-order logic and enter their representations into Prover9 but enter them into the *goals* box of the interface.
Consequences:

1. Sheffield is smaller than Leeds.
2. Leeds is located in the United Kingdom.
3. Leeds is not located in Lancashire.
4. Yorkshire is an administrative region.
5. Lancashire borders the sea.

**Step 3**

Try to identify axioms capturing logical relationships between the concepts in the previously given sentences that are necessary to derive the consequences from the given information.
Enter your axioms into the assumptions box. But keep them separate from the given information. For clarity, I advise that you use comments to give an informal description of each of your axioms. Comments can be given on lines starting with the'%' character.

**Step 4**

Try to prove each of the consequences from the combination of the initial information and your additional axioms.
To do this you should comment out all the goals apart from the one you wish to prove. Then press the 'Start' button to run Prover9. It should then tell you whether the goal follows from your formalisation of the initial information and axioms.
Use comments to make a note of which goals have been proved.
You don't need to worry about the form of the actual proofs as for the purpose of this exercise we are treating the theorem prover as a 'black box'.

## 4.15 UK Geography Reasoning

UK.1) The United Kingdom is an island country, spanning an archipelago including Great Britain, the northeastern part of the island of Ireland, and many small islands.

UK.2) Northern Ireland is the only part of the UK with a land border, sharing it with the Republic of Ireland.

UK.3) Apart from this land border, the UK is surrounded by the Atlantic Ocean, the North Sea, the English Channel and the Irish Sea.

**Consequences.**

The following propositions are either direct or indirect consequences of the above propositions about the United Kingdom:

UK-C.1) Great Britain is part of the United Kingdom.

UK-C.2) Great Britain consists of at least two islands.

UK-C.3) Great Britain is externally connected to an ocean.

UK-C.4) The United Kingdom is connected to the Republic of Ireland.

# 5 Natural Deduction

This section begins with a concise summary of the rules of a Natural Deduction proof system for propositional logic. Following that, I shall give a number of tips on how to construct proofs along with some examples. At the end of the section is the actual exercise, which is a list of sequents that you should try to prove using Natural Deduction.

## 5.1 The Rules

### Basic Rule Set

$$\frac{P \qquad Q}{P \wedge Q}\ \wedge\text{I} \qquad \frac{P \wedge Q}{P}\ \wedge\text{Er} \qquad \frac{P \wedge Q}{Q}\ \wedge\text{El}$$

$$\frac{P}{P \vee Q}\ \vee\text{Ir} \qquad \frac{Q}{P \vee Q}\ \vee\text{Il} \qquad \frac{A \vee B \qquad \begin{array}{c}[\ A\ ]_{na}\\ \vdots \\ P\end{array} \qquad \begin{array}{c}[\ B\ ]_{nb}\\ \vdots \\ P\end{array}}{P}\ \vee\text{E}_n$$

$$\begin{array}{c}[\ A\ ]_n\\ \vdots \\ P \wedge \neg P\end{array}\ \neg\text{I}_n \qquad \begin{array}{c}[\ \neg A\ ]_n\\ \vdots \\ P \wedge \neg P\end{array}\ \neg\text{E}_n$$

### Rules for $\rightarrow$ (can be derived via the definition of $\rightarrow$)

$$\begin{array}{c}[\ P\ ]_n\\ \vdots \\ Q\end{array}\ \rightarrow\text{I}_n \qquad \frac{P \qquad P \rightarrow Q}{Q}\ \rightarrow\text{E}$$

**Rules for $\bot$ (optional — will make some proofs shorter)**

$$\frac{P \qquad \neg P}{\bot}\ \bot\mathrm{I} \qquad \frac{\bot}{P}\ \bot\mathrm{E} \qquad \frac{\begin{array}{c}[\ A\ ]_n\\ \vdots\\ \bot\end{array}}{\neg A}\ \neg\mathrm{I}_n \qquad \frac{\begin{array}{c}[\ \neg A\ ]_n\\ \vdots\\ \bot\end{array}}{A}\ \neg\mathrm{E}_n$$

**Useful Derived Rules (derivable from the basic rule set)**

$$\frac{P \wedge \neg P}{A}\ EFSQ \qquad \frac{}{P \vee \neg P}\ EM \qquad \frac{\neg\neg P}{P}\ DNE$$

## 5.2 Derivations of Derived Rules

The following rules are quite fundamental, but can actually be derived using the basic introduction and elimination rules. Here are the proofs:

### *Ex falso sequitur quadlibet* (EFSQ)

From falisty (i.e. contradiction) follows everything.

$$\frac{\dfrac{\dfrac{P \wedge \neg P \qquad [\ \neg A\ ]_n}{(P \wedge \neg P) \wedge \neg A}\ \wedge\mathrm{I}}{P \wedge \neg P}\ \wedge\mathrm{Er}}{A}\ \neg\mathrm{E}_n \qquad \Longrightarrow \qquad \frac{P \wedge \neg P}{A}\ EFSQ$$

### Law of Excluded Middle (EM)

$$\frac{\dfrac{\dfrac{\dfrac{[\ P\ ]_1}{P \vee \neg P}\ \vee\mathrm{Ir} \qquad [\ \neg(\mathsf{P} \vee \neg P)\ ]_2}{(P \vee \neg P) \wedge \neg(P \vee \neg P)}\ \wedge\mathrm{I}}{\neg P}\ \neg\mathrm{I}_1}{P \vee \neg P}\ \vee\mathrm{Il} \qquad \frac{[\ \neg(P \vee \neg P)\ ]_2}{(P \vee \neg P) \wedge \neg(P \vee \neg P)}\ \wedge\mathrm{I}}{P \vee \neg P}\ \neg\mathrm{E}_2$$

$$\Longrightarrow \qquad \frac{}{P \vee \neg P}\ \mathrm{EM}$$

## Double Negation Elimination (DNE)

$$\dfrac{\dfrac{[\,\neg P\,]_1 \qquad \neg\neg P}{P \wedge \neg P}\,\wedge\mathrm{I}}{P}\,\neg\mathrm{E}_1 \qquad \Longrightarrow \qquad \dfrac{\neg\neg P}{P}\,DNE$$

## Modus Ponens (MP) — using equivalent $\neg A \vee B$ in place of $A \rightarrow B$

$$\dfrac{\neg A \vee B \qquad \dfrac{\dfrac{A \qquad [\,\neg A\,]_1}{A \wedge \neg A}\,\wedge\mathrm{I}}{B}\,EFSL \qquad [\,B\,]_2}{B}\,\vee\mathrm{E}_{1,2} \qquad \Longrightarrow \qquad \dfrac{A \qquad \neg A \vee B}{B}\,MP$$

## Conditional Proof (CP)

$$\dfrac{\dfrac{}{P \vee \neg P}\,EM \qquad \dfrac{\begin{array}{c}[\,P\,]_1\\ \vdots\\ Q\end{array}}{\neg P \vee Q}\,\vee\mathrm{Il} \qquad \dfrac{[\,\neg P\,]_2}{\neg P \vee Q}\,\vee\mathrm{Ir}}{\neg P \vee Q}\,\vee\mathrm{E}_{1,2} \qquad \Longrightarrow \qquad \dfrac{\begin{array}{c}[\,P\,]_n\\ \vdots\\ Q\end{array}}{\neg P \vee Q}\,\mathrm{CP}_n$$

## 5.3 Natural Deduction Tips and Examples

### When and What to Assume

One aspect of Natural Deduction proofs that you need to get the hang of is when and what to assume. There are actually four different reasons why an assumption may be made. Each of these is described and illustrated in this document.

**Case 1. Assume the premisses**
You should always assume all the formulae that are given as premisses (i.e. on the left hand side) of the sequent you wish to prove.

For example, to prove

$$P,\ P \to Q,\ Q \to R\ \vdash\ R$$

you should assume $P$, $P \to Q$ and $Q \to R$, then try to derive $R$:

$$\dfrac{\dfrac{P \qquad P \to Q}{Q}\ \to\text{E} \qquad Q \to R}{R}\ \to\text{E}$$

**Case 2. If you have a premiss $P \vee Q$, assume $P$ then assume $Q$**
If you have a disjunctive premiss, you can often prove the conclusion by successively assuming each disjunct and trying to prove the conclusion from each. Then you can conclude the proof by using the $\vee$E rule.

For example:

$$A,\ B \vee C,\ \vdash\ (A \wedge B) \vee (A \wedge C)$$

$$\dfrac{B \vee C \qquad \dfrac{\dfrac{A \qquad [\ B\ ]_{1a}}{A \wedge B}\ \wedge\text{I}}{(A \wedge B) \vee (A \wedge C)}\ \vee\text{Ir} \qquad \dfrac{\dfrac{A \qquad [\ C\ ]_{1b}}{A \wedge C}\ \wedge\text{I}}{(A \wedge B) \vee (A \wedge C)}\ \vee\text{Il}}{(A \wedge B) \vee (A \wedge C)}\ \vee\text{E}_1$$

(Note that the assumption $A$ occurs twice. Proofs often contain duplicates of the assumptions. This just means that the assumption takes part in more than one rule application.)

**Case 3. To Prove $P \rightarrow Q$ assume $P$**
If the conclusion you want to prove is of the form $P \rightarrow Q$, you should assume $P$ (along with any premisses) and try to prove $Q$. You can then use the rule $\rightarrow$E to discharge the assumption $P$ and derive $Q$.
For example, to prove

$$A \rightarrow B,\ B \rightarrow C \ \vdash\ A \rightarrow C$$

assume $A$ (as well as the premisses $A \rightarrow B$ and $B \rightarrow C$). Then prove $C$; and finally prove $A \rightarrow C$ by $\rightarrow$E:

$$\dfrac{\dfrac{\dfrac{[\ A\ ]_1 \qquad A \rightarrow B}{B}\ \rightarrow\text{E} \qquad B \rightarrow C}{C}\ \rightarrow\text{E}}{A \rightarrow C}\ \rightarrow\text{I}_1$$

**Case 4. Assume the opposite of what you want to prove**
A final tactic that you may need to use is to assume the negation of the conclusions you want to prove. The idea is to then derive a contradiction to show that the assumption must be false. One can then use either $\neg$E or $\neg$I to derive a conclusion that either removes or adds a negation to the formula that was assumed. The assumption itself is then discharged.
This mode of proof using $\neg$I is typically where the formula to be proved is negated:

$$\neg(P \vee Q)\ \vdash\ \neg P$$

$$\dfrac{\dfrac{\dfrac{[\ P\ ]_1}{P \vee Q}\ \vee\text{Ir} \qquad \neg(P \vee Q)}{(\mathsf{P} \vee Q) \wedge \neg(P \vee Q)}\ \wedge\text{I}}{\neg P}\ \neg\text{I}_1$$

But a similar form of proof using $\neg$E can be employed to prove an un-negated conclusion:

$$\neg\neg P\ \vdash\ P$$

$$\dfrac{\dfrac{[\ \neg P\ ]_1 \qquad \neg\neg P}{\neg P \wedge \neg\neg P}\ \wedge\text{I}}{P}\ \neg\text{E}_1$$

## 5.4 Natural Deduction Exercises

Construct Natural Deduction proofs of each of the following sequence, using any of the rules in the Basic Rule Set and the Rules for $\rightarrow$ (as specified in Section 5.1 above). All the given sequents are valid and therefore provable (since this rule set is complete for propositional logic reasoning).
Answers to cases a–f are given below.

a) $P \wedge Q \vdash P \wedge (Q \vee R)$

b) $P \rightarrow Q,\ \neg Q \vdash \neg P$

c) $P \rightarrow Q \vdash \neg Q \rightarrow \neg P$

d) $P \vee Q,\ P \rightarrow R,\ Q \rightarrow R \vdash R$

e) $P \rightarrow (\neg Q \vee R),\ P,\ Q \vdash R$

f) $P \wedge (Q \vee R) \vdash (P \wedge Q) \vee R$

g) $P \vdash \neg\neg P$

h) $P \vee Q,\ P \rightarrow R,\ Q \rightarrow R \vdash R$

i) $P \vdash Q \rightarrow P$

j) $\vdash (P \vee Q) \rightarrow (Q \vee P)$

k) $\vdash P \rightarrow (P \vee Q)$

l) $\neg(P \vee Q) \vdash \neg P$

m) $\neg(P \vee Q) \vdash \neg P \wedge \neg Q$

n) $P \wedge Q \vdash \neg(\neg P \vee \neg Q)$

o) $P \rightarrow Q \vdash \neg Q \rightarrow \neg P$

p) $(P \wedge Q) \vee (P \wedge R) \vdash P \wedge (Q \vee R)$

q) $\neg R,\ (P \vee Q) \rightarrow R \vdash \neg P$

r) $P \vee (Q \vee R) \vdash (P \vee Q) \vee R$

### 5.4.1 Answers

a)
$$\dfrac{\dfrac{P \wedge Q}{P}\,\wedge\text{Er} \qquad \dfrac{\dfrac{P \wedge Q}{Q}\,\wedge\text{El}}{Q \vee R}\,\vee\text{Ir}}{P \wedge (Q \vee R)}\,\wedge\text{I}$$

b)
$$\dfrac{\dfrac{\dfrac{[\ P\ ]_1 \qquad P \to Q}{Q}\,\to\text{E} \qquad \neg Q}{Q \wedge \neg Q}\,\wedge\text{I}}{\neg P}\,\neg\text{I}_1$$

c)
$$\dfrac{\dfrac{\dfrac{\dfrac{[\ P\ ]_1 \qquad P \to Q}{Q}\,\to\text{E} \qquad [\ \neg Q\ ]_2}{Q \wedge \neg Q}\,\wedge\text{I}}{\neg P}\,\neg\text{I}_1}{\neg Q \to \neg P}\,\to\text{I}_2$$

d)
$$\dfrac{P \vee Q \qquad \dfrac{[\ P\ ]_{1a} \qquad P \to R}{R}\,\to\text{E} \qquad \dfrac{[\ Q\ ]_{1b} \qquad Q \to R}{R}\,\to\text{E}}{R}\,\vee\text{E}_1$$

e)
$$\dfrac{\dfrac{P \qquad P \to (\neg Q \vee R)}{\neg Q \vee R}\,\to\text{E} \qquad \dfrac{\dfrac{Q \qquad \dfrac{\dfrac{[\ \neg Q\ ]_{2a} \qquad [\ \neg R\ ]_1}{\neg Q \wedge \neg R}\,\wedge\text{I}}{\neg Q}\,\wedge\text{Er}}{Q \wedge \neg Q}\,\wedge\text{I}}{R}\,\neg\text{E}_1 \qquad [\ R\ ]_{2b}}{R}\,\vee\text{E}_2$$

f)

$$\dfrac{\dfrac{P \wedge (Q \vee R)}{Q \vee R}\,\wedge\text{El} \qquad \dfrac{\dfrac{\dfrac{P \wedge (Q \vee R)}{P}\,\wedge\text{Er} \qquad [\ Q\ ]_{1a}}{P \wedge Q}\,\wedge\text{I}}{(P \wedge Q) \vee R}\,\vee\text{Ir} \qquad \dfrac{[\ R\ ]_{1b}}{(P \wedge Q) \vee R}\,\vee\text{Il}}{(P \wedge Q) \vee R}\,\vee\text{E}_1$$

# 6 Theorem Proving with Binary Resolution

By default, *Prover9* carries out a wide variety of different kinds of inference. Most of these are varieties of inference, but they may combine several resolution steps together (e.g. *Hyper-resoluton*) or they only operate in restricted cases (for instance when the result of the inference is particularly simple).[4] Often a combination of these muti-step and restricted application rules can perform better than using the basic *binary resolution* rule. However, depending on which rules are chosen, the resulting system may or may not provde a *complete* inference system.
To see how the simple *binary resolution* rule works, it is easiest if we run Prover9 from the command line rather than using the GUI. That way we have full control over the input file that the prover will use and can directly specify which rules we want it to do.
As a first example, put the following into a text file (let us call the file `binres_test.p9`):

```
set(raw).                % Clear all default rules
set(binary_resolution). % Add binary resolution rule
set(factor).             % Add factoring rule

formulas(assumptions).
A | B.
-A | -C | D.
A | D.
-A |  C.
-B | C.
-C | -D.
end_of_list.
```

The first line removes all the default inference rules and the following two add binary resolution and factoring rules. (For propositional reasoning we do not actually need to set the factoring fule, because Prover9 always merges identical literals within each new formula that it infers; but it is needed to get a complete inference system for first-order logic (i.e. if we have formula with quantifiers.)
Now from the command prompt (in the same directory as the input file) enter:[5]

```
/usr/local/prover9/bin/prover9 < restest.p9
```

Now look at the output. You can skip past the initial processing and search sections. The actual proof should be near the end of the file. Here you see that a series of resolution steps has been applied, resulting in a final cancellation of a literal with its negation. This results in the so-called 'empty clause', which in Prover9 is denoted by `$F`. This shows that the assumptions are inconsistent.

[4] An example of this is so-called *unit resulting (UR) resolution*, which is only used when the infered formula consists of a single literal.

[5] You may want to redirect the output to a file by adding `> output_file` at the end of the command.

## 6.1 Clausification and Conclusion Negation

In the previous example, we gave Prover9 a set of formulae all in clausal form form and with no conclusion that we wanted to prove. In this case the system simply applies binary resolution repeatedly. This gives us a complete test for consistency of the assumptions: the empty clause will be derived if and only if the empty clause `$F` can be derived.

In the following example the input is not quite so simple. For one thing, the assumptions are given in terms of arbitrary propositional formulae, not clauses. This means that, before using resolution, Prover9 must convert the formulae into clausal form. For another thing, we have a 'goal', i.e. a conclusion we want to prove. Since resolution detects inconsistency, we add the *negation* of the goal to the assumptions. If the negation of the goal is logically inconsistent with the other assumptions then the goal must be a logical consequence of the premisses.

Run Prover9 on the following example file (in the same way as for the previous example):

```
set(raw).                % Clear all default rules
set(binary_resolution). % Add binary resolution rule
set(factor).             % Add factoring rule

formulas(assumptions).
p & s.                  %  '&' symbol is for conjunction 'and'
p -> q.                 % '->' symbol is for implication 'implies'
q -> r.
end_of_list.

formulas(goals).
r | t.                  % '|' symbol is for disjunction 'or'
end_of_list.
```

If you look through the output, you will see how Prover9 first converts the input into a set of clauses by adding the the negation of the goal to the assumption and converting all formulae to clausal form. It then proceeds to compute resolution inferences in order to construct a proof. The proof consists of all the assumptions that are required for the proof together with all inferred formulae that contribute to the chain of inference leading to the derivation of `$F`.